TECHNICAL REPORT NO. 198

The Analysis of Algorithms
Errata and Answers to the Exercises

by

Paul W. Purdom

Revised: January 2008

COMPUTER SCIENCE DEPARTMENT
INDIANA UNIVERSITY
Bloomington, Indiana 47405-4101

Answers to the Exercises

Particular thanks goes to my coauthor, Cynthia Brown, and to the members of the 1985 analysis of algorithms class at Indiana University, Myung-Ki Choi, Moon-Sung Han, Rhys Price Jones, Cheng-Chee Koh, Verghis Koshi, Chao-Tai Mong, You-Fang Pan, Brad A. Pierce, Shing Shong (Bruce) Shei, and Wan-Cheng Wang, who found many corrections to the book and who provided answers to some of the exercises.

1–1. You should use the polynomial time algorithm because it is faster. (Normally the polynomial algorithm will be better even for moderate values of n, but if the constants associated with the polynomial time algorithm are particularly large and the constants associated with the exponential algorithm are particular small, then the exponential algorithm may be faster for moderate n.)

1.1–2. The time is likely to be too small to measure. Thus if the algorithm needs 20 microseconds and the computer has a clock that ticks each millisecond, it will be difficult or impossible to measure the time. This can sometimes be overcome by putting the straight-line code in a loop and doing it 10,000 times (one needs to compare the time for the loop with no code inside to the time for the loop with the code inside). On time sharing systems, several runs of the same program are likely to lead to different measured times.

1.1–3. Hopefully the two methods will give about the same time. If not, there are several possibilities to consider. You may have made a mistake in one of the two methods. The instruction timings in the manual may be wrong. The computer's clock may not measure what it claims to measure. The computer's clock may not measure what you think it does.

1.2.1–1. $7n+8$. (Step 1 has two assignments. It is done once. Step 2 has three arithmetic operations and two assignments. It is done $n+1$ times. Step 3 has one comparison and one goto. The comparison is done $n+1$ times, the goto n times. Step 4 has no operations of the type you were asked to count.)

1.3.1–1. $7n+8$.

1.3.1–2. $7n+12$. [Step 1 has one assignment. Step 2 has one assignment, one comparison, one arithmetic operation, and one goto, but the goto is done only once while the rest of the operations are done $n+2$ times. (There are plausible arguments that the assignment and/or the arithmetic operation could be done $n+1$ times, which would lead to a total time of $7n+10$ or $7n+11$.) Step 3 has two arithmetic operations and one assignment. Step 4 has one goto.]

1.4–1. The value of W is $d_n b^n + d_{n-1}b^{n-1} + \cdots + d_0 b^0$. The value of W mod b^k is congruent to $d_{k-1}b^{k-1} + d_{k-2}b^{k-2} + \cdots + d_0$ because $W = (d_n b^{n-k} + d_{n-1}b^{n-k-1} + \cdots + d_k b^0)b^k + d_{k-1}b^{k-1} + d_{k-2}b^{k-2} + \cdots + d_0$ where the first term is an integer times a number divisible by b^k. Now the size of $d_{k-1}b^{k-1} + d_{k-2}b^{k-2} + \cdots + d_0$ is between 0 (obtained by setting each d_i to its smallest possible value) and

$(b^{k-1}+b^{k-2}+\cdots+b^0)(b-1)$ (obtained by setting each d_i to its largest possible value). The sum of the geometric series $b^{k-1}+b^{k-2}+\cdots+b^0$ is $(b^k-1)/(b-1)$ (see section 2.4), so the upper limit is b^k-1, which is less than b^k. Since $d_{k-1}b^{k-1}+d_{k-2}b^{k-2}+\cdots+d_0$ is between 0 and b^k-1 and since it is congruent to W mod b^k, it must be equal to W mod b^k.

1.4–2. The value of W/b^k is $(d_nb^{n-k}+d_{n-1}b^{n-k-1}+\cdots+d_kb^0)+(d_{k-1}b^{k-1}+d_{k-2}b^{k-2}+\cdots+d_0)/b^k$. The first term is an integer and the second term is between 0 and 1 (but not equal to 1) (see the previous solution) so the integer part of W/b^k is equal to the first term.

1.4–3. Prove that $\lfloor -x\rfloor = -\lceil x\rceil$. Let $x=n+\epsilon$ where n is an integer and $0\le\epsilon<1$. If $\epsilon=0$, then $\lfloor -x\rfloor = -x = -\lceil x\rceil$. If $\epsilon>0$, then $\lfloor -x\rfloor = -n-1 = -(n+1) = -\lceil x\rceil$.

1.4.1–1. $12n+5$. Step 1 takes 1 time unit, Step 2 takes $3n+4$, Step 3 takes $3n$, Step 4 takes $5n$ if you assume one unit each for the divide, the mod, the integer part, and each of the assignments (the instruction sets on most computers suggest $3n$ is also a reasonable answer for this step, since it usually takes one instruction to compute the integer part of the quotient and the remainder), and Step 5 takes n. In counting the time for Step 2, we are assuming that the normal case (continuing the loop) has an add, a store, and a compare. When we fall out of the loop we have all that plus a goto. (If we skip the add on the first time of the loop, then the answer is reduced by one, but you probably need one more go to to do this, resulting in no change from the given answer.) The justification of the answer is as important as the exact value obtained.

1.4.2–1. Let $a=q_am+r_a$, $b=q_bm+r_b$, $x=q_xm+r_x$, and $y=q_ym+r_y$, where each q is an integer and each r is between 0 and m (but not equal to m). Since $a\equiv b \pmod m$, $r_a=r_b$. Likewise $r_x=r_y$. Now $(a+x)$ mod m equals r_a+r_x if $r_a+r_x<m$ and r_a+r_x-m if $r_a+r_x\ge m$. Likewise $(b+y)$ mod m equals r_b+r_y if $r_b+r_y<m$ and r_b+r_y-m if $r_b+r_y\ge m$. Thus $(a+x)$ mod $m=(b+y)$ mod m, so $a+x\equiv b+y \pmod m$. The proofs for subtraction and multiplication are similar to the proof for addition.

1.4.2–2. $ax\equiv by \pmod m$ implies that $ax=by+im$ for some integer i. $a\equiv b \pmod m$ implies that $a=b+jm$ for some integer j. Replacing b in the first equation gives $ax=(a-jm)y+im$, or $a(x-y)=(i-jy)m$. The right side of this last equation is a multiple of m, so the left side must be also. If a is relatively prime to m this is possible only if $x-y$ is a multiple of m. But if $x-y$ is a multiple of m, then $x\equiv y \pmod m$.

1.4.2–3. $x\equiv a \pmod p$ implies $x=a+ip$, $y\equiv b \pmod p$ implies $y=b+jp$, and $y\not\equiv 0 \pmod p$ implies $y=c+kp$ for some $c\ne 0$, where i, j, and k are integers. To have $x/y\equiv a/b \pmod p$, we need $(a+ip)/(b+jp)=a/b+kp$ or $ab+ibp=ab+jap+b^2kp+jbp^2$ for any prime p any integers a, b, i, j, and some integer k. Solving for k gives $k=(ib-ja)/(b^2-jp^2)$, which is not always an integer. (For example $a=1$, $b=2$, $i=1$, $j=1$, $p=3$ gives $k=-1/5$.)

1.4.2–4. Since for any integer k, $b^k=(b^{k-1}+b^{k-2}+\cdots+b^0)(b-1)+1$, where the first term is an integer times $(b-1)$, $b^k\equiv 1 \pmod{b-1}$. Using this with eq. 15 gives $d_kb^k\equiv d_k \pmod{b-1}$. Using this with eq. 13 gives the final result.

1.4.2–5. Use $b^k=(b^{k-1}-b^{k-2}+\cdots\pm b^0)(b+1)\mp 1$, where the upper signs apply for even k and the lower signs apply for odd k. Proceed as in the previous exercise.

1.4.3–1. 67231.

1.4.3–2. $1^{-1} \equiv 1$, $2^{-1} \equiv 5$, $4^{-1} \equiv 7$, $5^{-1} \equiv 2$, $7^{-1} \equiv 4$, $8^{-1} \equiv 8$.

1.5–1. $\lg(3/2) = \lg 3 - \lg 2 = \lg 3 - 1$ so $1 + \lg(3/2) = \lg 3$.

1.5–2. Since $\lfloor \lg n \rfloor \le \lg n$, $2^{\lfloor \lg n \rfloor} \le 2^{\lg n} = n$. The lower limit is more difficult to prove. Let $n = 2^k + j$, where $0 \le j < 2^k$. Then $2^{\lfloor \lg n \rfloor} = 2^k$ (which does not depend on j), while $(n+1)/2 = (2^k + j + 1)/2$. The largest value for $(n+1)/2$ is obtained by setting j to its largest value, $2^k - 1$. So the largest that $(n+1)/2$ can be (for a fixed k) is $(2^k + 2^k)/2 = 2^k$. Thus $(n+1)/2 = 2^{\lfloor \lg n \rfloor}$ for n one less than a power of 2, while $(n+1)/2 < 2^{\lfloor \lg n \rfloor}$ for all other values of n.

1.5–3. $2^{2 \lg x} = 2^{\lg x^2} = x^2$.

1.5–4. Let $y = 2^{\frac{\ln\left(1+\frac{t(-\ln(1-p))}{\ln 2}\right)}{-\ln(1-p)}}$. Then $\ln y = \frac{\ln 2 \ln\left(1+\frac{t(-\ln(1-p))}{\ln 2}\right)}{-\ln(1-p)} = \ln\left(1+\frac{t(-\ln(1-p))}{\ln 2}\right)^{\frac{\ln 2}{-\ln(1-p)}}$, so $y = \left(1+\frac{t(-\ln(1-p))}{\ln 2}\right)^{\frac{\ln 2}{-\ln(1-p)}}$. See the following problem.

1.5–5. Let $z = x^{\ln y}$. Then $\ln z = \ln x \ln y = \ln y^{\ln x}$. Raising e to the $\ln z$ power gives $z = y^{\ln x}$.

1.5–6. $n(3/2)^{\lg n} = n3^{\lg n}/2^{\lg n} = 3^{\lg n}$. From the previous exercise we have that $3^{\lg n} = n^{\lg 3}$.

1.5.1–1. If we assume that the array is initialized so that x_0 is smaller than any other element, then we can change the Binary Search Algorithm (Algorithm 1.6), so that starting at Step 5 it reads:

Step **5.** If $q = x_b$ then exit.

Step **6.** For $i \leftarrow n$ down to $b+1$ do:

Step **7.** $x_{i+1} \leftarrow x_i$.

Step **8.** $x_{b+1} \leftarrow q$.

If q can be smaller than any element in the table, then we should change Step 5 to

Step **5.** If $q = x_b$ then exit; otherwise if $q < x_0$, then $b \leftarrow -1$.

1.5.1–2. Using subscripts to distinguish various times around the loop, at the end of Step 4 of the Euclidean Algorithm we have $c_i = d_{i-1}$, $d_i = r_{i-1}$, and $c_i = q_i d_i + r_i$, where $0 \le r_i < d_i$ and $q_i \le 1$ for $i \ge 1$ (if we did not have the condition that $n < m$, then the first time through the loop, q_i could be zero). Considering consecutive values of i gives $r_i = q_{i+2} r_{i+1} + r_{i+2}$ and $r_{i+1} = q_{i+3} r_{i+2} + q_{i+2} r_{i+3}$. Combining these two equations gives $r_i = (q_{i+2} q_{i+3} + 1) r_{i+2} + q_{i+2} r_{i+3}$. Since $r_{i+3} \ge 0$, $r_i \ge (q_{i+2} q_{i+3} + 1) r_{i+2}$. Since $q_{i+1} q_{i+2} \ge 1$, $r_i \ge 2 r_{i+2}$. Thus, except for the first time through the loop, every two times through the loop results in r going down by at least a factor of 2. Using $i = 0$ to index the initial time through the loop, when we go through the loop an even number of times we have $r_0 < n$, which gives $2^i r_{2i} < n$, $r_{2i} < n 2^{-i}$, $r_{2i} < 1$ for $i \ge \lg n$, or $r_k < 1$ for $k \ge 2 \lg n$. The algorithm does not go around the loop again when $r < 1$. When the number of times around the loop is odd, it will be 1 more than what was just calculated.

1.6.2–1. The rest of the digits get initialized at Step 8.

1.6.2–2. The answer can never be too big. The biggest possible answer is obtained by squaring the biggest n-digit number. The result is $(b^n - 1)^2 = b^{2n} - 2b^n + 1$. The program has enough memory for any $2n$ digit number, so it can handle any number up to $b^{2n} - 1$. Thus, the amount to spare is $2b^n - 2$. For $b \geq 2$, this is positive for $n \geq 1$.

1.6.2–3. In Algorithm 1.8, change n to m in Steps 1, 4, and 8. In Step 9 change $2n$ to $m + n$. Also fix up the explanation by changing n to m as needed.

1.6.2–4. It uses $2n$ words for z, n words for x, and n words for y, for a total of $4n$ words. If one counts the storage for the single word variables, an additional 4 words are used, for a total of $4n + 4$ words (The words for simple variables are often not counted.) If you count two words for p (which needs to store values with twice as many digits as any other variable), then a total of $4n + 5$ words are used.

1.6.2–5. The algorithm normally taught in grade school uses $O(n^2)$ time and n^2+5n storage. The two initial numbers are written down ($2n$ digits), the partial products are written down (n rows of at most $n + 1$ digits), and the answer is written down (at most $2n$ digits). Some children use additional storage to write down carries during the addition. Some may say you should also count the $i - 1$ places at the right of the i^{th} row. Doing this increases the total by $n(n - 1)/2$.

1.6.4–1. Essentially the same calculation is being done if you replace the multiplication and addition in the Matrix Multiplication Algorithm with addition and minimization respectively. The order of the loops in Matrix Multiplication can be changed to agree with Shortest Path (but not vise versa). Finally, the initialization is a little different.

1.6.4–2. Step 1, $n + 1$. Step 2, $n^2 + n$. Step 3, $n^3 + n^2$. Step 4, n^3. Step 5, n^3. Step 6, n^2. Step 7, n.

1.7–1. 5050.

1.7.1–1. $(m+1)(n+1)$. Multiplications are done for each $k = i - j$ in the range $0 \leq k \leq n$. For each k one multiplication is done for each j in the range $0 \leq j \leq m$.

1.7.1–2. The two sums are the same (term by term) for the terms that are included in the sum of the previous exercise. The terms for $j < 0$ are zero because $a_j = 0$ for $j < 0$. The terms for $j < i - m$ are zero because b_{i-j} is zero for $i - j$ above m (when $j < i - m$, $i - j > m$). The terms for $j > n$ are zero because $a_j = 0$ for $j > n$. The terms for $j > i$ are zero because b_{i-j} is zero for $i - j < 0$. Thus the additional terms that might be in the sum for this exercise are all zero.

1.7.1–3. Algorithm 1.8 will do the job if all occurrences of b are replaced by infinity. (To obtain a reasonable algorithm, initialize all of z to zero, change Step 5 to $z_{i+j} \leftarrow x_i y_j + z_{i+j}$, and drop Steps 6 and 8.)

1.8–1. $1/n$.

1.8.1–1. (A) mutually exclusive. (B) mutually exclusive. (C) club is a subcase of black. (D) independent.

1.8.1–2. (A) 0. (B) 0. (C) 1/4. (D) $(1/4)(1/13) = 1/52$. (E) $1/13 + 1/13 = 2/13$. (F) $1/4 + 1/2 = 3/4$. (G) 1/2. (H) $1/4 + 1/13 - 1/52 = 4/13$.

1.8.2–1. $(1/2)^3 = 1/8$.

1.8.2–2. 3/8.

1.8.2–3. 1/6.

1.8.2–4. $(1/6)^n$.

1.8.2–5. $(1/2)^3 = 1/8$.

1.8.2–6. 3/4.

1.8.2–7. $(1/2)(9/19)(8/18) = 2/19$. Notice that this is a little smaller than the answer to Exercise 5.

1.8.2–8. You need anything except three reds or three blacks. $1 - 4/19 = 15/19$. Notice that this is a little larger than the answer to Exercise 6.

1.9–1. Since $T_{\text{worst case}} = \max_{1\le i\le k} T_i$ and $T_{\text{best case}} = \min_{1\le i\le k} T_i$, $T_{\text{best case}} \le T_i \le T_{\text{worst case}}$. Multiplying by p_i and summing gives $\sum_i p_i T_{\text{best case}} \le \sum_i p_i T_i \le \sum_i p_i T_{\text{worst case}}$. Factoring out $T_{\text{best case}}$ and $T_{\text{worst case}}$, using $\sum_i p_i = 1$ and the definition of average gives $T_{\text{best case}} \le T_{\text{average}} \le T_{\text{worst case}}$.

1.9–2. If the number of items is $2^k + j$ where $0 \le j < 2^k$, then $2^k - j$ of the items can be found in k searches and the other $2j$ items require $k+1$ searches. The average number of searches is

$$\frac{2^k - j}{n}k + \frac{2j}{n}(k+1) = k + \frac{2j}{n} = \lfloor \lg n \rfloor + 2\frac{j}{n} = \lfloor \lg n \rfloor + 2\frac{n - 2^{\lfloor \lg n \rfloor}}{n}.$$

Using $A = \lfloor \lg n \rfloor + 2(n - 2^{\lfloor \lg n \rfloor}/n$, the average time is $t_1 + t_2 + t_5 + A(t_2 + t_3 + t_4)$.

1.9–3. The average of the square of the number of searches is

$$\begin{aligned}\frac{2^k - j}{n}k^2 + \frac{2j}{n}(k+1)^2 &= k^2 + \frac{2j}{n}(2k+1) = \lfloor \lg n \rfloor^2 + 2\frac{j}{n}(2k+1) \\ &= \lfloor \lg n \rfloor^2 + 2\frac{n - 2^{\lfloor \lg n \rfloor}}{n}(2\lfloor \lg n \rfloor + 1).\end{aligned}$$

Subtracting the square of the average gives

$$\begin{aligned}&2\frac{n - 2^{\lfloor \lg n \rfloor}}{n}(2\lfloor \lg n \rfloor + 1) - 4\lfloor \lg n \rfloor \frac{n - 2^{\lfloor \lg n \rfloor}}{n} - 4\left(\frac{n - 2^{\lfloor \lg n \rfloor}}{n}\right)^2 \\ &\qquad = 2\left(\frac{n - 2^{\lfloor \lg n \rfloor}}{n}\right)\left(1 - 2\frac{n - 2^{\lfloor \lg n \rfloor}}{n}\right).\end{aligned}$$

Using $V = 2[(n - 2^{\lfloor \lg n \rfloor})/n][1 - 2(n - 2^{\lfloor \lg n \rfloor})/n]$, the variance of the time is $V(t_2 + t_3 + t_4)^2$.

1.9.1–1. The best case occurs when the elements are already sorted. In this case Step 4 is done $n-1$ times. It can clearly not be done a lesser number of times because it must be done at least once each time the outer loop is done.

1.10–1. $O(nmp)$. $t_5 nmp + O(nm)$.

1.10–2. The worst-case time is $\frac{1}{2}(t_3+t_4+t_5+t_6)n^2+(t_1+t_2+\frac{1}{2}t_3+\frac{1}{2}t_4+\frac{1}{2}t_5+\frac{1}{2}t_6+t_8+t_9)n-(t_2+t_3+t_8+t_9)$, with reasonable assumptions about the size of t_7. Regardless of the size of t_7, the worst-case time is $O(n^2)$ and it is $\frac{1}{2}(t_3+t_4+t_5+t_6)n^2+O(n)$. The average time is $O(n^2)$ and it is $\frac{1}{4}(t_3+t_4+t_5+t_6)n^2+O(n)$. The number of times that Step 4 is done is no more than the number of times that Step 3 is done and no less than the number of times that Step 3 is done minus $(n-1)$. Therefore we know that Step 4 is done $\frac{1}{4}n^2 + O(n)$, which is accurate enough to answer this problem.

1.10–3. If $f_i(n)$ is the function on row i, then $f_i(n) = O(f_j(n))$ for $j \geq i$, but $f_i(n) \neq O(f_j(n))$ for $j < i$.

1.10–4. $g(n) = O(f(n))$ is proved following eq. (77). To prove $f(n) = O(g(n))$, chose $\epsilon = a/2$ in eq. (78). This gives $||g(n)|/f(n) - a| < a/2$, which implies $|g(n)|/f(n) - a > -a/2$, or $|g(n)|/f(n) > a/2$. Since $g(n) \geq 0$, the absolute value signs on $g(n)$ can be dropped. This gives $f(n) < 2g(n)/a = O(g(n)$, since a is a positive constant.

1.11–1. The relation "congruent mod n" is reflexive because, for any x, we have $x \bmod n = x \bmod n$ so $x \equiv x \pmod{n}$. The relation is symmetric because, for any x and any y where $x \equiv y \pmod{n}$, we have $x \bmod n = y \bmod n$, which implies $y \bmod n = x \bmod n$, so $x \equiv y \pmod{n}$ implies that $y \equiv x \pmod{n}$. The relation is transitive because for any x, y, and z where $x \equiv y \equiv z \pmod{n}$, we have $x \bmod n = y \bmod n = z \bmod n$, which implies $x \bmod n = z \bmod n$, so $x \equiv y \equiv z \pmod{n}$ implies that $x \equiv z \pmod{n}$. Since the relation is reflexive, symmetric, and transitive, it is an equivalence relation.

1.11–2.

×	[0]	[1]	[2]	[3]	[4]
[0]	[0]	[0]	[0]	[0]	[0]
[1]	[0]	[1]	[2]	[3]	[4]
[2]	[0]	[2]	[4]	[1]	[3]
[3]	[0]	[3]	[1]	[4]	[2]
[4]	[0]	[4]	[3]	[2]	[1]

1.11–3.

+	[0]	[1]	[2]	[3]	[4]	[5]
[0]	[0]	[1]	[2]	[3]	[4]	[5]
[1]	[1]	[2]	[3]	[4]	[5]	[0]
[2]	[2]	[3]	[4]	[5]	[0]	[1]
[3]	[3]	[4]	[5]	[0]	[1]	[2]
[4]	[4]	[5]	[0]	[1]	[2]	[3]
[5]	[5]	[0]	[1]	[2]	[3]	[4]

×	[0]	[1]	[2]	[3]	[4]	[5]
[0]	[0]	[0]	[0]	[0]	[0]	[0]
[1]	[0]	[1]	[2]	[3]	[4]	[5]
[2]	[0]	[2]	[4]	[0]	[2]	[4]
[3]	[0]	[3]	[0]	[3]	[0]	[3]
[4]	[0]	[4]	[2]	[0]	[4]	[2]
[5]	[0]	[5]	[4]	[3]	[2]	[1]

1.11–4. Define $x \prec y$ if and only if either $|x| < |y|$ or $|x| = |y|$ with $x > y$. There are many other orderings. This orders numbers by their absolute value, with a positive

number being smaller than the negative number with the same absolute value. With this ordering zero is the smallest number.

1.11–5. Define $a_1a_2\ldots a_i \prec b_ib_2\ldots b_j$ if and only if either $i<j$ or $i=j$ with the a string preceding the b string in lexicographic ordering. This question has many correct answers.

1.11–6. Define

$$(m_1,n_1,p_1)\prec(m_2,n_2,p_2)\quad\begin{cases}\text{if } m_1<m_2,\\ \text{if } m_1=m_2 \text{ and } n_1<n_2,\\ \text{if } m_1=m_2,\ n_1=n_2, \text{ and } p_1<p_2.\end{cases}$$

This question has many correct answers.

1.12–1. Base case ($n=0$): $\sum_{1\le i\le n} i^2 = 0$ and $(2n^3+3n^2+n)/6 = 0$, so the result is true for $n=0$. Assume the result is true for $n-1$. Then $\sum_{1\le i\le n} i^2 = \sum_{1\le i\le n-1} i^2+n^2 = [2(n-1)^3+3(n-1)^2+(n-1)]/6+n^2 = (2n^3-6n^2+6n-2+3n^2-6n+3+n-1+6n^2)/6 = (2n^3+3n^2+n)/6$, so the result is also true up to n.

1.12–2. Base case ($n=1$): $\sum_{0\le i\le n} x^i = 1+x$ and $(x^{n+1}-1)/(x-1) = (x^2-1)/(x-1) = x+1$, so the result is true for $n=1$. Assume the result is true for $n-1$. Then $\sum_{0\le i\le n} x^i = \sum_{0\le i\le n-1} x^i + x^n = (x^n-1)/(x-1)+x^n = (x^{n+1}-1)/(x-1)$, so the result is true up to n.

1.12–3. Do induction on the length of the formula. The base case is formulas of one symbol. Such formulas consist of just one constant, so $n_0=1$ and $n_i=0$ for $i>0$. In this case the proposed formula reduces to $1=1$, which is true. Now suppose the formula has been proved for all lengths up to $k-1$. Consider a formula of length $k>1$. It consists of a main operator and subterms, say f_l and $t_1, t_2, \ldots, t_{d_l}$. Each subterm has length no more than $k-1$, so the formula holds for them. Let n_{ij} be the number of operators of degree i in subterm j. For each subterm t_j, we have $n_{0j}+n_{1j}+n_{2j}+\cdots+n_{Nj} = 1+n_{1j}+2n_{2j}+\cdots+Nn_{Nj}$. Also, by counting the number of operators of each degree, we have $n_i = \sum_{1\le j\le d_l} n_{ij}+\delta_{id_l}$. Adding the first equation over j, we obtain $\sum_{1\le j\le d_l} n_{0j}+\sum_{1\le j\le d_l} n_{1j}+\sum_{1\le j\le d_l} n_{2j}+\cdots+\sum_{1\le j\le d_l} n_{Nj} = d_l+\sum_{1\le j\le d_l} n_{1j}+\cdots+N\sum_{1\le j\le d_l} n_{Nj}$. Replacing each sum by the appropriate n_i (or $n_{d_l}-1$ when $i=d_l$) gives $n_0+n_1+n_2+\cdots+n_N-1 = d_l+n_1+2n_2+\cdots+Nn_N-d_l$, which simplifies to $n_0+n_1+n_2+\cdots+n_N = 1+n_1+2n_2+\cdots+Nn_N$. Thus, the result is true up to length k also.

1.12.1.2–1. In order to use eq. (96) to evaluate $C_{m,n-2^t}$, it is necessary for $n-2^t$ to be odd. It is also required that $m\le(n-1)/2-2^{t-1}$ [the condition on eq. (96) says that the second index minus one, divided by two, must be greater than or equal to the first index for the formula to apply]. This requirement can be written as $2m+1\le n-2^t$. Clearly this means that $2m\le n$, whatever the value of t is. But then, since $t=\lfloor\lg(n/m)\rfloor$, we have $t\ge 1$.

Now consider the possible values of t, and what conditions we need on m and n to make $2m+1\le n-2^t$. For $t=1$, we have $2m+1\le n-2$, or $2m+3\le n$. For $t\ge 2$, the fact that $t=\lfloor\lg(n/m)\rfloor\le\lg(n/m)$ implies $2^t\le 2^{\lg(n/m)}=n/m$, so $n-2^t\ge n(1-1/m)$. Also $n\ge 4m$ (since $t=\lfloor\lg(n/m)\rfloor\ge 2$), so $n-2^t\ge 4m(1-1/m)$. Thus for $t\ge 2$ the condition $2m+1\le n-2^t$ is satisfied when $2m\le 4m(1-1/m)-1$.

Dropping the -1 and factoring out $2m$, we have $1 \le 2 - 2/m$, or $m \ge 2$. Thus the use of eq. (95) on $C_{m,n-2^t}$ is okay if $m \ge 2$ and $n \ge 2m+3$, provided $n - 2^t$ is odd. But for $t \ge 1$, $n - 2^t$ is odd when n is odd, and $n \ge 2m+3$ implies that $t \ge 1$. Thus eq. (96) can be used on $C_{m,n-2^t}$ whenever $m \ge 2$, $n \ge 2m+3$, and n is odd.

It is okay to use eq. (96) on the other term in eq. (104) ($C_{m-1,n}$) provided n is odd and $2(m-1) \le n-1$, which is less restrictive than the condition in the last paragraph.

So we have that $C_{m,n} = 1 + \max\{C_{m,n/2-2^{t-1}} + m,\ t + C_{m-1,n/2} + m\} = 1 + \max\{C_{m,n/2-2^{t-1}},\ t + C_{m-1,n/2}\} + m$ provided n is odd, $m \ge 2$, and $n \ge 2m+3$. By eq. (103) $1 + \max\{C_{m,n/2-2^{t-1}},\ t + C_{m-1,n/2}\} = C_{m,n/2}$, so we have $C_{m,n} = C_{m,n/2} + m$. when n is odd, $m \ge 2$, and $n \ge 2m+3$, provided our induction hypothesis is true.

1.12.1.2–2. Suppose $m = 1$ and $n = 2^t + 1$ for some integer t. Then using eq. (103) gives $C_{1,2^t+1} = 1 + \max\{C_{1,1},\ t + C_{0,2^t+1}\} = 1 + t$. Use of eq. (96) on $C_{1,2^t+1}$ gives $1 + C_{1,2^t-1}$. Now we can use eq. (110). This gives $C_{1,2^t-1} = t$, which agrees with the previous result, so the proposed solution is working for $m = 1$, $n = 2^t + 1$.

1.12.1.2–3. When $n = 2m+1$, $t = 1$. Applying eq. (96) to the left side of eq. (103) gives $C_{m,2m+1} = C_{m,m} + m$. Applying eq. (97) gives $C_{m,2m+1} = 3m - 1$. The right side of eq. (103) is $1 + \max\{C_{m,2m-1},\ 1 + C_{m-1,2m+1}\}$. We can use eq. (97) to obtain $C_{m,2m-1} = 3m - 2$ and eq. (96) and eq. (97) to obtain $C_{m-1,2m+1} = C_{m-1,m} + m = 3m - 2$. Therefore in this case the right side of eq. (103) reduces to $3m - 1$, which is the same as the left side.

2.1–1. $\sum_{0 \le i \le \lfloor (n-1)/3 \rfloor} a_{3i+1}$ or $\sum_{\substack{1 \le i \le n \\ i \bmod 3 = 1}} a_i$.

2.1–2. $a_{n+1} - a_1$.

2.1–3. $[n(n+1)/2]^2$.

2.1–4. Let $S = \sum_{0 \le i \le n} x^i$. $xS - S = \sum_{0 \le i \le n} x^{i+1} - \sum_{0 \le i \le n} x^i = x^{n+1} - x^0$. (Most of the terms in the two sums cancel out). Solving for S gives $S = (x^{n+1} - 1)/(x - 1)$.

2.1.1–1. The probability that the step is done i times is $1/n$ for $0 \le i \le n-1$, so the average is $(1/n) \sum_{0 \le i \le n-1} i = (n-1)/2$.

2.1.1–2. $\sqrt{(n^2-1)/12}$. The variance of x_i and $x_i - 1$ are the same, so eq. (26) can be used.

2.1.2–1. $\sum_{i \ge 0} x^i = \lim_{n \to \infty} \sum_{i \ge 0} x^i = \lim_{n \to \infty} (x^{n+1} - 1)/(x-1) = (\lim_{n \to \infty} x^{n+1} - 1)/(x - 1)$. If $|x| < 1$, then $\lim_{n \to \infty} x^{n+1} = 0$ and the answer is $1/(1-x)$. If $x \ge 1$ the original sum is of positive increasing numbers, so the original sum diverges. If $x < -1$ the original sum diverges with the terms increasing in absolute value with oscillating signs. For $x = -1$ the original sum (up to n) oscillates with n.

2.1.2–2. The sum diverges when $|x| \ge 1$. For $|x| < 1$, $\sum_{i \ge 0} i x^i = \lim_{n \to \infty} \sum_{i \ge 0} i x^i = \lim_{n \to \infty} ((n-1)x^{n+1} - nx^n + x)/(x-1)^2 = (\lim_{n \to \infty} (n-1)x^{n+1} - \lim_{n \to \infty} nx^n + x)/(x-1)^2 = x/(x-1)^2$.

2.1.3–1. The sum is zero with either convention.

2.1.3–2. The sum is zero with the first convention and $-1 - 2 = -3$ with the second convention.

2.2–1. $\Sigma\{n^2\} = \{(2n^3 + 3n^2 + n)/6\}$.

2.2–2. $\mathbf{I}\{i^2\} = \{i^2\}$, $\mathbf{E}\{i^2\} = \{(i+1)^2\} = \{i^2+2i+1\}$, $\mathbf{\Delta}\{i^2\} = \{2i+1\}$, $\mathbf{\Sigma}\{i^2\} = \{(2i^3+3i^2+i)/6\}$, $\mathbf{\Delta E}\{i^2\} = \{2i+3\}$, $\mathbf{E\Delta}\{i^2\} = \{2i+3\}$, $\mathbf{\Sigma\Delta}\{i^2\} = \{i^2+2i+1\}$, $\mathbf{\Delta\Sigma}\{i^2\} = \{i^2+2i+1\}$.

2.2–3. $\mathbf{E}\{a_i\} = \{a_{i+1}\}$. $\mathbf{\Delta E}\{a_i\} = \{a_{i+2} - a_{i+1}\}$. $\mathbf{\Delta}\{a_i\} = \{a_{i+1} - a_i\}$. $\mathbf{E\Delta}\{a_i\} = \{a_{i+2} - a_{i+1}\}$. Thus for any sequence $\mathbf{\Delta E}$ gives the same result as $\mathbf{E\Delta}$.

2.3–1. $\sum_{1\le i\le n}(i+3)^2 = \sum_{4\le i\le n+3} i^2 = [2(n+3)^3 + 3(n+3)^2 + (n+3)]/6 - (2\cdot 3^3 + 3\cdot 3^2 + 3)/6 = (2n^3 + 21n^2 + 73n)/6$.

2.3–2. The number of multiplications for the clever way is $\sum_{1\le i\le n}\sum_{1\le k\le i}(i-k+1) = \sum_{1\le i\le n}[i(i+1) - i(i+1)/2] = \sum_{1\le i\le n} i(i+1)/2 = (2n^3+3n^2+n)/12 + n(n+1)/4 = (n^3+3n^2+2n)/6$. The number for the traditional algorithm is n^3. For large n, the clever method uses about $1/6$ as many multiplications. Using big O notation, the ratio is $1/6 + O(1/n)$.

2.3.1–1. You don't need the bottom one because the element x, which is taken from position x_l, serves as a sentinel. You do need the top one in the case where Step 6 sets i beyond r.

2.3.1–2. The time is $O(n)$. Assuming normal instruction timings, the worst case occurs when the elements are in reverse order and the splitting element is the middle most element. This causes the algorithm to move every element.

2.3.1–3. The worst case occurs when the file is always split so that one part has $n-1$ elements and the other part has none. Since the time to Split n elements is $O(n)$, the time for this case is given by $\sum_{1\le i\le n} O(i) = O(n^2)$.

2.3.1–4. The variation still has the case where one part has size $n-1$, so the worst-case time is the same as for the previous algorithm.

2.3.1–5. The worst case occurs when the first part has $n-1$ elements. In this case, $n-1$ stack cells of three words each are used, for a total of $3n-3$ words. The use of tail recursion does not help the worst-case space for this algorithm.

2.3.1–6. For the variation, the second part is always as large as the first part. When tail recursion is used, the worst case occurs when the two parts are equal. This results in the size of the first part decreasing by a little over a factor of two on each call, so the number of cells is no more than $\lg n$. (If you want an exact value, you need to use the technique of Section 5.2.1.) If tail recursion is not used, the worst case occurs when the second part has $n-1$ elements. In this case $n-1$ stack cells are needed. So, the alternate algorithm uses much less stack space, provided it is implemented with tail recursion. If tail recursion is not used, the variation uses the same space. It has the disadvantages of being slightly slower and slightly more complex.

2.4.1–1. $k/N > 1/2$.

2.4.1–2. $k/N = 1/2$.

2.4.1–3. $U_k = (1/k)\sum_{0\le i<k} T_i$.

2.4.1–4. $(d/dN)(N^2/(N-k)) = N(N-2k)/(N-k)^2$. To minimize, set the derivative to zero to obtain $N = 2k$. (The solution $N = 0$ is not suitable because we need at least k slots in the table.)

2.4.1–5. Proceeding as in the previous exercise, $(d/dN)[ab + aN + bN/(N-k) + N^2/(N-k)] = a - bk/(N-k)^2 + N(N-2k)/(N-k)^2$. $(a+1)N^2 - 2(a+1)kN +$

$ak^2 - bk = 0$. $N = [2(a+1)k + \sqrt{4(a+1)^2k^2 - 4(a+1)(ak^2 - bk}]/[2(a+1)] = k + \sqrt{k(k+b)/(a+1)}$.

2.4.1–6. There is no known way to generate an entirely random sequence of hash values. Still, the analysis is quite useful in suggesting how more realistic algorithms can be expected to behave.

2.4.1–7. One approach is to look up the item that you want to delete and delete it. Then start a series of linear scans of the table. Rehash each item in the table. If any item moves, do another scan. Continue until no items move. This clearly takes a time that is at least proportional to the size of the table. If the table is very full, then the time will be much worse.

2.5–1. $\sum_{i\geq 1} i^2x^i = \lim_{n\to\infty} \sum_{0\leq i\leq n} i^2x^i$. Now use summation by parts with $b_i = i^2$ and $a_i = x^i/(x-1)$ to obtain $\lim_{n\to\infty} \sum_{0\leq i\leq n} i^2x^i = \lim_{n\to\infty}[n^2x^n/(x-1) - \sum_{0\leq i\leq n}(2i+1)x^{i+1}/(x-1)] = \lim_{n\to\infty}\{n^2x^n/(x-1) - 2[(n-1)x^{n+2} - nx^{n+1} + x^2/(x-1)^3] - x(x^n-1)/(x-1)^2\}$. The original sum clearly diverges for $x \geq 1$ and for $x \leq -1$. For $-1 < x < 1$, the limit of x^n goes to zero faster than any power of n goes to infinity, so $\sum_{i\geq 1} i^2x^i = -2x^2/(x-1)^3 + x/(x-1)^2 = (x+x^2)/(1-x)^3$.

2.5–2. $\sum_{i\geq 1} i^3x^i = (x + 4x^2 + x^3)/(1-x)^4$, for $|x| < 1$.

2.5–3. This problem has an error and it should probably be moved to section 3.2 after it is corrected. It should read as follows. Express $\sum_{0\leq k<n} k^m z^k$ in terms of sums for the form $\sum_{0\leq k<n} k^j z^{k+1}$, where $j < m$. Use summation with $a_k = z^k/(z-1)$ by parts to obtain $\sum_{0\leq k<n} k^m z^k = n^m z^n/(z-1) - \sum_{0\leq k<n}[(k+1)^m - k^m]z^{k+1}/(z-1) = n^m z^n/(z-1) - \sum_{0\leq k<n}\sum_{0\leq i<m}\binom{m}{i}k^i z^{k+1}/(z-1)$.

2.5–4. $\sum_{i\geq 1} i^2x^i = (x+x^2)/(1-x)^3$ and $\sum_{i\geq 1} ix^i = x/(x-1)^2$, so $\sum_{i\geq 1}(i^2 - i)x^i = (2x^2)/(1-x)^3$. (This answer is not right.)

2.5.2–1. As the text explains, Step 2 of Heap is done as often as any other step, and Step 2 is done at most $\lg N$ times each time Heap is called. Heap is called less than $2N$ times. Thus the time is no more than $2tN \lg N$ (where t is the time for all the steps), which $O(N \lg N)$.

2.5.2–2. $\lfloor \lg(N/i) \rfloor = \lfloor \lg N - \lg i \rfloor$. For $N > 1$ and N equal to a power of 2, $\lg N$ is an integer. In this case, $\lfloor \lg N - \lg i \rfloor = \lg N - \lceil \lg i \rceil$. Also $\lceil \lg i \rceil = 1 + \lfloor \lg i \rfloor$, except when $\lg i$ is an integer (which happens $\lfloor \lg N \rfloor + 1$ times). From eq.(2–83) we get $\sum_{1\leq i\leq N} \lfloor \lg i \rfloor = N\lfloor \lg N \rfloor - 2^{\lfloor \lg N \rfloor + 1} + \lfloor \lg N \rfloor + 2$. From the above discussion we get $\sum_{1\leq i\leq N} \lceil \lg i \rceil = N\lfloor \lg N \rfloor + N - 2^{\lfloor \lg N \rfloor + 1} + 1$. Thus, the expression reduces as follows: $\lfloor N/2 \rfloor + \sum_{1\leq i\leq N/2} \lfloor \lg(N/i) \rfloor = N/2 + (N \lg N)/2 - \sum_{1\leq i\leq N/2} \lceil \lg i \rceil = N/2 + (N \lg N)/2 - (N/2)\lg(N/2) - N/2 + 2^{\lg(N/2)+1} - 1 = (N \lg N)/2 - (N \lg N)/2 + N/2 + N - 1 = 3N/2 - 1$. For the case where N is not a power of 2, $\lfloor \lg N - \lg i \rfloor = \lg N - \lceil \lg i \rceil + \epsilon(i)$, where $-1 < \epsilon(i) < 1$. Thus the answer changes from the above calculation by no more than $\pm N/2$ (plus a small constant when N is odd).

2.5.2–3. See the reference.

2.6.1–1. Since $1/(i^2 - 1) = 1/2[(i-1)] - 1/[2(i+1)]$, the sum simplifies to $(1/2) + (1/4) - 1/(2n) - 1/[2(n+1)] = 3/4 - 1/(2n) - 1/[2(n+1)]$.

2.6.1–2. Since $1/[i(i+1)(i+2)] = 1/(2i) - 1/(i+1) + 1/[2(i+2)]$, the sum simplies to

$1/2+1/4-1/2-1/(n+1)+1/[2(n+1)]+1/[2(n+2)] = 1/4-1/[2(n+1)]+1/[2(n+2)]$.

2.6.1–3. Nothing cancels.

2.7–1. $(x+x^2)/(1-x)^3$.

3–1. $\prod_{1\le i\le n} a^i = \exp\left(\sum_{1\le i\le n} i\ln a\right) = \exp\{[n(n+1)/2]\ln a\} = a^{n(n+1)/2}$.

3.1–1. 1.3×10^{12}.

3.1–2. $\ln(n!)$.

3.1.1–1. 1

3.1.1–2. *abc, abd, abe, acb, acd, ace, adb, adc, ade, aeb, aec, aed, bac, bad, bae, bca, bcd, bce, bda, bdc, bde, bea, bec, bed, cab, cad, cae, cba, cbd, cbe, cda, cdb, cde, cea, ceb, ced, dab, dac, dae, dba, dbc, dbe, dca, dcb, dce, dea, deb, dec, eab, eac, ead, eba, ebc, ebd, eca, ecb, ecd, eda, edb, edc.*

3.1.3–1. $q_i - q_{i+1} = (k!/N!)[(N-i+1)!/(k-i+1)! - (N-i)!/(k-i)!] = [k!(N-i)!]/[N!(k-i)!][(N-i+1)/(k-i+1)-1] = [k!(N-i)!]/[N!(k-i)!][(N-k)/(k-i+1)] = [k!(N-i)!]/[N!(k-i+1)!](N-k)$.

3.1.3–2. We need $q_2 > 1/2$, which gives $k/N > 1/2$.

3.1.4–1. Base case: Since $0! = 1$ and $\Gamma(1) = 1$, $\Gamma(n+1) = n!$ for $n = 0$. General case: Assume that $\Gamma(k+1) = k!$ for $k < n$. Then $\Gamma(k+2) = (k+1)\Gamma(k+1)$ and $(k+1)! = (k+1)k!$ for $k < n$. Using $k = n-1$ gives $\Gamma(n+1) = n\Gamma(n) = n(n-1)! = n!$, so the result is also true for $k = n$.

3.1.4–2. See reference.

3.1.4–3. $(1-t/m)^m = e^{m\ln(1-t/m)}$. $-t/m \ge \ln(1-t/m) \ge -t/m(1-t/m)^{-1}$ (eq. 4.17). Now for $0 \le t$, $-t/m(1-t/m)^{-1} \ge -t/m+(t/m)^2$, so $\ln(1-t/m) \le -t/m+(t/m)^2$. This gives $e^{-t} \ge (1-t/m)^m \ge e^{-t}e^{t^2/m}$. Now $e^{t^2/m} \ge 1+t^2/m$ (eq. 4.14), so $0 \le e^{-t}-(1-t/m)^m \le t^2e^{-t}/m$. $\int_0^\infty e^{-t}t^{x-1}\,dt - \Gamma_m(x) = \int_0^\infty e^{-t}t^{x-}\,dt - \int_0^m \left(1-\frac{t}{m}\right)^m t^{x-1}\,dt = \int_m^\infty e^{-t}t^{x-1}\,dt + \int_0^m \left(e^t - \left(1-\frac{t}{m}\right)^m\right) t^{x-1}\,dt$. Now for large enough m, $t^{x-1}e^{-t/2} < 1$, so the first integral is less than $\int_m^\infty e^{-t/2}\,dt = e^{-m/2}$ for large m, which goes to zero as m goes to infinity. The second integral is between zero and $1/m\int_0^\infty t^2e^{-t}\,dt = 2/m$, so it also goes to zero.

3.1.4–4. $\Gamma(p+\frac{1}{2}) = (p-\frac{1}{2})\Gamma(p-\frac{1}{2}) = (p-\frac{1}{2})(p-\frac{3}{2})\Gamma(p-\frac{3}{2}) = \cdots = (p-\frac{1}{2})(p-\frac{3}{2})\cdots\frac{3}{2}\frac{1}{2}\Gamma(\frac{1}{2}) = (2p-1)(2p-3)\cdots 1\Gamma(\frac{1}{2})/2^p = (2p)!\Gamma(\frac{1}{2})/[2^p(2p)(2p-2)\cdots 2] = (2p)!\Gamma(\frac{1}{2})/[2^p2^pp!] = (2p)!\Gamma(\frac{1}{2})/[2^{2p}p!]$.

3.2–1. $\binom{\frac{1}{2}}{n+1} = \frac{1}{2}(-\frac{1}{2})(-\frac{3}{2})\cdots(-n+\frac{1}{2})/(n+1)! = 1(-1)(-3)\cdots(-2n+1)/[2^{n+1}(n+1)!] = (-1)^n1\cdot 3\cdots(2n-1)/[2^{n+1}(n+1)!] = (-1)^n(2n)!/[2^{n+1}(n+1)!2\cdot 4\cdots 2n] = (-1)^n(2n)!/[2^{2n+1}(n+1)!n!] = (-1)^n\binom{2n}{n}2^{-2n-1}/(n+1)$.

3.2–2. $\binom{n}{k} = n(n-1)\cdots(n-k+1)/[k(k-1)\cdots 1] = n^{\underline{k}}/k!$.

3.2.1–1. $\sum_i i\binom{n}{i} = \sum_i n\binom{n-1}{i-1} = n2^{n-1}$.

3.2.1–2. $\sum_{0\le i\le n/2} i\binom{n}{i} = \sum_{0\le i\le n/2} n\binom{n-1}{i-1} = \sum_{0\le i\le n/2-1} n\binom{n-1}{i}$. Each term in the range from 0 to $n/2-1$ has a matching term of the same value in the range from n to $n/2+1$, so the sum is equal to one half of the sum over the full range, except that for odd n there is a correction for the terms near the middle. For odd n the answer is $n2^{n-2} + n\binom{n-1}{\lfloor n/2\rfloor}/2$, while for even n the answer is $n2^{n-2}$

3.2.1–3. $\sum_i \binom{n}{i}\binom{i}{k}x^i = \sum_i \binom{n}{k}\binom{n-k}{i-k}x^i = \sum_i \binom{n}{k}\binom{n-k}{i}x^{i+k} = \binom{n}{k}x^k(1+x)^{n-k}$.

3.2.1–4. $\sum_i \binom{n+i}{n}x^i = \sum_i \binom{n+i}{i}x^i = (1-x)^{-n-1}$ so long as all the sums converge ($|x| < 1$).

3.2.1–5. $\sum_i \binom{n-i}{n-m}\binom{n}{i} = \sum_i (n-i)!n!/[(n-m)!(m-i)!i!(n-i)!] = n!/[(n-m)!m!]\sum_i m!/[(m-i)!i!] = \binom{n}{m}2^m$.

3.2.1–6. Left side: $\binom{-n}{i} = (-n)(-n-1)\cdots(-n-i+1)/(1\cdot 2\cdots i) = (-1)^i(n+i-1)(n+i-2)\cdots n/(1\cdot 2\cdots i)$. Right side: $\binom{-i-1}{n-1} = (-i-1)(-i-2)\cdots(-i-n+1)/(1\cdot 2\cdots(n-1)) = (-1)^{n-1}(n+i-1)(n+i-2)\cdots(i+1)/(1\cdot 2\cdots(n-1))$. For most values of n and i, one of the two products has terms that cancel. Suppose $i \geq n$. Then the left side product becomes $(-1)^i(n+i-1)(n+i-2)\cdots i+1/(1\cdot 2\cdots(n-1))$, so the two products differ by a factor of $(-1)^{n+i+1}$. If $i = n-1$ the two products are the same. If $i < n-1$, terms in the right side product cancel, and again the two products are equal except for a factor of $(-1)^{n+i+1}$.

3.2.1–7. Define $S_m = \sum_{0\leq i<n}\binom{i}{m}x^i$ Use summation by parts to obtain $S_m = \binom{n}{m}x^n/(x-1) - \sum_{0\leq i<n}\binom{i}{m-1}\frac{x^{i+1}}{x-1}$. Apply this result repeatedly to obtain $S_m = [x^n/(x-1)\sum_{0\leq i\leq m}\binom{n}{m-i}[x/(x-1)]^i + [-x/(x-1)]^m S_0$. The sum S_0 can be worked out directly to obtain $S_m = [x^n/(x-1)\sum_{0\leq i\leq m}\binom{n}{m-i}[x/(x-1)]^i + [-x/(x-1)]^m[(x^n-1)/(x-1)]$. This is a simplification provided $m < n$. For $m > n$ the sum is zero (because the binomial is always zero in this case).

3.2.2–1. See reference.

3.2.4–1. $k(N+1)(N-k)/[(N-k+1)^2(N-k+2)] = \{kN(N-k)/[(N-k+1)^2(N-k+2)]\}(1+1/N) = \{kN(N-k)/[(N-k+1)^2(N-k+2)]\}[1+O(1/N)] = \{kN(N-k+2)/[(N-k+1)^2(N-k+2)]\}[1+O(1/N)][1-2/(N-k)] = [kN/(N-k+1)^2][1+O(1/N)][1+O(1/[N-k])] = [kN/(N-k+1)^2][1+O(1/N)+O(1/[N-k])+O(1/[N(N-k)])] = [kN/(N-k+1)^2][1+O(1/[N-k])] = [kN/(N-k)^2][(N-k)/(N-k+2)]^2[1+O(1/[N-k])] = [kN/(N-k)^2][1-2/(N-k+2)]^2[1+O(1/[N-k])] = [kN/(N-k)^2][1+O(1/[N-k])]^3 = [kN/(N-k)^2][1+3O(1/[N-k])+3O(1/[N-k]^2)+3O(1/[N-k]^3) = [kN/(N-k)^2][1+O(1/[N-k])]$. (See Chapter 4 for a detailed explanation of why each step is true.

3.2.4–2. The terms with $i > k+1$ are zero because $i!$ is infinite for integer $i < 0$. The terms with $i < 1$ are not zero.

3.3–1. $\sum_{i\geq 1}\sum_{0\leq j\leq k-1}(j/N)^{i-1}[1-(j/N)] = \lim_{n\to\infty}\sum_{1\leq i\leq n}\sum_{0\leq j\leq k-1}(j/N)^{i-1}[1-(j/N)] = \lim_{n\to\infty}\sum_{0\leq j\leq k-1}\sum_{1\leq i\leq n}(j/N)^{i-1}[1-(j/N)] = \lim_{n\to\infty}\sum_{0\leq j\leq k-1}[1-(j/N)][1-(j/N)^n]/(1-j/N) = \sum_{0\leq j\leq k-1}[1-(j/N)][1-\lim_{n\to\infty}(j/N)^n]/(1-j/N)$. Now $\lim_{n\to\infty}(j/N)^n$ is zero for $j/N < 1$, one for $j/N = 1$, and ∞ for $j/N > 1$, so the double sum reduces to $\sum_{0\leq j\leq k-1}(1-j/N)/(1-j/N) = k$ for $N > k-1$, to $k-1$ for $N = k-1$, and to $-\infty$ for $N < k-1$.

3.3–2. Interchange the order of summation to obtain $\sum_{n\geq 0}z^n\sum_{0\leq i\leq n-1}C_iC_{n-i-1} = \sum_{i\geq 0}C_i\sum_{n\geq i+1}z^nC_{n-i-1} = z\sum_{i\geq 0}z^iC_i\sum_{n\geq i+1}z^{n-i-1}C_{n-i-1}$. Let $j = n-i-1$ ($n = j+i+1$) to obtain $z\sum_{i\geq 0}z^iC_i\sum_{j\geq 0}z^jC_j$.

3.3–3. $\sum_{0\leq i<n}y^{-ki}\sum_{0\leq j<n}y^{ij}x_j = \sum_{0\leq i<n}\sum_{0\leq j<n}y^{i(j-k)}x_j = \sum_{0\leq j<n}x_j\sum_{0\leq i<n}y^{(j-k)i}$. If $y^{j-k} \neq 1$, then the sum over i is $(y^{(j-k)n} -$

$1)/(y^{(j-k)}-1)$, and the final result is $\sum_{0\le j<n} x_j(y^{(j-k)n}-1)/(y^{(j-k)}-1)$. If $y^{j-k}=1$, then the sum over i is n and the final result is $n\sum_{0\le j<n} x_j$.

3.3–4. $\sum_m \sum_{t_1} \sum_n \sum_{t_2} p_{i-1,t_1}(m)p_{i-1,t_2}(n) = \left(\sum_m \sum_{t_1} p_{i-1,t_1}(m)\right)$ $\left(\sum_n \sum_{t_2} p_{i-1,t_2}(n)\right)$.

3.3–5. The average time is $(1/n^2)\sum_{1\le i\le n}\sum_{1\le j\le n}[t_0(1-\delta_{ij})+t_1|i-j|] =$ $(1/n^2)\sum_{1\le i\le n}\sum_{1\le j\le n} t_0(1-\delta_{ij}) + (1/n^2)\sum_{1\le i\le n}\sum_{1\le j\le n} t_1|i-j| = [n(n-1)/n^2]t_0 + (1/n^2)t_1\sum_{1\le i\le n}\left(\sum_{1\le j<i}(i-j)+\sum_{i<j\le n}(j-i)\right) = (1-1/n)t_0 +$ $(1/n^2)t_1\sum_{1\le i\le n}[i(i-1)-i(i-1)/2+n(n+1)/2-i(i+1)/2-(n-i)i = (1-1/n)t_0+(1/n^2)t_1\sum_{1\le i\le n}[i^2-i+n^2/2+n/2-ni] = (1-1/n)t_0+(1/n^2)t_1[n^2(n+1)/2+(2n^3+3n^2+n)/6-n(n+1)^2/2] = (1-1/n)t_0+t_1(n-1/n)/3$.

3.3–6. $\sum_{m_1,m_2,\ldots,m_k} \prod_{1\le i\le k}(-1)^{m_i}\binom{n}{m_i}x^{m_i} =$ $\sum_{m_1,m_2,\ldots,m_{k-1}} \prod_{1\le i\le k-1}(-1)^{m_i}\binom{n}{m_i}x^{m_i}\left(\sum_{m_k}\binom{n}{m_k}(-x)^{m_k}\right) =$ $\sum_{m_1,m_2,\ldots,m_{k-1}} \prod_{1\le i\le k-1}(-1)^{m_i}\binom{n}{m_i}x^{m_i}(1-x)^n$. Repeated application gives $(1-x)^{kn}$.

3.3.1–1. The average of the sum of the squares is $\sum_{i,j} j^2 q_{ij} = \sum_i Np_i/k\sum_{1\le j\le i} j^2 =$ $\sum_i Np_i(2i^3+3i^2+i)/(6k) = N/(6k)\sum_i(2i^3+3i^2+i)\binom{k}{i}(N-1)^{k-i}N^{-k} =$ $(N/k)\sum_i(2\binom{i}{3}+3\binom{i}{2}+i)\binom{k}{i}(N-1)^{k-i}N^{-k} = (N/k)[k/N+(3/2)k(k-1)/N^2+(2/6)k(k-1)(k-2)/N^3] = 1+(3/2)(k-1)/N+(1/3)(k-1)(k-2)/N^2$. Subtracting the square of the average gives the variance: $1+(3/2)(k-1)/N+(1/3)(k-1)(k-2)/N^2-[1+(k-1)/2N]^2 = (k-1)/(2N)+(k^2-6k+5)/(12N^2)$.

3.3.1–2. Although there is a simple relation between the average time to build a hash tables and the average time to look items up, there is apparently no such relation between the variances.

3.4–1. $\sum_j a_j = \sum_{j\text{ even}} a_j + \sum_{j\text{ odd}} a_j$ and $\sum_j(-1)^j a_j = \sum_{j\text{ even}} a_j - \sum_{j\text{ odd}} a_j$, so $\frac{1}{2}\sum_j a_j - \frac{1}{2}\sum_j(-1)^j a_j = \sum_{j\text{ odd}} a_j$.

3.4–2. $\sum_j\binom{n}{2j}x^j = \sum_j\binom{n}{2j}(x^{1/2})^{2j} = \sum_{j\text{ even}}\binom{n}{j}(x^{1/2})^j = \frac{1}{2}[(1+x^{1/2})^n+(1-x^{1/2})^n]$.

3.4–3. $\sum_j\binom{n}{2j}x^{2j} = \sum_{j\text{ even}}\binom{n}{j}(x)^j = \frac{1}{2}[(1+x)^n+(1-x)^n]$.

3.4–4. $\sum_{0\le i<n/2}\binom{n}{2i+1}x^{2i+1} = \sum_{i\text{ odd}}\binom{n}{i}x^i = \frac{1}{2}[(1+x)^n-(1-x)^n]$.

3.4–5. $\sum_j\binom{n}{4j} = \frac{1}{4}[2^n+(1+i)^n+\delta_{n0}+(1-i)^n]$. Continuing with the techniques of the next section gives $\frac{1}{4}[2^n+(2\sqrt{2})^n\cos(\pi n/4)+\delta_{n0}+(-2\sqrt{2})^n\cos(3\pi n/4)$.

3.4–6. See Exercise 3.4.1–2.

3.4–7. $\sum_{j\bmod 4=1} a_j = \frac{1}{4}\sum_j a_j + \frac{1}{4}\sum_j(i)^{j-1}a_j + \frac{1}{4}\sum_j(-1)^{j-1}a_j + \frac{1}{4}\sum_j(-i)^{j-1}a_j$.

3.4.1–1. $(x+iy)(x-iy) = x^2-i^2y^2 = x^2+y^2 = r^2$.

3.4.1–2. $\sum_{0\le j<n}\omega^{-mj}\sum_k\omega^{jk}x_k = \sum_k x_k\sum_{0\le j<n}\omega^{(k-m)j}$. Now $\sum_{0\le j<n}\omega^{(k-m)j} = (1-\omega^{(k-m)n})/(1-\omega^{k-m}) = 0$ provided $\omega^{k-m}\ne 1$, i.e., if $k\ne m$. If $\omega^{k-m}=1$, then the sum is n. Therefore, $(1/n)\sum_{0\le j<n}\omega^{-mj}\sum_k\omega^{jk}x_k = \sum_k\delta_{km}x_m = x_m$.

3.4.2–1. See Ex 2. The answer is $(1/5)\{2^n+2[2\cos(\pi/5)]^n\cos(\pi n/5)+2[2\cos(2\pi/5)]^n\cos(2\pi n/5)\}$.

3.4.2–2. See Ex. 3 and make the simplification $x=1$.

3.4.2–3. $\sum_j\binom{n}{mj+k}x^j = (1/m)x^{-k/m}\sum_j\binom{n}{mj+k}(x^{1/m})^{mj+k} =$ $(1/m)x^{-k/m}\sum_{0\le l<m}\sum_j\omega^{l(j-k)}\binom{n}{j}(x^{1/m})^j =$

$(1/m)x^{-k/m}\sum_{0\le l<m}\omega^{-kl}\sum_j \omega^{jl}\binom{n}{j}(x^{1/m})^j = (1/m)x^{-k/m}\sum_{0\le l<m}\omega^{-kl}(1+\omega^l x^{1/m})^n$. To remove the complex numbers, use $\omega = \cos(2\pi n/m)+i\sin(2\pi n/m)$ and polar representation of complex numbers. $(1+\omega^l x^{1/m}) = \sqrt{[1+x^{1/m}\cos(2\pi l/m)]^2+[x^{1/m}\sin(2\pi l/m)]^2}$ times a complex unit vector, which simplifies to $\sqrt{2+2x^{1/m}\cos(2\pi l/m)}$ times the unit vector. The unit vector has angle $n\arcsin[\sin(2\pi l/m)/\sqrt{2+2x^{1/m}\cos(2\pi l/m)}]$. In the sum over l, the terms with $1 \le l < m/2$ have the same real parts an opposite imaginary parts as the terms with $m-1 \ge l > m/2$. Taking just the real part of the terms gives $\sum_j \binom{n}{mj+k}x^j = (1/m)\sum_{0\le l<m}[2+2x^{1/m}\cos(2\pi l/m)]^{n/2}\cos\{2\pi kl/m + n\arcsin[\sin(2\pi l/m)/\sqrt{2+2x^{1/m}\cos(2\pi l/m)}]\}$. When $x = 1$ the sum simplifies more, as suggested by exercise 2.

3.5–1. For positive integer r and s, $(1+x)^{r+s} = \sum_n \binom{r+s}{n}x^n$, so the coefficient of x^n in $(1+x)^{r+s}$ is $\binom{r+s}{n}$. $(1+x)^r(1+x)^s = \left(\sum_i \binom{r}{i}x^i\right)\left(\sum_n \binom{s}{n}x^n\right) = \sum_i\sum_n \binom{r}{i}\binom{s}{n}x^{i+n} = \sum_i\sum_n \binom{r}{i}\binom{s}{n-i}x^n = \sum_n\left(\sum_i \binom{r}{i}\binom{s}{n-i}\right)x^n$, so the coefficient of x^n is $\sum_i \binom{r}{i}\binom{s}{n-i}$. Since the two expressions are equal for all x, they must be equal term by term. Therefore, $\binom{r+s}{n} = \sum_i \binom{r}{i}\binom{s}{n-i}$ for positive integer r and s. As explained in the text, this result must also hold for all r and s because both sides are polynomials in r and in s, and both sides are equal at a large enough number of points.

3.5–2. The coefficient of x^k in $(1+x)^{2m}$ is $\binom{2m}{k}$. $(1+2x+x^2)^m = \sum_i \binom{m}{i}(2x+x^2)^{m-i} = \sum_i \binom{m}{i}\sum_j \binom{m-i}{j}2^j x^{2m-2i-j} = \sum_i \binom{m}{i}\sum_j \binom{m-i}{2m-2i-j}2^{2m-2i-j}x^j$. Equating powers of x gives $\binom{2m}{j} = \sum_i \binom{m}{i}\binom{m-i}{2m-2i-j}2^{2m-2i-j}$. Applying eq. (53) gives $\binom{2m}{j} = \sum_i \binom{m}{i}\binom{m-i}{i+j-m}2^{2m-2i-j}$.

3.5–3. Both sides are polynomials in s of degree $n+r$, and, by the derivation, the two sides are equal for all positive integer s. Therefore the two sides must be equal for all s.

3.5–4. The substitution used in the derivation of eq. (167) works when s and $r+s$ are nonnegative integers, so eq. (167) is true in this case due to the derivation. Now both sides of eq. (167) are polynomials in s of degree $r+n$. The previous argument shows that the two sides are equal at an infinite number of points. Since both sides are polynomials in s, they must be equal for all s.

3.5–5. With $n \ge s \ge 0$, the lower summation limit in eq. (168) is greater than or equal to zero. It can, however, be replaced with zero, because $\binom{s+i}{n}$ is zero for $n \ge s$, $i \ge 0$. This proves the result so long as $m \le r$. When $m > r$ (and $n \ge s$), the sum has no terms, so it is zero (this also works with the other summation convention because the resulting negative terms produced by the other convention are zero) and the right side is also zero.

3.5–6. Apply eq. (69) to the second binomial in eq. (167) to obtain $\sum_i \binom{r}{i}\binom{-s-1+n+i}{n+i}(-1)^{n+i} = \binom{r+s}{r+n}$. Use eq. (53) to obtain $\sum_i \binom{r}{i}\binom{-s-1+n+i}{-s-1}(-1)^{n+i} = \binom{r+s}{r+n}$. Let $n' = -s-1$, $s' = -s-1+n$ ($s = -n'-1$, $n = s'-n'$) to obtain $\sum_i \binom{r}{i}\binom{s+i}{n}(-1)^i = (-1)^{s-n}\binom{r-n-1}{r+s-n}$. Apply

eq. (69) followed by eq. (53)to the right side of this to obtain $\sum_i \binom{r}{i}\binom{s+i}{n}(-1)^i = (-1)^r\binom{s}{r+s-n}(-1)^r\binom{s}{n-r}$. These steps all work if n, r, and s are nonnegative integers. Since both sides are polynomials in s that are equal for an infinite number of points, the equation is true for all s.

3.5–7. Apply eq.(72) to the first binomial in eq. (167) to obtain $\sum_i \binom{-i-1}{-r-1}\binom{s}{n+i}(-1)^{-r+i+1} = \binom{r+s}{r+n}$. Let $m = -r-1$, $r'-i' = -i-1$, and $i'-t = n+i$ ($i = i'-r'-1$, $r = -m-1$, and $n = r'-t+1$) to obtain $\sum_i \binom{r-i}{m}\binom{s}{i-t}(-1)^i = (-1)^{m-r+1}\binom{-m-1+s}{-m+r-t}$. Apply eq. (53) to the right side gives $(-1)^{m-r+1}\binom{-m-1+s}{-1+s-r+t}$. Applying eq.(71) gives $(-1)^t\binom{r-s-t}{r-t-m}$.

3.5–8. By eq. (169), we have that the sum is equal to $(-1)^m\binom{0}{n-m}$ which is equal to $(-1)^m\delta_{mn}$.

3.5–9. $\sum_{m_1+m_2+\cdots+m_k=p} \prod_{1\le i\le k}(-1)^{m_i}\binom{n}{m_i}x^{m_i} =$
$\sum_{m_1,m_2,\ldots,m_{k-1}} \prod_{1\le i\le k-1}$
$(-1)^{m_i}\binom{n}{m_i}x^{m_i}\binom{n}{p-m_1-m_2-\cdots-m_{k-1}}(-x)^{p-m_1-m_2-\cdots-m_{k-1}} =$
$\sum_{m_1,m_2,\ldots,m_{k-2}} \prod_{1\le i\le k-2}(-1)^{m_i}\binom{n}{m_i}x^{m_i}$
$\left(\sum_{m_{k-1}}\binom{n}{p-m_1-m_2-\cdots-m_{k-1}}\binom{n}{m_{k-1}}(-x)^{p-m_1-m_2-\cdots-m_{k-1}}(-x)^{m_{k-1}}\right) =$
$\sum_{m_1,m_2,\ldots,m_{k-2}} \prod_{1\le i\le k-2}(-1)^{m_i}\binom{n}{m_i}x^{m_i}$
$\left(\sum_{m_{k-1}}\binom{n}{p-m_1-m_2-\cdots-m_{k-1}}\binom{n}{m_{k-1}}(-x)^{p-m_1-m_2-\cdots-m_{k-2}}\right) =$
$\sum_{m_1,m_2,\ldots,m_{k-2}} \prod_{1\le i\le k-2}$
$(-1)^{m_i}\binom{n}{m_i}x^{m_i}\binom{2n}{p-m_1-m_2-\cdots-m_{k-2}}(-x)^{p-m_1-m_2-\cdots-m_{k-2}}$. Repeated application gives $\binom{kn}{p}$.

3.6–1. $\sum_{n=n_1+n_2+\cdots+n_i}\binom{n_1+n_2+\cdots+n_i}{n_1,n_2,\ldots,n_i} = \sum_{n=n_1+n_2+\cdots+n_i}\binom{n_1+n_2+\cdots+n_i}{n_1,n_2,\ldots,n_i}1^{n_1}1^{n_2}\cdots 1^{n_i} = (1+1+\cdots+1)^n = i^n$, where $1+1+\cdots+1$ is short for i ones. The sum is done by the multinomial theorem.

3.7–1. $i^k = \sum_j \left\{{k \atop j}\right\}\binom{i}{j}j!$, so $\sum_i(-1)^i\binom{m}{i}i^k = \sum_i(-1)^i\binom{m}{i}\sum_j\left\{{k \atop j}\right\}\binom{i}{j}j! =$
$\sum_j\left\{{k \atop j}\right\}j!\sum_i(-1)^i\binom{m}{i}\binom{i}{j} = \sum_j\left\{{k \atop j}\right\}j!(-1)^m\delta_{jm} = \left\{{k \atop m}\right\}m!(-1)^m$. Now $\left\{{k \atop m}\right\} = 0$ for $k < m$ and $\left\{{k \atop m}\right\} = 1$ for $k = m$, so the result follows.

3.7–2. Show that $\sum_i \binom{n}{i}(-1)^i i^m = (-1)^n n!\left\{{m \atop n}\right\}$. This can be done with the same technique as exercise 3.7–1.

3.7–3. Consider a permutation of n objects which has k cycles. The last object either is in a unit cycle or it is in a larger cycles. The number of permutations with the last item in a unit cycle is $c_{n-1,k-1}$, because in this case removing the last item gives a permutation with $n-1$ objects and $k-1$ cycle and each permutation with $n-1$ objects and $k-1$ cycles can be obtained in just one way. The number of permutations with the last item in a cycle of more than one element is $(n-1)c_{n-1,k}$, because each such permutation can be obtained in one and only one way by starting with a permutation of $n-1$ item and k cycles and adding the n^{th} item. There are $c_{n-1,k}$ permutations to start with and for each one there are $n-1$ places to insert the n^{th} item. Thus $c_{nk} = (n-1)c_{n-1,k} + c_{n-1,k-1}$.

3.7–4. The recurrence of the previous exercise can be used to compute c_{nk} for all positive integer n and k provided we are given c_{0k} and c_{n0}. With zero objects, the

permutation must have zero cycles, so $c_{0k} = \delta_{0k}$. The only way to have zero cycles is to have zero objects, so $c_{n0} = \delta_{n0}$. Since c_{nk} obeys the same recurrence as $\left[{n \atop k}\right]$, since it obeys the same boundary conditions, and since the values can be computed (using the recurrence) from the boundary conditions, the values must be the same. If you want a more formal argument, use proof by induction.

3.7.1–1. A good order is p_{1i} for $1 \le i \le 5$, p_{2i} for $1 \le i \le 5$, etc.

n	p_{n1}	p_{n2}	p_{n3}	p_{n4}	p_{n5}
1	1	0	0	0	0
2	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0
3	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{1}{2}$	0	0
4	$\frac{1}{4}$	$\frac{11}{24}$	$\frac{1}{4}$	$\frac{1}{24}$	0
5	$\frac{1}{5}$	$\frac{5}{12}$	$\frac{7}{24}$	$\frac{1}{12}$	$\frac{1}{120}$

3.7.1–2. Base case ($n = 0$). For $n = 0$, Step 4 is done zero times. General case ($n > 0$). Eq. (194) was proved in the text. Assume $p_{ni} = \left[{n \atop i+1}\right]/n!$ for $n < n_*$. Then $p_{n_*i} = (1/n_*)\left[{n_*-1 \atop i}\right]/(n_*-1)! + ((n_*-1)/n_*)\left[{n_*-1 \atop i+1}\right]/(n_*-1)! = \{\left[{n_*-1 \atop i}\right] + (n_*-1)\left[{n_*-1 \atop i+1}\right]\}/n_*! = \left[{n_* \atop i+1}\right]/n_*!$, for the result is also true for $n = n_*$

3.7.1–3. $\left[{n+1 \atop 2}\right] = \sum_{1 \le i_1 < i_2 < \cdots < i_{n-1} < n+1} i_1 i_2 \cdots i_{n-1}$. This is $n-1$ increasing numbers in the range 1 to n, so each number must be one more than the one below it, except that one number is left out. Thus the sum has n terms, and its value is $\sum_{1 \le i \le n} n!/i$.

3.8–1. $\sum_i \binom{r}{i}\binom{s}{n+i} = \sum_i r!s!/[i!(r-i)!(n+i)!(s-n-i)!] = r!s!\sum_i (-r)^{\overline{i}}(-s+n)^{\overline{i}}/[i!r!n!(s-n)!(n+1)^{\overline{i}}] = \binom{s}{n}{}_2F_1[-r, -s+n; n+1; 1] = \binom{s}{n}[n!(r+s)!/(n+r)!s!] = (r+s)!/[(n+r)!(s-n)!] = \binom{r+s}{r+n}$.

3.8–2. $\sum_{v \ge 0} \frac{w\binom{n}{w}\binom{m}{v}}{(n+m)\binom{n+m-1}{w+v-1}} = \sum_{v \ge 0} n!m!(w+v-1)!(n+m-w-v)!/[(w-1)!(n-w)!v!(m-v)!(n+m)!] = n!m!/[(w-1)!(n-w)!(n+m)!]\sum_{v \ge 0}(w+v-1)!(n+m-w-v)!/[v!(m-v)!] = n!m!/[(w-1)!(n-w)!(n+m)!]\sum_{v \ge 0}(w-1)!w^{\overline{v}}(-m)^{\overline{v}}(n+m-w)!/[v!m!(-n-m+w)^{\overline{v}}] = n!(n+m-w)!/[(n-w)!(n+m)!]\sum_{v \ge 0} w^{\overline{v}}(-m)^{\overline{v}}/[v!(-n-m+w)^{\overline{v}}] = n!(n+m-w)!/[(n-w)!(n+m)!]{}_2F_1[w, -m; -n-m+w; 1] = n!(n+m-w)!/[(n-w)!(n+m)!]\Gamma(-n-m+w)\Gamma(-n)/[\Gamma(-n-m)\Gamma(-n+w)] = n!(n+m-w)!/[(n-w)!(n+m)!](-1)^w\Gamma(n+m+1)\Gamma(n-w+1)/[(-1)^w\Gamma(n+m-w+1)\Gamma(n+1)] = n!(n+m-w)!/[(n-w)!(n+m)!](n+m)!(n-w)!/[(n+m-w)!n!] = 1$.

3.8–3. $\sum_{v \ge 0} \frac{vw\binom{n}{w}\binom{m}{v}}{(n+m)\binom{n+m-1}{w+v-1}} = \sum_{v \ge 1} \frac{vw\binom{n}{w}\binom{m}{v}}{(n+m)\binom{n+m-1}{w+v-1}} = \sum_{v \ge 1} n!m!(w+v-1)!(n+m-w-v)!/[(w-1)!(n-w)!(v-1)!(m-v)!(n+m)!] = \sum_{v \ge 0} n!m!(w+v)!(n+m-w-v-1)!/[(w-1)!(n-w)!v!(m-v-1)!(n+m)!] = n!m!/[(w-1)!(n-w)!(n+m)!]\sum_{v \ge 0}(w+v)!(n+m-w-v-1)!/[v!(m-v-1)!] = n!m!/[(w-1)!(n-w)!(n+m)!]\sum_{v \ge 0} w!(w+1)^{\overline{v}}(-m+1)^{\overline{v}}(n+m-w-1)!/[v!(m-1)!(-n-m+w+1)^{\overline{v}}] = n!(n+m-w-1)!mw/[(n-w)!(n+m)!]\sum_{v \ge 0}(w+1)^{\overline{v}}(-m+1)^{\overline{v}}/[v!(-n-m+w+1)^{\overline{v}}] = n!mw(n+m-w-1)!/[(n-w)!(n+m)!]{}_2F_1[w+1, -m+1; -n-m+w+1; 1] = \{n!mw(n+m-w-1)!/[(n-w)!(n+m)!]\}\{\Gamma(-n-m+w+$

$1)\Gamma(-n-1)/[\Gamma(-n-m)\Gamma(-n+w)]\} = \{n!mw(n+m-w-1)!/[(n-w)!(n+m)!]\}\{(-1)^{(w+1)}\Gamma(n+m+1)\Gamma(n-w+1)/[(-1)^{(w+1)}\Gamma(n+m-w)\Gamma(n+2)]\} = \{n!mw(n+m-w-1)!/[(n-w)!(n+m)!]\}\{(n+m)!(n-w)!/[(n+m-w-1)!(n+1)!]\} = mw/(n+1)$. This all works only when $w \le n$, because otherwise the original sum has a division by zero.

3.8–4. $\sum_i i\binom{n}{i}^2 = \sum_i n!n!/[i!(i-1)!(n-i)!(n-i)!] = (n!)^2\sum_i[(-n)^{\bar{i}}]^2/[i!(-1)!0^{\bar{i}}(n!)^2] = 1/(-1)!{}_2F_1[-n,-n;0;1] = 1/[(-1)!]\Gamma(0)\Gamma(2n)/[\Gamma(n)]^2 = (2n-1)!/[(n-1)!]^2 = n\binom{2n-1}{n}$.

3.8–5. $\sum_{i\ge 0}\frac{(2n)!}{(i!)^2[(n-i)!]^2} = (2n)!\sum_{i\ge 0}1/[(i!)^2[(n-i)!]^2] = (2n)!\sum_{i\ge 0}[(-n)^{\bar{i}}]^2/[(i!)0!1^{\bar{i}}(n!)^2] = (2n)!/(n!)^2{}_2F_1[-n,-n;1;1] = (2n)!/(n!)^2\Gamma(1)\Gamma(2n+1)/[\Gamma(n+1)]^2 = (2n)!/(n!)^2(2n)!/(n!)^2 = \binom{2n}{n}^2$.

3.8–6. The answer is $\binom{2j}{2n}$, at least for integer j, n. This can be proved by induction. If you try the method of this section, you get that the sum is $[(2n+1)!j!/(2n)!]_3F_2[j+1,-n,-n+\frac{1}{2};\frac{3}{2},j-2n+1;1]$, which is almost (but not quite) formula III.16 from Slater. Does Slater have an error? Is this a new result for hypergeometric series? Is the answer true for non integer j, n? (Send your answers to the authors.)

3.9–1. The number of permutations with no cycles of length one is equal to the total number of permutations minus the number with some cycles of length one. The number of permutations of n objects is $n!$. The number of permutations with the first i items forming one-cycles (and the remaining $n-i$ items doing anything) is $(n-i)!$. To have at least i one-cycles, there are $\binom{n}{i}$ ways to select i items to be one-cycles. This, of course, overcounts the cases which have more than i cycles. Putting this all together with inclusion and exclusion gives the answer $n! + \sum_{1\le i\le n}(-1)^i\binom{n}{i}(n-i)! = n!\left(1+\sum_{1\le i\le n}(-1)^i/i!\right) = n!\sum_{2\le i\le n}(-1)^i/i!$.

3.9–2. If you put at most a_1 pigeons in the first hole, at most a_2 pigeons in the second hole, ..., and at most a_n pigeons in the n^{th} hole, then you have put in at most $a_1+a_2+\cdots+a_n$ pigeons. To put in one more pigeon, at least one hole must have an extra pigeon.

3.9–3. See Alan Tucker, *Applied Combinatorics*, John Wiley and Sons New York (1984), pp. 313–314.

3.9–4. This problem is like 3.6–1, except for the $n_i \ge 1$ condition. The principle of inclusion and exclusion can be used to express this sum in terms of the sum with $n_i \ge 0$ and the sums obtained with various $n_i = 0$. $\sum_{n_1\ge 1,n_2\ge 1,\ldots,n_i\ge 1}\binom{n_1+n_2+\cdots+n_i}{n_1,n_2,\ldots,n_i} = \sum_k(-1)^k\binom{i}{k}\sum_{n_1=0,n_2=0,\ldots,n_k=0}\binom{n_1+n_2+\cdots+n_i}{n_1,n_2,\ldots,n_i} = \sum_k(-1)^k\binom{i}{k}(i-k)^n = \sum_k(-1)^{i-k}\binom{i}{k}k^n = i!\left\{{n \atop i}\right\}$.
$n=n_1+n_2+\cdots+n_i$ $n=n_1+n_2+\cdots+n_i$

4.1–1. $e^x = 1+x+x^2e^c/2$ where $0\le c\le x$, so for $x\ge 0$, $1+x+x^2/2 \le e^x \le 1+x+x^2e^x/2$. The lower limit is the lower limit needed. From the upper limit we get $e^x(1-x^2/2) \le 1+x$. For $|x|<\sqrt{2}$, the term in parentheses is positive, so in this case $e^x \le (1+x)/(1-x^2/2)$. For $-\sqrt{2}<x<0$, $1+x+x^2e^x/2 \le e^x \le 1+x+x^2/2$, so the limits are just reversed.

4.1–2. Proceed as in the last problem, but carry the expansions out to n terms.

4.1–3. $\ln(1+x) = x - x^2/[2(1+c)^2]$ for some c in the range $0 \le c \le x$, so $\ln(1+x) \le x - x^2/[2(1+x)^2]$. $\ln(1+x) = x - x^2/2 + c^3/[3(1+c)^3]$ for for c in the range $0 \le c \le x$, so $\ln(1+x) \ge x - x^2/2$.

4.1–4. Use eq.(16). For $0 \le c \le |x|$, the smallest $|(-1)^n c^{n+1}/[(n+1)(1+c)^{n+1}]|$ can be is 0 and the largest it can be is $|(-1)^n x^{n+1}/[(n+1)(1+x)^{n+1}]|$.

4.1–5. $\sin x = x - \frac{1}{2}x^2 \sin c$ for some c in the range $0 \le c \le x$. For $0 \le x \le \pi/2$, the last term is between 0 and $-\frac{1}{2}x^2 \sin x$, so $x/(1+x^2/2) \le \sin x \le x$.

4.1.1–1. The tree is too large to give here. It should have 16 in its root node, a $\binom{6}{2}$ tree for the down tree (see Fig. 4.2) and a $\binom{6}{3}$ tree for the right tree.

4.1.1–2. The number of leaves in a tree is one if the tree just has a root and it is the sum of the number of leaves in the children otherwise. The children of the root of a $\binom{d}{h}$ tree are a $\binom{d-1}{h}$ and a $\binom{d-1}{h-1}$ tree. Thus, letting L_{dh} be the number of leaves in a $\binom{d}{h}$ tree, we have $L_{dh} = L_{d-1,h} + L_{d-1,h-1}$ when $d > h$. Also L_{dd} and L_{d0} are 1. Thus, the numbers L_{dh} are generated by the same set of rules as the binomial coefficients are, so they must be the same. (Actually, the rules are slightly different when $d = h$, but the answer is still the same.) This can be given more formally as a proof by induction.

4.1.1–3. $\binom{(h!N)^{1/h}+h-1}{h} = [(h!N)^{1/h} + h - 1][(h!N)^{1/h} + h - 2]\cdots[(h!N)^{1/h}]/h!$. Replacing each factor on the top with $(h!N)^{1/h}$ makes the top smaller (or the same when $h \le 1$).

4.1.1–4. Prove by induction. Now $d(\binom{i}{0}) = 0$ and $d(\binom{i}{i}) = 0$ because the tree has a single node which is a root. In particular $d(\binom{1}{0}) = d(\binom{1}{1}) = 0$. For $i > h > 0$, $d(\binom{i}{h}) = \max\{1 + d(\binom{i-1}{h}), 1 + d(\binom{i-1}{h-1})\}$. By the inductive hypothesis, this reduces to $d(\binom{i}{h}) = \max\{1+i-2, 1+i-2\} = i-1$ for $h < i-1$. For $h = i-1$, it reduces to $d(\binom{i}{i-1}) = \max\{1, 1+i-2\} = i-1$. Now consider $h(T)$; $h(\binom{i}{0}) = 0$ and $h(\binom{i}{i}) = 0$ because the tree has a single node which is a root. For $d > i$, $h(\binom{d}{i}) = \max\{h(\binom{d-1}{i}), 1+h(\binom{d-1}{i-1})\}$. By the inductive hypothesis, this reduces to $h(\binom{d}{i}) = \max\{i, 1+i-1\} = i$ for $i < d-1$. For $i = d-1$, it reduces to $h(\binom{d}{d-1}) = \max\{1, 1+i-1\} = i$.

4.1.1–5. To prove that the method is not optimal consider Figure 4.2 with the circled node 14 modified so that its right son is a circled node 15. The down child of this 15 is a leaf with a 15 (we cannot afford to do a test on the down child, because all the tests have been used up) and the right child is a leaf with a 16. Thus it is possible to decide among 16 values using at most 5 test and 2 failures. This leaves the interesting question of what is the best method for doing destructive testing. The binomial tree algorithm is optimum if there is a limit on the number of failing tests and a limit on the number of successful tests rather than a limit on the number of failing tests and a limit on the total number of tests.

4.2–1. $x/(x-1) = 1 + x^{-1} + x^{-2} + x^{-3} + x^{-3}/(x-1) = 1 + x^{-1} + x^{-2} + O(x^{-3})$ for $x > 1 + \epsilon$, where ϵ is any positive constant.

4.2–2. $1/(1+cx) < 1/[1+O(x)] < 1/(1-cx)$ for some $c > 0$ and all $x \ge 0$ with $|x|$ below some limit, ϵ. Therefore, for such x, $1 - cx/(1+cx) < 1/[1+O(x)] < 1 + cx/(1-cx)$. A similar calculation is needed for $x < 0$. For $|x|$ below the minimum of $1/(2c)$

and ϵ, we have $1 - 2cx < 1/[1 + O(x)] < 1 + 2cx$, so $1/[1 + O(x)] = 1 + O(x)$. The second part of the problem is false. Consider the function $1/1 + 2x$. As x approaches $-1/2$, there is no limit on the size of the function, whereas $1 + O(x)$ stays below some limit.

4.2–3. For $c > 0$, $C = c$, and $C' = c$, $Cf(x) = cf(x) = C'f(x)$ for all x with $C > 0$ and $C' > 0$.

4.2–4. $\log_b x = \ln x / \ln b$, so let $C = C' = 1/\ln b$. (It is essential that b is a constant and not a function of x.)

4.2–5. It is necessary to assume $f(x) \geq 0$. With that assumption, $|g(x)f(x)| = |g(x)|f(x) \leq cf(x) = O(f(x))$.

4.2–6. It is necessary to assume $f(x) \geq 0$. With that assumption, $g(x)f(x) \geq cf(x) = \Omega(f(x))$.

4.2–7. $\Theta(\Theta(f(x)))$ means all the functions in the range $c_1c_2f(x)$ to $c_3c_4f(x)$ while $\Theta(f(x))$ means all the functions in the range $c_5f(x)$ to $c_6f(x)$. These two ranges are the same since any positive c_5 can be expressed as the product of some c_1 and c_2, where c_1 and c_2 are positive. Likewise any positive pair c_1, c_2 correspond to some positive c_5. Similar remarks are true for c_3, c_4, and c_6.

4.2–8. From the first Θ we have $C_1f(x) \leq \Theta(f(x)) \leq C_1'f(x)$ and from the second one we have $C_2f(x) \leq \Theta(f(x)) \leq C_2'f(x)$, so $(C_1 + C_2)f(x) \leq \Theta(f(x)) + \Theta(f(x)) \leq (C_1' + C_2')f(x)$. Since $C_1 + C_2 > 0$ and $C_1' + C_2' > 0$, this implies $\Theta(f(x)) + \Theta(f(x)) = \Theta(f(x))$. If the plus signs are changed to minus, then you do not know the sign of $C_1 - C_2$ or of $C_1' - C_2'$, so you can not conclude that the result is $\Theta(f(x))$; it might be too small. Eq. 41 is a big O result, so we only need an upper limit. More precisely, $-(C_1 + C_2)f(x) \leq O(f(x)) \pm O(f(x)) \leq (C_1 + C_2)f(x)$, so $O(f(x)) \pm O(f(x)) = O(f(x))$.

4.2–9. $f(x) = \Omega(g(x))$ implies $f(x) \geq Cg(x)$ implies $|g(x)| \leq f(x)/C$, provided $g(x)$ is positive. (The result is not always true for zero or negative $g(x)$.

4.2.1–1. By eq. (13) $e^x = \sum_{0 \leq i \leq n} \frac{x^i}{i!} + e^c x^{n+1}/(n+1)!$ for some c in the range $0 \leq c \leq x$, so $e^x = \sum_{0 \leq i \leq n} x^i/i! + \Theta(x^{n+1})$ with $C' = e^r/(n+1)!$ and $C = 1/(n+1)!$ in the definition of Θ [eq. (23)].

4.2.1–2. By eq. (16) $\ln(1+x) = -\sum_{1 \leq i \leq n}(-1)^i x^i/i + (-1)^n/(n+1)[x/(1+c)]^{n+1}$ for some c in the range $0 \leq c \leq x$, so $\ln(1+x) = -\sum_{1 \leq i \leq n}(-1)^i x^i/i + (-1)^n \Theta(x)^{n+1}$ with $C' = 1/(n+1)$ and $C = 1/[(n+1)(1+c)^{n+1}]$.

4.2.1–3. (Notice that direct application of eq.(49) does not work, because r must be fixed.) Note: this solution is for a corrected version of the problem. $(1 + a/x)^x = \exp[x \ln(1+a/x)] = \exp[-a + a^2/(2x) - \Theta(1/x^2)] = e^{-a} \exp[a^2/(2x) - \Theta(1/x^2)] = e^{-a}[1 + a^2/(2x) + \Theta(1/x^2) - \Theta(1/x^2)]$, where the first Θ term comes from the square term in the power series expansion of the exponential function and the second Θ term comes from the Θ that had been in the exponential. This gives us $(1 - a/x)^x = e^{-a}[1 + a^2/(2x) + O(1/x^2)]$, but it does not give a big Θ result because we don't know the sign of the difference of the two Θ terms. The next exercise shows that the difference can be positive, negative, or zero.

4.2.1–4. Note: this solution is for a corrected version of the problem. $(1 + a/x)^x = \exp[x \ln(1+a/x)] = \exp[-a + a^2/(2x) - a^3/(3x^2) + \Theta(1/x^3)] = e^{-a} \exp[a^2/(2x) - a^3/(3x^2) + \Theta(1/x^3)] = e^{-a}[1 + a^2/(2x) + a^4/(2x^2) - a^3/(3x^2) + O(1/x^3)] =$

$e^{-a}[1+a^2/(2x)+(3a^4-2a^3)/x^2+O(1/x^3)]$. The $(3a^4-2a^3)/x^2$ is positive for $a > 3/2$ and for $a < 0$.

4.2.1–5. See Cohen [23, p 131].

4.2.2–1. $1/[1+O(1/x)]$ is the set of functions between $1/(1+C/x)$ and $1/(1-C/x)$ for some constant C and for $x \geq x_0$. By long division, these limits are $1-(C/x)/(1+C/x)$ and $1+(C/x)/(1-C/x)$. Choose $x_1 = \max\{x_0, 2C\}$. For $x \geq x_1$, the upper limit is smaller than $1+(C/x)/(1-C/x_1)$ and the lower limit is larger than $1-(C/x)/(1-C/x_1)$, so $1-C'/x \leq 1/[1+O(1/x)] \leq 1+C'/x$, where $C' = C/(1-C/x_1) > 0$. So $1/[1+O(1/x)] = 1+O(1/x)$.

4.2.2–2. a/v goes to zero as v goes to infinity, but a does not change size. You can expand e^{-a} in a power series, you just don't know whether the error term will be small.

4.2.2–3. $e^{1/x} = 1+1/x+1/(2x^2)+O(1/x^3)$ so $e^{1/x}(1-1/x) = 1-1/(2x^2)+O(1/x^3)$.

4.2.2–4. $e^{-a}-\Theta(1/v)$ stands for the set of function in the range $e^{-a}-C_1/v$ to $e^{-a}-C_2/v$ for some positive constants C_1 and C_2. $e^{-a}+O(1/v)$ stands for the set of functions in the range $e^{-a} \pm C/v$. The first set is contained in the second one. $e^{-a}+O(1/v) = e^{-a}[1+e^aO(1/v)] = e^{-a}[1+O(1/v)]$ since e^a is a constant.

4.2.2–5. $(1-a/v^3)^{v^2} = \exp[v^2\ln(1-a/v^3)] = \exp[-a/v+O(1/v^4)] = 1+O(1/v)$.

4.2.2–6. $\ln(1+1/x^2) = 1/x^2-1/(2x^4)+1/(3x^6)-1/(4x^8)+O(1/x^{10})$, so $\ln(1+1/x^2)e^{1/x^3} = 1/x^2-1/(2x^4)+1/(3x^6)-1/(4x^8)+O(1/x^{10})+1/x^5-1/(2x^7)+O(1/x^9)+1/(2x^8)+O(1/x^{10}) = 1/x^2-1/(2x^4)+1/x^5+1/3x^6-1/(2x^7)+1/(4x^8)+O(1/x^9)$.

4.2.2–7. $[1-t^2/(2n)+O(t^3/n^{3/2})]^n = \exp\{n\ln[1-t^2/(2n)+O(t^3/n^{1/2})]\} = \exp[-t^2/2+O(t^3/n^{1/2})+O(t^4/n)] = \exp[-t^2/2+O(t^3/n^{1/2})]$.

4.3–1. Since $\lfloor\lg i\rfloor \leq \lg i < \lfloor\lg i\rfloor+1$, $(N+1)\lfloor\lg N\rfloor-2^{\lfloor\lg N\rfloor+1}+2 \leq \sum_{1\leq i\leq N}\lg i < (N+1)\lfloor\lg N\rfloor-2^{\lfloor\lg N\rfloor+1}+2+N$, $(N+1)(\lg N-1)-2N+2 < \sum_{1\leq i\leq N}\lg i < (N+1)\lg N-N+2+N$, $N\lg N-3N+\lg N+1 < \sum_{1\leq i\leq N}\lg i < N\lg N+\lg N+2$, $2^{N\lg N-3N+1} < N! < 2^{N\lg N+\lg N+2}$, $16(N/8)^{N+1} < N! < 4N^{N+1}$.

4.3.1.1–1. The probability that a clause contains variable x_1 is p. The probability that it contains x_1 and its negation is p^2. To have an average fraction f of a set of clauses containing x_1 and its negation, it is necessary that $f = p^2$.

4.3.1.1–2. The probability that a clause does not contain x_1 and its negation is $1-p^2$. The probability that a clause does not contain any variable and its negation is $(1-p^2)^v$. Therefore, the probability that it does contain a clause and its negation is $1-(1-p^2)^v$ and the answer to the problem is $f = 1-(1-p^2)^v$.

4.3.1.1–3. This problem is worked in the text. See eq.(110)

4.3.1.1–4. This problem and the next refer to eq.(111). Start with eq.(120). When $\lim_{v\to\infty} vp = 0$, $(1-p)^{v+1} = \exp[(v+1)\ln(1-p)] = \exp(-vp+O(vp^2)) = 1-vp+O(v^2p^2)$, so $N_{i_*} = \exp[v\ln 2+t-tvp+O(tv^2p^2)]$. This simplifies to $N_{i_*} = \exp(v\ln 2+t-tvp)[1+O(tv^2p^2)]$. Eq.(110) implies that $t < O(1/p)$. When $\lim_{v\to\infty} vp = \infty$, $(1-p)^{v+1} \to 0$, so the logarithm can be expanded in a power series to give $N_{i_*} = \exp\{v\ln 2-t(1-p)^{v+1}+O[t(1-p)^{2(v+1)}]\}$. When $t(1-p)^{v+1} \to 0$ this can be further simplified.

4.3.1.1–5. See reference [78] and apply the technique to this problem.

4.3.1.1–6. See reference [78].

4.4–1. For $n > 2$, $\sum_{1\leq i\leq n} i! = n!+(n-1)!+(n-2)!+(n-3)!+\sum_{1\leq i\leq n-4} i! = n!\{1+1/n+1/[n(n-1)]+1/[n(n-1)(n-2)]+O(1/n^3)\} = n!\{1+1/n+1/[n(n-1)]+O(1/n^3)\}$.

4.4–2a. The answer is a constant, so there is no useful big O answer. $10^{100} < \sum_{0\le i\le 10} i^{100} < 10^{100}(1+10(9/10)^{100}) < 1.0003\times 10^{100}$.

4.4–2b. The answer is a constant, so there is no useful big O answer. $1000^{1,000,000} < \sum_{0\le i\le 1000} i^{1,000,000} < 1000^{1,000,000}(1+1000(999/1000)^{1,000,000})$. The upper limit is less that one part in 10^{500} larger than the lower limit.

4.4–2c. $\sum_{0\le i\le 1000} i^x = 1000^x[1+O((999/1000)^x)]$.

4.4–2d. $\sum_{0\le i\le n} i^x = n^x\{1+O[(1-1/n)^x]\}$.

4.4–3. $\sum_{0\le i\le n}(i+1)^{m+1} - \sum_{1\le i\le n+1} i^{m+1} = 0$. It is also equal to $\sum_{0\le i\le n} i^{m+1} + \sum_{0\le i\le n}(m+1)i^m + O(n^m) + O(1) - \sum_{1\le i\le n+1} i^{m+1} = -(n+1)^{m+1} + \sum_{0\le i\le n}(m+1)i^m + O(n^m) + O(1)$. If we assume $m\ge 0$, this gives $\sum_{0\le i\le n}(m+1)i^m = (n+1)^{m+1} + O(n^m)$. But $(n+1)^{m+1} = n^{m+1} + O(n^m)$, so $\sum_{0\le i\le n}(m+1)i^m = n^{m+1} + O(n^m)$ for $m\ge 0$.

4.4–4. $\sum_{0\le i\le n}(i+1)^{m+1} - \sum_{1\le i\le n+1} i^{m+1} = 0$. $\sum_{0\le i\le n} i^{m+1} + \binom{m+1}{1}\sum_{0\le i\le n} i^m + \binom{m+1}{2}\sum_{0\le i\le n} i^{m-1} + O(n^{m-1}) - \sum_{1\le i\le n+1} i^{m+1} = 0$, so $(m+1)\sum_{0\le i\le n} i^m = (n+1)^{m+1} - 0^{m+1} - \binom{m+1}{2}\sum_{0\le i\le n} i^{m-1} + O(n^{m-1})$. Expanding $(n+1)^{m+1}$ with the binomial theorem and using the result of Exercise 3 for the right side sum gives $\sum_{0\le i\le n} i^m = n^{m+1}/(m+1) + n^m/2 + O(n^{m-1})$. Now carrying the calculations out to one more term, $\sum_{0\le i\le n} i^{m+1} + \binom{m+1}{1}\sum_{0\le i\le n} i^m + \binom{m+1}{2}\sum_{0\le i\le n} i^{m-1} + \binom{m+1}{3}\sum_{0\le i\le n} i^{m-2} + O(n^{m-2}) - \sum_{1\le i\le n+1} i^{m+1} = 0$, so $(m+1)\sum_{0\le i\le n} i^m = (n+1)^{m+1} - 0^{m+1} - \binom{m+1}{2}\sum_{0\le i\le n} i^{m-1} - \binom{m+1}{3}\sum_{0\le i\le n} i^{m-2} + O(n^{m-2})$. $\sum_{0\le i\le n} i^m = n^{m+1}/(m+1) + n^m/2 + mn^{m-1}/12 + O(n^{m-2})$.

4.4–5. For large x, the highest power with a nonzero coefficient is most important. $\sum_i \binom{n}{i}\binom{n}{j-i}x^i = \binom{n}{j}x^j[1+O(1/x)]$ if $j\le n$. If $j\ge n$, then the answer is $\binom{n}{j-n}x^n[1+O(1/x)]$.

4.4–6. For small x, the lowest power is most important. $\sum_i \binom{n}{i}\binom{n}{j-i}x^i = \binom{n}{j}[1+O(x)]$ if $j\le n$. If $j\ge n$, then the answer is $\binom{n}{j-n}x^{j-n}[1+O(x)]$.

4.4–7. The biggest term is for $j=n/2$, so for even n the sum is greater than $(n/2)^{2k}$ and smaller than n times that amount. For odd n, replace $n/2$ by $(n-1)/2$.

4.5–1. $\int_0^{n-1} x^{1/2}\,dx \le \sum_{1\le i<n} i^{1/2} \le \int_1^n x^{1/2}\,dx$, $\frac{2}{3}x^{3/2}|_0^{n-1} \le \sum_{1\le i<n} i^{1/2} \le \frac{2}{3}x^{3/2}|_1^n$, $\frac{2}{3}(n-1)^{3/2} \le \sum_{1\le i<n} i^{1/2} \le \frac{2}{3}(n^{3/2}-1)$.

4.5–2. $\sum_{1\le i<n} f(i) = \sum_{1\le i<m} f(i) + \sum_{m\le i<n} f(i)$. Apply eq.(146) to obtain the final answer.

4.5–3. $\sum_{1\le i<m} i^{-2} + \int_m^n x^{-2}\,dx \le \sum_{1\le i<n} i^{-2} \le \sum_{1\le i<m} i^{-2} + \int_{m-1}^{n-1} x^{-2}\,dx$, $\sum_{1\le i<m} i^{-2} - x^{-1}|_m^n \le \sum_{1\le i<n} i^{-2} \le \sum_{1\le i<m} i^{-2} - x^{-1}|_{m-1}^{n-1}$, $\sum_{1\le i<m} i^{-2} + m^{-1} - (n)^{-1} \le \sum_{1\le i<n} i^{-2} \le \sum_{1\le i<m} i^{-2} + (m-1)^{-1} - (n-1)^{-1}$. So the constant in the lower limit is $\sum_{1\le i<m} i^{-2} + m^{-1}$ and the one in the upper limit is $\sum_{1\le i<m} i^{-2} + (m-1)^{-1}$. For $m=1, 2, 3, 4$, the constant part of the lower limits are 1, $1+1/2 = 3/2$, $1+1/4+1/3 = 19/12$, $1+1/4+1/9+1/4 = 29/18$, and the constant part of the upper limits are $1/0 = \infty$, $1+1/1 = 2$, $1+1/4+1/2 = 7/4$, $1+1/4+1/9+1/3 = 61/36$.

4.5.1–1. $\sum_{1\le i<n} i^{1/2} = \int_1^n x^{1/2}\,dx - \frac{1}{2}(n^{1/2}-1) + \int_1^n B_1(\{x\})(d\,x^{1/2}/dx)\,dx = \frac{2}{3}n^{3/2} - \frac{1}{2}n^{1/2} - \frac{1}{6} - \frac{1}{2}\int_1^n B_1(x)x^{-1/2}\,dx$. Now, $-1/2 \le B_1(\{x\}) \le 1/2$, so $-\frac{1}{4}\int_1^n x^{-1/2}\,dx \le \frac{1}{2}\int_1^n B_1(x)x^{-1/2}\,dx \le \frac{1}{4}\int_1^n x^{-1/2}\,dx$. $\frac{1}{4}\int_1^n x^{-1/2}\,dx = \frac{1}{2} - x^{1/2}|_1^n = \frac{1}{2}(1-n^{1/2})$. Putting the pieces together gives the answer.

4.5.1–2. Do it the same way as 4.5–2.

4.5.1–3. $\sum_{1\le i<n} i^{-2} = \sum_{1\le i<m} i^{-2} + \int_m^n x^{-2}\,dx - \frac{1}{2}(n^{-2}-m^{-2}) + \int_m^n B_1(\{x\})(d\,x^{-2}/dx)\,dx = \sum_{1\le i<m} i^{-2} + m^{-1} - n^{-1} - \frac{1}{2}(n^{-2}-m^{-2}) - 2\int_m^n B_1(\{x\})x^{-3}\,dx$. The last integral is bounded by $\pm\frac{1}{2}(m^{-2}-n^{-2})$. Therefore, $\sum_{1\le i<m} i^{-2} + m^{-1} - n^{-1} \le \sum_{1\le i<n} i^{-2} \le \sum_{1\le i<m} i^{-2} + m^{-1} - n^{-1} + m^{-2} - n^{-2}$. For $m = 1, 2, 3, 4$, the constant part of these limits are 1, $1+1/2 = 3/2$, $1+1/4+1/3 = 19/12$, $1+1/4+1/9+1/4 = 29/18$, and $1+1 = 2$, $1+1/2+1/4 = 7/4$, $1+1/4+1/3+1/9 = 61/36$, $1+1/4+1/9+1/4+1/16 = 241/144$.

4.5.2–1 $\sum_{1\le i<n} i^{1/2} = \int_1^n x^{1/2}\,dx - \frac{1}{2}(n^{1/2}-1) + \frac{1}{24}(n^{-1/2}-1) - \frac{1}{1920}(n^{-5/2}-1) - \frac{1}{8}\int_1^n B_2(\{x\})x^{-3/2}\,dx = \frac{2}{3}n^{3/2} - \frac{1}{2}n^{1/2} - \frac{2}{3} + \frac{1}{2} - \frac{1}{24} + \frac{1}{1920} - \frac{1}{24}n^{-1/2} - \frac{1}{1920}n^{-5/2} - \frac{1}{8}\int_1^n B_2(\{x\})x^{-3/2}\,dx$. By eq.(168), the error term is no more than the last term. This gives a final answer between $\frac{2}{3}n^{3/2} - \frac{1}{2}n^{1/2} - \frac{399}{1920} + \frac{1}{24}n^{-1/2} - \frac{1}{1920}n^{-5/2}$ and $\frac{2}{3}n^{3/2} - \frac{1}{2}n^{1/2} - \frac{1}{8} + \frac{1}{24}n^{-1/2}$

4.5.2–2. $\sum_{1\le i<n} i^{-2} = \sum_{1\le i<k} i^{-2} + \int_k^n x^{-2}\,dx - \frac{1}{2}(n^{-2}-k^{-2}) - \frac{1}{6}(n^{-3}-k^{-3}) + \frac{1}{30}(n^{-5}-k^{-5}) + 5\int_k^n B_4(\{x\})x^{-6}\,dx = \sum_{1\le i<k} i^{-2} + k^{-1} - n^{-1} - \frac{1}{2}(n^{-2}-k^{-2}) - \frac{1}{6}(n^{-3}-k^{-3}) + \frac{1}{30}(n^{-5}-k^{-5}) + 5\int_k^n B_4(\{x\})x^{-6}\,dx$. The error term is no more than the last term, so for $k = 1, 2, 3$, the lower limit is $\frac{49}{30} - n^{-1} - \frac{1}{2}n^{-2} - \frac{1}{6}n^{-3} + \frac{1}{30}n^{-5}$, $\frac{1579}{960} - n^{-1} - \frac{1}{2}n^{-2} - \frac{1}{6}n^{-3} + \frac{1}{30}n^{-5}$, $\frac{23983}{14580} - n^{-1} - \frac{1}{2}n^{-2} - \frac{1}{6}n^{-3} + \frac{1}{30}n^{-5}$, and the upper limit is $\frac{5}{3} - n^{-1} - \frac{1}{2}n^{-2} - \frac{1}{6}n^{-3}$, $\frac{79}{48} - n^{-1} - \frac{1}{2}n^{-2} - \frac{1}{6}n^{-3}$, $\frac{533}{324} - n^{-1} - \frac{1}{2}n^{-2} - \frac{1}{6}n^{-3}$.

4.5.2–3. (Assume $k \ge 2$ and constant; for small k the answer can be obtained directly.) $\sum_{0\le i\le n} i^k = n^k + \int_0^n x^k\,dx - n^k/2 + kn^{k-1}/12 + [k(k-1)/2]\int_0^n B_2(\{x\})x^{k-2}\,dx = n^{k+1}/(k+1) + n^k/2 + kn^{k-1}/12 + [k(k-1)/2]\int_0^n B_2(\{x\})x^{k-2}\,dx$. The last term can be bounded in absolute value using $(2/\pi^2)\int_0^n x^{k-2}\,dx = 2n^{k-1}/[\pi^2(k-1)]$. This gives $\sum_{0\le i\le n} i^k = n^{k+1}/(k+1) + n^k/2 + O(n^{k-1})$, where the constant implied by the big O is between $k\{1/12 + 1/[\pi^2]\}$ and $k\{1/12 - 1/[\pi^2]\}$.

4.5.4–1. See [39, pp 527–528].

4.5.4–2. By eq. (167) the error is no more than $\frac{1}{132}n^{-10}$.

4.5.4–3. $\binom{n+1}{m+1}[H_{n+1} - 1/(m+1)]$. See [9, p 75].

4.5.4–4. $\sum_{1\le i<n} i^{-2} = \sum_{1\le i<\infty} i^{-2} - \sum_{n\le i<\infty} i^{-2} = \frac{1}{6}\pi^2 - \sum_{n\le i<\infty} i^{-2} = \frac{1}{6}\pi^2 - \int_n^\infty x^{-2}\,dx - \frac{1}{2}n^{-2} - \frac{1}{6}n^{-3} - 5\int_k^n B_4(\{x\})x^{-6}\,dx = \frac{1}{6}\pi^2 - n^{-1} - \frac{1}{2}n^{-2} + \frac{1}{6}n^{-3} + O(n^{-5})$.

4.5.4–5. $1 < \sum_{i\ge 0} i^{-2k} < 1 + \int_1^\infty x^{-2k}\,dx = 1 + 1/(2k-1) \le 2$, for $k \ge 2$. Direct evaluation of the first few terms of the sum leads to much more accurate results.

4.5.4.1–1. Use eq. (184). to obtain $5[-\ln(.8) - O(1/(N-k))] = 1.12 - O(1/N)$.

4.5.4.1–2. $1/N^2 < 1/(N-k)^2$ in the range of interest ($0 \le k < N$) so the $O(1/N^2)$ term can be absorbed into the $O(1/(N-k)^2)$ term.

4.5.4.1–3. Neither term can account for the error over the entire range of interest. The first big O term is dominant for k near $N/2$ while the second one is dominant for k near zero and near N.

4.5.4.1–4. In eq. (182) approximate the H_N (but not the H_{N-k}) to obtain $A = (N/k)[\ln N - \gamma + 1/(2N) + O(1/N^2) - H_{N-k}]$.

4.5.4.1–5. $A = \sum_{i\geq 1}(1/k)\sum_{0\leq j\leq k-1} j!(N-i)!(N-j)/[N!(j-i+1)!] = (1/k)\sum_{0\leq j\leq k-1} \sum_{i\geq 1}(N-i)!j!(N-j)/[N!(j-i+1)!]$. From eq. (3–93), the sum over i is equal to $(N+1)/(N-j+1)$, so $A = [(N+1)/k]\sum_{0\leq j\leq k-1} 1/(N-j+1) = [(N+1)/k](H_{N+1} - H_{N-k+1})$.

4.6–1. 100.

4.6–2. $\sum_{i\geq 1}(-1)^i/i = \sum_{j\geq 1}[-1/(2j-1)+1/(2j)] = -\sum_{j\geq 1} 1/[2j(2j-1)]$. $\sum_{i\geq 1}(-1)^i/i = -1+\sum_{i\geq 2}(-1)^i/i = -1+\sum_{j\geq 1}[1/(2j)-1/(2j+1)] = -1+\sum_{j\geq 1} 1/[2j(2j+1)]$.

4.6–3. By repeating the previous exercise with upper limits, you get $\sum_{1\leq i\leq 2n}(-1)^i/i = -\sum_{1\leq j\leq n} 1/[2j(2j-1)]$ and $\sum_{1\leq i\leq 2n+1}(-1)^i/i = -1+\sum_{1\leq j\leq n} 1/[2j(2j+1)]$. Now, applying eq. (203) (replacing n with ∞ and k with $2n$) gives the result. The value of the sum from $n+1$ to infinity can be approximated with an integral. About 25 terms are needed for accuracy ± 0.01.

4.6–4. Base cases. (1) $\sum_{0\leq i\leq k}(-1)^i a_i = \sum_{0\leq i\leq n}(-1)^i a_i$ for $k = n$, (2) $\sum_{0\leq i\leq k}(-1)^i a_i = \sum_{0\leq i\leq k+1}(-1)^i a_i + a_n < \sum_{0\leq i\leq n}(-1)^i a_i$ for $k = n-1$ and k even, (3) $\sum_{0\leq i\leq k}(-1)^i a_i = \sum_{0\leq i\leq k+1}(-1)^i a_i - a_n < \sum_{0\leq i\leq n}(-1)^i a_i$ for $k = n-1$ and k odd. Suppose $\sum_{0\leq i\leq k_*}(-1)^i a_i \geq \sum_{0\leq i\leq n}(-1)^i a_i \leq \sum_{0\leq i\leq k_*-1}(-1)^i a_i$, where k is even. Then $\sum_{0\leq i\leq k-2}(-1)^i a_i = \sum_{0\leq i\leq k}(-1)^i a_i - (a_{k-1} - a_k) > \sum_{0\leq i\leq k}(-1)^i a_i > \sum_{0\leq i\leq n}(-1)^i a_i$, which establishes the upper limit for $k = k_* - 2$. In a similar way the lower limit can be established. Thus, if eq. (203) is true for k_* (possibly with the greater thans replaced with equals), then it is true for $k_* - 2$ (with no replacement of the greater thans). The base cases provide the two required starting values.

4.7–1. $x_0 = t$, $x_1 = te^{-t}$, $x_2 = te^{-te^{-t}} = t[1 - te^{-t} + O(t^2e^{-2t})]$, $x_3 = -te^{-t+t^2e^{-t}+O(t^3e^{-2t})} = t\{1-t+t^2e^{-t}-O(t^3e^{-2t})+(1/2)[-t+t^2e^{-t}+O(t^3e^{-2t}]^2+O(t^3)\}$. (Perhaps more details would help.) This sequence is not converging. The error term is getting bigger. The problem is that e^{-x} is changing its relative size too quickly. The form with the logarithm is much better behaved.

4.7–2. Let $x = \ln t - \ln\ln t - a(\ln\ln t/\ln t)$ and plug in. The left side is $\ln t - \ln\ln t - a(\ln\ln t/\ln t)$ and the right side is $\ln t - \ln[\ln t - \ln\ln t - a(\ln\ln t/\ln t)] = \ln t - \ln\{\ln t[1 - (\ln\ln t)/\ln t - a(\ln\ln t)/(\ln t)^2]\} = \ln t - \ln\ln t + \ln[1 - (\ln\ln t)/\ln t - a(\ln\ln t)/(\ln t)^2] = \ln t - \ln\ln t + (\ln\ln t)/\ln t - a(\ln\ln t)/(\ln t)^2 + O((\ln\ln t)^2/(\ln t)^2)$. The first two terms on each side are equal. The third terms are equal for $a = -1$. All the remaining terms are much small (for large t) than the third terms. Therefore for large t and $a > -1$ the left side is bigger while for $a < -1$ the right side is bigger. Each side is a continuous function of a, so the two sides must be equal near $a = -1$.

4.7–3. Write the equation as $x = \sqrt{u + \ln x}$. Starting with $x_0 = \sqrt{u}$, we get $x_1 = \sqrt{u + \frac{1}{2}\ln u} = \sqrt{u}\sqrt{1 + (\ln u)/(2u)} = \sqrt{u}[1 + (\ln u)/(4u) + O((\ln u)^2/u^2)] =$

$u^{1/2} + (\ln u)/(4u^{1/2}) + O((\ln u)^2/u^{3/2})$,
$x_2 = \sqrt{u + \ln[u^{1/2} + (\ln u)/(4u^{1/2}) + O((\ln u)^2/u^{3/2})]} =$
$\sqrt{u}\{1 + \ln[u^{1/2}(1 + (\ln u)/(4u) + O((\ln u)^2/u^2))]/u\}^{1/2} = \sqrt{u}\{1 + \ln[u^{1/2}(1 +$
$(\ln u)/(4u) + O((\ln u)^2/u^2))]/(2u)$
$- [\ln[u^{1/2}(1 + (\ln u)/(4u) + O((\ln u)^2/u^2))]]^2/(8u^2)\} = \sqrt{u}[1 + (\ln u)/(4u) +$
$(\ln u)/(8u^2) + O((\ln u)^2/u^3) + 3(\ln u)^2/(16u^2) + O((\ln u)^2/u^3)]$
$= \sqrt{u}[1 + (\ln u)/(4u) + 3(\ln u)^2/(16u^2) + (\ln u)/(8u^2) + O((\ln u)^2/u^2)] = u^{1/2} +$
$(\ln u)/(4u^{1/2}) + 3(\ln u)^2/(16u^{3/2}) + (\ln u)/(8u^{3/2}) + O((\ln u)^2/u^{3/2})]$. x_3 is the same as x_2 to the calculated accuracy. You can prove that x_2 is the solution using the same technique as the used in the previous exercise.

5.2–1. $T_n = T_0 a^n + b(a^n - 1)/(a - 1)$.

5.2–2. $T_n = a^n + b(a^n - 1)/(a - 1)$.

5.2–3. $T_n = T_0 n!$.

5.2–4. $T_n = T_0 + n(n + 1)/2$.

5.2–5. By eq. (29), the solution is $T_n = T_0 \prod_{1\le i\le n} a_n$. If a_n is a rational function of n, then it can be factored (using complex numbers) into the form $(n - x_1)^{e_1}$ $(n - x_2)^{e_2} \cdots (n - x_j)^{e_j}/[(n - y_1)^{f_1}(n - y_2)^{f_2} \cdots (n - y_k)^{f_k}]$, so
$T_n = \prod_{1\le i\le n}(i - x_1)^{e_1} \prod_{1\le i\le n}(i - x_2)^{e_2} \cdots \prod_{1\le i\le n}(i - x_j)^{e_j}/$
$[\prod_{1\le i\le n}(i - y_1)^{f_1} \prod_{1\le i\le n}(i - y_2)^{f_2} \cdots \prod_{1\le i\le n}(i - y_k)^{f_k}]$.
Since $\prod_{1\le i\le n}(i - x) = \Gamma(n - x)/\Gamma(-x)$, $T_n =$
$T_0 \left([\Gamma(-y_1)]^{f_1}[\Gamma(-y_2)]^{f_2} \cdots [\Gamma(-y_k)]^{f_k}/\{[\Gamma(-x_1)]^{e_1}[\Gamma(-x_2)]^{e_2} \cdots [\Gamma(-x_k)]^{e_k}\}\right)$
$\left([\Gamma(-x_1)]^{e_1}[\Gamma(-x_2)]^{e_2} \cdots [\Gamma(-x_k)]^{e_k}/\{[\Gamma(-y_1)]^{f_1}[\Gamma(-y_2)]^{f_2} \cdots [\Gamma(-y_k)]^{f_k}\}\right)$.

5.2–6. $U_n = a_{2n}U_{n-1} + b_{2n} = T_0 \prod_{1\le i\le n} a_{2i} + \sum_{1\le i\le n} b_{2i} \prod_{i<j\le n} a_{2j}$,
$V_n = a_{2n+1}V_{n-1} + b_{2n+1} = T_1 \prod_{1\le i\le n} a_{2i+1} + \sum_{1\le i\le n} b_{2i+1} \prod_{i<j\le n} a_{2j+1}$,
$T_n = (1 + (-1)^n)U_{n/2}/2 + (1 - (-1)^n)V_{(n-1)/2}/2$. (You can object to the answer because U and V have not been defined for half integer indices, but the answer *works* for any definition.)

5.2.1–1. The solution to the secondary recurrence is $d_i = (3n - 4)/(3 \cdot 4^i) + 4/3 = (2 \cdot 4^{k-i} + 4)/3$ where $k = \frac{1}{2}\lg(3n-4) - \frac{1}{2}$. ($n$ must have the form $2 \cdot 4^k/3 + 4/3$.) The solution to the recurrence is $T_n = 2 \cdot 3^k - 1 = 2(3n-4)^{\frac{1}{2}\lg 3}/3^{1/2} - 1 \approx 1.155(3n-4)^{0.792} - 1$.

5.2.1–2. 3 to $4 = 0.75$.

5.2.1–3. In this case for eq. (39), $T_n = 3^k + 2n[(\frac{3}{2})^k - 1]$, where $k = \lg n - \lg 3$, so the ratio for large n is $7/3^{\lg 3}$ to $4 \approx 0.307$. This method is fairer because if the second algorithm takes one unit of time to do problems of size 3 then the first one probably does also.

5.2.1–4. The algorithm needs time t_1 time to go through the current level of the recurrence. When there are $2n + 1$ item, there is one chance in $2n + 1$ that the item is found. Otherwise (with probability $2n/(2n + 1)$) it is necessary to solve a problem of size T_n. Problems of size 1 can be solved directly.

5.2.1–5. The solution to the recurrence of the previous exercise has the secondary recurrence $2d_{i+1} + 1 = d_i$, which has the solution $d_i = (n + 1)/2^i - 1$. Letting $k = \lg(n + 1) - 1$, the solution to the recurrence is $T_n = t_2 \prod_{0\le i<k}[1 - 2^i/(2n + 1)] + t_1 \sum_{0\le i<k} \prod_{0\le j<i}[1 - 2^j/(2n+1)]$. To finish the problem, it is necessary to determine how this answer behaves for large n. On computers which do not have

three-way branch instructions, it is necessary to multiply this result by a factor to account for the fact that on the average it takes about $1\frac{1}{2}$ two-way branches to do a three way branch. (The factor is smaller than 1.5, because much of the loop concerns computing the index rather than comparing; an analysis of the assembly language code would be needed to determine the exact factor.)

5.2.1–6. The secondary recurrence is $d_{i+1}^2/2^r = d_i$, or $2\lg d_{i+1} - r = \lg d_i$. The solution of the secondary recurrence for the boundary condition $d_k = 4^r$ is $d_i = 2^{r(2^{k-i}+1)}$. Perhaps the easiest thing to do is to write the original recurrence in terms of F_i where F_i is defined to be $T(2^{r(2^i+1)})$. This gives $F_i = 2^{r(2^{i-1}+1)}F_{i-1} + 2^{2r(2^{i-1}+1)}$, which has the solution $F_k = F_0 2^{r(2^k+k-1)} + b2^{r(2^k+k+2)}/(1-2^{-r}) - b2^{r(2^k+1)}/(1-2^{-r})$. Now $k = \lg[(\lg n)/r - 1]$ so

$$T(n) = 2^r n \left(\frac{\lg n}{r} - 1\right)^r \left(\frac{T(4^r)}{8^r} + \frac{b}{1-2^{-r}}\right) - \frac{bn}{1-2^{-r}}.$$

5.2.2–1. The second method is faster for $x < 2$. It might be faster for $x = 2$ (unlikely in practice). For the second method, the time is linear for $x < 1$, $O(n \ln n)$ for $x = 1$, and $O(n^x)$ for $x > 1$.

5.2.2–2. The second method is better for $k < 8$. From the information given, you cannot tell which method is better for $k = 8$ (in practice the first probably will be better).

5.2.3.1–1. Let T_n be the time for going down to $n = 3$, U_n be the time for going down to $n = 18$, and V_n be the time for nonrecursive algorithm. $T_n = V_3(n-2)^{\lg 3} + 6(n-2)^{\lg 3} - 6(n-2) - 1 = 18(n-2)^{\lg 3} - 6n + 11$. $U_n = (V_{18}/3^4)(n-2)^{\lg 3} + (6/3^4)(n-2)^{\lg 3} - (6/2^4)(n-2) - 1 = (116/27)(n-2)^{\lg 3} + (3/8)n - 1/4$. For large n the leading terms are the most important, and the ratio of the coefficients is $18/(116/27) = 486/116 \approx 4.19$, so the analysis suggests that stopping the recurrence at 18 instead of 3 increases the efficiency by over a factor of 4. This is the answer intended by the authors. It is actually too large. The analyses for T_n and U_n were based on upper limits rather than exact equations. Use of the exact equations would make the analysis much more difficult, but it would also give a lower (and correct) answer for the amount of improvement. See Exercise 5.2.3.1–3 for an indication of what must be considered in a more careful analysis.

5.2.3.1–2. When students at Indiana University did this problem, they found break points in the 18–36 range for programs written in Pascal. The students with the better (faster) programs often found the higher break points.

5.2.3.1–3. Using the assumptions from the text with even n, $n_*^2 + n_* = (n_*/2+1)^2 + (n_*/2+1) + 2(n_*/2)^2 + 2(n_*/2) + 3n_*$, or $n_*^2/4 - 9n_*/2 - 2 = 0$. The solution is $9 \pm \sqrt{89}$. The positive root is between 18 and 19. For odd n, $n_*^2 + n_* = 2(n_*/2+1)^2 + 2(n_*/2+1) + (n_*/2)^2 + (n_*/2) + 3n_*$, or $n_*^2/4 - 9n_*/2 - 2 = 0$. The solution is $11 \pm \sqrt{137}$. The positive root is between 21 and 22. Thus, at least one recursion should be done for $n = 20$ and for $n \geq 22$, but the direct approach should be used for $n = 21$ and for $n \leq 19$. In practice, one would probably chose to have a single break point (remember it would take extra time to have several break points). The break-even point calculated in the text is too large by over a factor of two. For a real program there are additional factors such as the time for parameter passing. For this reason it can be very valuable to

check an analysis against some actual measurements. Measurements don't lead to analytical answers, but they usually do not omit important effects. Comparing an analysis with measurements can uncover effects that might be missed using either technique by itself.

5.2.4–1. The sequence was chosen so that the middle element was first and so that each call to Split algorithm interchanges only one pair of elements (the last element in the first part is interchanged with the middle element). After this one interchange, the group of elements less than the middle element obeys this property as do the elements larger than the middle element. The same is true of the two parts, and the same continues to hold for the subparts.

5.2.4–2. See the previous answer.

5.2.4–3. For each call to the algorithm, steps 3 and 6 together are done a total of $2^k = n+1$ times. If we exit at Step 7 without going around the loop, then Steps 1 and 2 are done once, the test in Step 4 is done once, Step 5 is done once and all of Step 7 is done once. Steps 8 and 9 are done once. The algorithm also generates two recursive calls, one for a problem of size n_1 and one for a problem of size $n-n_1-1$ (all elements except the splitting element go into one set or the other).

5.2.4–4. This type of problem is discussed in much more detail in Section 7.4. Use induction. Assume that the result has been shown for all $k < n$. Then $T_n = an + b + a[(n_1+1)/2]\lg[(n_1+1)/2] + (b-a+c)n_1/2 + (c-a-b)/2 + a[(n_2+1)/2]\lg[(n_2+1)/2] + (b-a+c)n_2/2 + (c-a-b)/2 = (a/2)\{(n_1+1)\lg[(n_1+1)/2] + (n_2+1)\lg[(n_2+1)/2]\}(a+b+c)n/2 + (c-a-b)/2$. Only the first term depends on n_1. Replacing n_2 with $n-1-n_1$, we must find the value of n_1 that minimizes $(n_1+1)\lg[(n_1+1)/2] + (n-n_1)\lg[(n-n_1)/2]$. Taking the derivative with respect to n_1 and setting it equal to zero gives $\lg[(n_1+1)/2] + 1 - \lg[(n-n_1)/2] - 1 = 0$ or $\lg[(n_1+1)/(n-n_1)] = 0$. Raising 2 to the power given by the equation gives $(n_1+1)/(n-n_1) = 1$, $2n_1 = n-1$, or $n_1 = (n-1)/2$ and $n_2 = (n-1)/2$. These values give a minimum (rather than a maximum) because the derivative is an increasing function of n_1.

5.2.4–5. By eq. (38), $T_{2^{k+1}-1} = 2^k T_1 + a\sum_{0\le i<k} 2^i(2^{k-i+1}-1) + b\sum_{0\le i<k} 2^i = c2^k + a(k2^k - (2^k-1)) + b(2^k-1) = c2^k + a(k-1)2^k + a + b(2^k-1)$. Expressing everything in terms of n gives $T_n = c(n+1)/2 + a\{\lg[(n+1)/2]-1\}(n+1)/2 + a + b(n-1)/2 = a(n+1)\lg[(n+1)/2] + cn/2 - an/2 + bn/2 + c/2 - a/2 - b/2$.

5.2.5–1. If you write down what is calculated you get $S_1 = A_{21}+A_{22}$, $S_2 = A_{21}+A_{22}-A_{11}$, $S_3 = A_{11}-A_{21}$, $S_4 = A_{11}+A_12-A_{21}-A_{22}$, $S_5 = B_{12}-B_{11}$, $S_6 = B_{11}+B_{22}-B_{12}$, $S_7 = B_{22}-B_{12}$, $S_8 = B_{11}+B_{22}-B_{12}-B21$, $M_1 = (A_{21}+A_{22}-A_{11})(B_{11}+B_{22}-B_{12}) = -A_{11}B_{11} + A_{11}B_{12} - A_{11}B_{22} + A_{21}B_{11} - A_{21}B_{12} + A_{21}B_{22} + A_{22}B_{11} - A_{22}B_{12} + A_{22}B_{22}$, $M_2 = A_{11}B_{11}$, $M_3 = A_{12}B_{21}$, $M_4 = (A_{11}-A_{21})(B_{22}-B_{12}) = -A_{11}B_{12} + A_{11}B_{22} + A_{21}B_{12} - A_{21}B_{22}$, $M_5 = (A_{21}+A_{22})(B_{12}-B_{11}) = -A_{21}B_{11} + A_{21}B_{12} - A_{22}B_{11} + A_{22}B_{12}$, $M_6 = (A_{11}+A_12-A_{21}-A_{22})B_{22} = A_{11}B_{22} + A_12B_{22} - A_{21}B_{22} - A_{22}B_{22}$, $M_7 = A_{22}(B_{11}+B_{22}-B_{12}-B_{21}) = A_{22}B_{11} - A_{22}B_{12} - A_{22}B_{21} + A_{22}B_{22}$, $T_1 = A_{11}B_{12} - A_{11}B_{22} + A_{21}B_{11} - A_{21}B_{12} + A_{21}B_{22} + A_{22}B_{11} - A_{22}B_{12} + A_{22}B_{22}$, $T_2 = A_{21}B_{11} + A_{22}B_{11} - A_{22}B_{12} + A_{22}B_{22}$, $C_{11} = A_{11}B_{11} + A_{12}B_{21}$, $C_{12} = A_{11}B_{12} + A_{21}B_{22}$, $C_{21} = A_{21}B_{11} + A_{22}B_{21}$, $C_{22} = A_{21}B_{12} + A_{22}B_{22}$.

5.2.5–2. $T_n = n^{\lg 7} \approx n^{2.81}$.

5.2.5–3. $T_n = n^{\lg 7} + n^2 \sum_{0\le i<\lg n}(7/4)^i = n^{\lg 7} + n^2[(7/4)^{\lg n} - 1]/[(7/4) - 1] = n^{\lg 7} + (4/3)n^{\lg 7} - (4/3)n^2 = (7/3)n^{\lg 7} - (4/3)n^2 \approx 2.3n^{2.81}$.

5.2.5–4. The time for the 3×3 method is $O(n^{\log_3 k})$, while the time for Strassen's method is $O(n^{\lg 7})$, so for the 3×3 method to be fast, we need $\log_3 k < \lg 7$, $k < 3^{\lg 7} \approx 21.8$. Similarly, to be better than the classical method requires $k < 27$.

5.2.5–5. $an^3 + bn^2 = 7(an^3/8 + bn^2/4) + cn^2/4$, $(a/8)n^3 = [(3/4)b + c/4]n^2$, $n = (6b + 2c)/a$. (The roots at $n = 0$ do not give the break-even point.)

5.2.5–6. See Spiess [131] for an indication of what answers to expect.

5.3.1–1. Consider the sum $\sum_{0\le i\le n}(F_{i+2} - F_{i+1} - F_i)$ which is equal to zero by the definition of Fibonacci numbers. It is also equal to $\sum_{2\le i\le n+2} F_i - \sum_{1\le i\le n+1} F_i - \sum_{0\le i\le n} F_i = F_{n+2} + F_{n+1} - 1 - 1 - F_{n+1} + 1 - \sum_{0\le i\le n} F_i = F_{n+2} - 1 - \sum_{0\le i\le n} F_i$. Since this sum is zero, we obtain $\sum_{0\le i\le n} F_i = F_{n+2} - 1$. This technique can be extended to do the sum $\sum_{0\le i\le n} F_i x^i$, as shown in Knuth [9, Section 1.2.8].

5.3.1–2. $bF_{n+1} + aF_n$. The recurrence is the Fibonacci recurrence, so you can just consider how much of F_n and F_{n-1} you need to match the boundary conditions.

5.3.1–3. Let A_n be the number of additions needed. Then $A_n = 1 + A_{n-1} + A_{n-2}$ with boundary conditions $A_0 = A_1 = 0$, because to calculate F_n one addition is done on the results of recursive calls to the procedure with parameters $n-1$ and $n-2$. Let $G(z) = \sum_{i\ge 0} A_i z^i$. Then $G(z) - A_0 - zA_1 = z^2/(1-z) + zG(z) - zA_0 + z^2G(z)$, so $G(z) = z^2/[(1-z)(1-z-z^2)] = -1/(1-z) + 1/(1-z-z^2)$, and $A_n = F_n - 1$.

5.3.1–4. One addition is done for each time around the loop, so $n-1$ additions are done. For large n, this algorithm uses many fewer additions than the previous one.

5.3.3.1–1. The answer is twice that given by eq. (158). Eq. (158) was determined by the amount of output needed. The answer to this problem is determined by the amount of input plus output, which is just twice the amount of output.

5.3.3.1–2. This has the same answer as the last exercise. Each number that is output needs to be rewound.

5.3.3.1–3. This time the answer is three times that given by eq. (158).

5.3.3.2–1. The answer is twice that given by eq. (178).

5.3.3.2–2. The answer is twice that given by eq. (178).

5.3.3.2–3. The answer is three times that given by eq. (178).

5.3.3.2–4, 5, 6. The comparison is the same as the one given in the text following eq. (158).

5.4–1. For year n let z_n, o_n, t_n, and f_n be the number of new born foxes, the number of one year old foxes, the number of two year old foxes, and the total number of foxes respectively. Then $z_n = o_n + t_n$, $o_n = z_{n-1}$, $t_n = o_{n-1}$, and $f_n = z_n + o_n + t_n$. The first three equations can be combined to give $z_n = z_{n-1} + z_{n-2}$, so $z_n = c_1F_n + c_2F_{n-1}$, where F_n is the n^{th} Fibonacci number. The boundary conditions are $z_0 = 0$ and $z_1 = 1$, so $z_n = F_n$. From the second and third equations we have $o_n = F_{n-1}$ and $t_n = F_{n-2}$ (for $n > 1$), so $f_n = F_n + F_{n-1} + F_{n-2} = 2F_n$ (for $n > 1$). Also (by direct calculation) $f_0 = 1$ and $f_1 = 2$. Thus, $f_0 = 1$ and $f_n = 2F_n$ for $n \ge 1$.

5.4.1–1. If $\sum_{0\le i\le k} a_{k-i}\lambda^i = 0$ then (replacing i with $k-i$) $\sum_{0\le i\le k} a_i\lambda^{k-i} = 0$, so $\sum_{0\le i\le k} a_i\lambda^{-i} = 0$ unless $\lambda = 0$.

5.4.1–2. $T_n = T_{n-i} + T_{n-j} + O(n^x)$.

5.4.1–3. Let b_{ij} be the largest solution of the equation $1 - b_{ij}^{-i} - b_{ij}^{-j} = 0$. The requested solutions are $b_{11} = 2$, $b_{12} = (1+\sqrt{5})/2 \approx 1.6180$ $b_{22} = \sqrt{2} \approx 1.4142$, $b_{13} = \frac{1}{3}[(\frac{29}{2} + \frac{3}{2}\sqrt{93})^{1/3} + (\frac{29}{2} + \frac{3}{2}\sqrt{93})^{1/3}] \approx 1.3493$, $b_{23} = (\frac{1}{2} + \frac{1}{6}\sqrt{\frac{23}{3}})^{1/3} + (\frac{1}{2} - \frac{1}{6}\sqrt{\frac{23}{3}})^{1/3} \approx 1.3247$, $b_{33} = 2^{1/3} \approx 1.2599$, $b_{14} = 1.3803$, $b_{24} = \sqrt{(1+\sqrt{5})/2} \approx 1.2720$, $b_{34} = 1.2207$ $b_{44} = 2^{1/4} \approx 1.1892$.

5.4.2–1. The general solution is $T_n = c_1 + c_2 n$, so the particular solution is $T_n = n$.

5.4.2–2. $F_n = c_1 n + c_0$ works when $c_1(n+2) + c_1(n+1) + c_1 n + 3c_0 = n$. This is true when $3c_1 = 1$ and $3c_0 + 3c_1 = 0$, or $c_1 = 1/3$ and $c_0 = -1/3$. The characteristic solution of the homogenous equation is $z^2 + z + 1 = 0$, which has roots $z = (-1 \pm \sqrt{-3})/2$. Letting $\lambda_1 = (-1 + i\sqrt{3})/2$ and $\lambda_2 = (-1 - i\sqrt{3})/2$, the general solution is $F_n = c_1\lambda_1^n + c_2\lambda_2^n + n/3 - 1/3$.

5.4.2–3. Look for solutions of the form $F_n = an^2 2^n + bn2^n + c2^n$. $a(n+3)^2 2^{n+3} + b(n+3)2^{n+3} + c2^{n+3} - a(n+2)^2 2^{n+2} - b(n+2)2^{n+2} - c2^{n+2} + a(n+1)^2 2^{n+1} + b(n+1)2^{n+1} + c2^{n+1} - an^2 2^n - bn2^n - c2^n = n^2 2^n$. $(5a-1)n^2 2^n + (36a+5b)n2^n + (58a+18b+5c)2^n = 0$. Since each coefficient must be zero, $a = 1/5$, $b = -36/25$, and $c = -358/125$.

5.4.2–4. Correct the problem to read $T_n = 4T_{n-1} - 5T_{n-2}$. The general solution is $T_n = a(2+i)^n + b(2-i)^n$. The particular solution is $T_n = i(2+i)^n/2 - i(2-i)^n/2$.

5.4.2–5. Let $\theta = \arctan(1/2)$. Then $(2 \pm i) = \sqrt{5}(\cos\theta \pm i\sin\theta)$ and $(2 \pm i)^n = 5^{n/2}(\cos n\theta \pm i\sin n\theta)$.

5.4.3–1. The average length is $\sum_{k\geq 0} kp_k = (1-\rho)\sum_{k\geq 0} k\rho^k = \rho/(1-\rho)$.

5.4.3–2. $p_0 = 1 - \rho$. Letting l be the average queue length, $l = \rho/(1-\rho)$, so $\rho = 1/(l+1)$, $p_0 = 1/(l+1)$, and $l = (1-p_0)/p_0$. To have 20 percent idle time, l must be 4. To have 10 percent idle time, l must be 9.

5.5.1–1. In this answer, we will stick all the effects of rounding up and of rounding down into a big O term of the recurrence. For $c = 7$, we have $C(n) = C(n/7) + C(5n/7) + (16/7)n + O(1)$. Letting $C(n) = \alpha n + \beta$, we get $(1 - 1/7 - 5/7)\alpha = 16/7$, or $\alpha = 16$, which is worse than $\alpha = 14\frac{2}{3}$. For $c = 11$, we have $C(n) = C(n/11) + C(8n/11) + (30/11)n + O(1)$. Letting $C(n) = \alpha n + \beta$, we get $(1 - 1/11 - 8/11)\alpha = 30/11$, or $\alpha = 15$, which is worse than $\alpha = 14\frac{2}{3}$.

5.5.1–2. For n a multiple of 18, the question is when is $S(n) > S(n/9+1) + S(13n/18+4) + 22n/9$, where $S(n)$ is the time used to sort n numbers. If you assume $S(n) = n\lceil \lg n \rceil - 2^{\lceil \lg n \rceil} + 1$ (see [11, p. 184]), then the recursive method uses less comparisons for $n \geq 90$.

6.1–1. $G(z) - a_1 z - a_0 = zG(z) - a_0 z + z^3(dG(z)/dz) + z^2 G(z)$. $z^3(dG(z)/dz) = (1 - z - z^2)G(z) + (a_0 - a_1)z + a_0$, $z^3(dG(z)/dz) = (1 - z - z^2)G(z) + 1$.

6.1–2. The last element is either in a 1-cycle or a 2-cycle. If it is in a 1-cycle, then the remaining $n-1$ elements form an involution of $n-1$ elements. If the last element is in a 2-cycle then there are $n-1$ possible elements for second element of the 2-cycle. In these cases, the $n-2$ elements that are not part of that 2-cycle form an involution of $n-2$ elements. Thus, $t_n = t_{n-1} + (n-1)t_{n-2}$.

6.1–3. The boundary conditions are $t_0 = 1$ and $t_1 = 1$, so this exercise has the same answer as Exercise 1.

6.1.2–1. The generating function obeys the differential equation $G'' - 2G' - 3G = 0$ subject to the initial conditions $G(0) = 2$ and $G'(0) = 2$. Therefore the generating function is $G(x) = e^{3x} + e^{-x} = \sum_{i\geq 0}[3^i + (-1)^i]x^i/i!$, and $F_n = [3^n + (-1)^n]/n!$

6.1.2–2. $G(x, z) = (1 - 2xz + x^2)^{-1/2}$.

6.1.2–3. The correct recurrence is $J_{n+1}(z) - \frac{2n}{z}J_n(z) + J_{n-1}(z) = 0$, $G(x, z) = \exp[\frac{1}{2}z(x - x^{-1})]$.

6.1.2–4. You can show that the given hypergeometric function is a solution by plugging it in. The following indicates how you can obtain a general solution to the equation. First, there are several special cases that are easy to solve, including (1) $c = 0$, (2) $a = b = 0$, and (3) $a = 0$, $c = -1$. Expressing the solution to these special case cases as hypergeometric functions gives some clue to the form of the general answer. The author found the general solution by looking up known results on the hypergeometric function in [16, p. 558] and adjusting coefficients so that the exercise looked like eq. 15.2.10 of [16]. Using capital letters for the parameters in [16], the two equations match when $A = nc/(Z - 1)$, $B = (a + b)/Z)$, and $C = b - n - nc/(Z - 1)$ for any Z. Choosing $Z - 1 = c$ and $Z - 1 = -c$ gives the general solution $F_n = K_{12}F_1[n, (a+b)/(1+c); b; 1+c] + K_{22}F_1[-n, (a+b)/(1-c); b - 2n; 1 - c]$.

6.2.1–1. To calculate $B(n)$ of the recurrence $a_0(n)B_n + a_1(n)B_{n-1} + a_2(n)B_{n-2} + \cdots + a_k(n)B_{n-k} = 0$, use $B_n = -[a_1(n)B_{n-1} + a_2(n)B_{n-2} + \cdots + a_k(n)B_{n-k}]/a_0(n)$ (so long as $a_0(n) \neq 0$. This gives zero if $B_{n-1} = 0$, $B_{n-2} = 0$, ..., $B_{n-k} = 0$. If you have zeros up to n, and you have at least k zeros in a row, then this gives you zeros up to $n + 1$ and you still have at least k zero in a row. To calculate B_{n-k} use $B_{n-k} = -[a_0(n)B_n + a_1(n)B_{n-2} + \cdots + a_{k-1}(n)B_{n-k+1}]/a_k(n)$ (so long as $a_k(n) \neq 0$. The rest of the argument is similar to the previous case.

6.2.1–2. Consider the following set of k solutions to a recurrence in the form of eq. 41, where i $(0 \leq i \leq k-1)$ indexes the solution and j $A_i(n) = \delta_{in}$ for $0 \leq n \leq k-1$, $A_i(n) == -[a_1(n)A_i(n-1) + a_2(n)A_i(n-2) + \cdots + a_{k-1}(n)A_i(n-k)]/a_0(n)$ for $n \geq k$, $A_i(n-k) == -[a_0(n)A_i(n) + a_1(n)A_i(n-1) + \cdots + a_{k-1}(n)A_i(n-k+1)]/a_k(n)$ for $n < 0$, Suppose you have some other solution $A_k(n)$. Define $B(n) = \sum_{0\leq i\leq k-1} A_{k+1}(i)A_i(n)$. Then $B(n) - A_k(n) = 0$ for $0 \leq n \leq k-1$. Also $B(n) - A_k(n)$ is a solution of the equation since any linear combinations of solutions is a solution. Finally by the previous exercise, $B(n) - A_k(n) = 0$ for all n, so A_k is a linear combination of the other A_i.

6.2.1–3. $W_n = B_1(n)B_2(n+1) - B_1(n+1)B_2(n)$. $W_{n+1} = B_1(n+1)B_2(n+2) - B_1(n+2)B_2(n+1)$. $B_i(n+2) = [-a_1(n+2)B_i(n+1) - a_2(n+2)B_i(n)]/a_0(n+2)$ for $i = 1, 2$, so $W_{n+1} = B_1(n+1)[-a_1(n+2)B_2(n+1) - a_2(n+2)B_2(n)]/a_0(n+2) - B_2(n+1)[-a_1(n+2)B_1(n+1) - a_2(n+2)B_1(n)]/a_0(n+2) = -B_1(n+1)a_2(n+2)B_2(n)/a_0(n+2) + a_2(n+2)B_1(n)B_2(n+1)/a_0(n+2) = -a_2(n+2)W_n/a_0(n+2)$.

6.2.2–1. $r_1 = (n-2)^2$, $r_2 = -n + 2$, $D_n = 2D_2/[(n-2)!n!]$, giving a solution of $F_n = C_1 n + C_2 \sum_{2\leq i\leq n} 1/[(i-2)!i!]$.

6.2.2–2. See Bender and Orszag [21, p. 43, Example 5].

6.2.2–3. Write the recurrence as $F_n - [(2n-3)/(n-2)]F_{n-1} + [(n-2)/(n-3)]F_{n-2} = (n-1)(n-2)$. $r_1(n) = n - 1$, $r_2(n) = n - 1 - [(2n-3)/(n-2)](n-2) = -n + 2$, $r_3(n) = n - 1 - [(2n-3)/(n-2)](n-2) + [(n-2)/(n-3)](n-3) = 0$. $(n-1)D_n =$

$(n-2)D_{n-1}$. $D_n = D_2/(n-1)$. $B_n - B_{n-1} = c/(n-1)$. $B_n = B_1 + cH_{n-1}$. $A_n = b(n-1)+c(n-1)H_n$ is the solution to the homogeneous equation. Trying a polynomial for the particular solution gives $F_n = (n-1)(n^3-3n^2+8n)/3$, so the general solution is $F_n = (n-1)(n^3-3n^2+8n)/3 + b(n-1) + c(n-1)H_n$.

6.2.2–4. See Bender and Orszag [21, pp 44-46].

6.2.2–5. This is an Euler recurrence equation. $r(r-1)+r-1/4 = 0$, $r = \pm 1/2$. $F_n = a\Gamma(n+1/2)/\Gamma(n) + b\Gamma(n-1/2)/\Gamma(n)$.

6.2.2–6. $r(r-1)-1/4 = 0$. $r = (1\pm\sqrt{2})/2$. $F_n = a\Gamma(n+(1+\sqrt{2})/2)/\Gamma(n) + b\Gamma(n+(1-\sqrt{2})/2)/\Gamma(n)$.

6.2.3–1. Clearly any permutation of one object leaves the object fixed, so there are zero derangements of one object. There are two permutations of two objects, (1)(2) and (12). The second one is a derangement and the first one is not, so $D_2 = 1$. Now consider a derangement of n objects. The last object is either part of a 2-cycle or it is part of a larger cycle. If the last object is part of a 2-cycle, then removing the last object and its partner leaves a derangement of $n-2$ objects. Each derangement of $n-2$ objects along with a choice for the partner of the n^{th} object leads to a unique derangement of n objects, and every derangement of n objects with the last object in a 2-cycle can be obtained in just one way. There are $n-1$ choices for the partner. This explains the $(n-1)D_2$ term in the recurrence. If the last object is not part of a 2-cycle, then dropping the last object from its cycle gives a derangement of $n-1$ objects. Each derangement of $n-1$ objects along with a choice of an object to permute into the n^{th} object leads to a unique derangement of n objects, and every derangement of n objects with the last object in a cycle with more than two objects can be obtained in just one way. There are $n-1$ choices for the object to permute into the n^{th} object. This explains the $(n-1)D_1$ term in the recurrence.

6.2.3–2. Write the recurrence as $D_{n+2}-(n+1)D_{n+1}-(n+1)D_n = 0$, $\mathbf{E}^2D_n-(n+1)\mathbf{E}D_n-(n+1)D_n = 0$, $\mathbf{E}^2D_n-\mathbf{E}(n+2)D_n-(n+1)D_n = 0$, $\mathbf{E}^2D_n-\mathbf{E}(n+1)D_n-\mathbf{E}D_n-(n+1)D_n = 0$, $[\mathbf{E}+1][\mathbf{E}-(n+1)]D_n = 0$. Solving $(\mathbf{E}-1)Y_i = 0$ gives $Y_i = (-1)^i c$, where c is a constant. Solving $[\mathbf{E}-(n+1)]D_n = (-1)^i c$ gives $D_n = n!\sum_{1\le i\le n}(-1)^{i+1}/i!$.

6.2.3–3. The sum is the first n terms of the power series for e^{-x} with $x = 1$. It is an alternating sum where each term is smaller in absolute value than the one before, so the difference between the given sum and the corresponding infinite sum is less than the last term in the sum.

6.2.6–1. The roots of the characteristic equation are $\phi_1 = (-1+\sqrt{3}i)/2$ and $\phi_2 = (-1+\sqrt{3}i)/2$. The easiest way to finish the problem is to try to find a particular solution of the form $c2^n$ (this works because 2 is not a root of the characteristic equation). $2^n/7$ is a particular solution, so the general solution is $F_n = a\phi_1^n + b\phi_2^n + 2^n/7$.

6.2.6–2. The short-cut of the previous exercise does not work in this case. The roots of the characteristic equation are both 2. Letting $B_n = 2^n$ and $D_n = n2^n$, $W_n = 2^n(n+1)2^{n+1}-2^{n+1}n2^n = 2^{2n+1}$. $F_n = a2^n+bn2^n-2^n\sum_{2\le i\le n}[(i-1)2^{i-1}2^i]/[42^{2i-3}]-n2^n\sum_{2\le i\le n}[2^{i-1}2^i]/[42^{2i-3}] = a2^n+bn2^n-2^n\sum_{2\le i\le n}[(i-1)-n2^n\sum_{2\le i\le n}1 = a2^n+bn2^n-2^n n(n-1)/2-n2^n(n-1) = a'2^n+b'n2^n-n^2 2^{n-1}$.

6.2.6–3. $G_n = 1$ is a particular solution to the homogeneous equation. Reducing the order gives $r_1 = n^2$, $r_2 = -n$, $n^2 D_n = nD_n$, $D_n = c/n!$. The general solution to the homogeneous equation is $G_n = a + b\sum_{1\le i\le n} 1/i!$. $W_n = 1/(n-1)!$. $G_n = a + b\sum_{1\le i\le n} 1/i! - \sum_{2\le i\le n}[2^i \sum_{1\le j\le n-1} 1/j!]/[1/(i-3)!] - \sum_{1\le j\le n}(1/j!)\sum_{2\le i\le n}[2^i]/[1/(i-3)!]$
$= a + b\sum_{1\le i\le n} 1/i! - (1/n!)\sum_{2\le i\le n}[2^i]/[1/(i-3)!] = a + b\sum_{1\le i\le n} 1/i! - \sum_{2\le i\le n} 2^i(i-3)!/n!$.

6.2.6–4. See Bender and Orszag [21, p 50].

6.2.6–5. Letting $A = (-3 + \frac{4}{3}\sqrt{23})^{1/3}/4 \approx 0.376$ and $B = (-3 - \frac{4}{3}\sqrt{23})^{1/3}/4 \approx -0.528$, be roots of the characteristic equation are $\phi_1 = A + B - 1/4 \approx -0.402$, $\phi_2 = (A+B)/2 - 1/4 + (A-B)\sqrt{3}i/2 \approx -0.326 + 0.782i$, and $\phi_3 = (A+B)/2 - 1/4 - (A-B)\sqrt{3}i/2 \approx -0.326 - 0.782i$. A particular solution to the equation is given by a low degree polynomial, and it is most easily found by undetermined coefficients. Trying $an + b$ gives $(4+3+2+1)a = 1$ and $(4\cdot 3 + 3\cdot 2 + 2\cdot 1)b = 0$, so the general solution is $F_n = \phi_1^n + \phi_2^n + \phi_3^n + n/10$.

6.3–1. If $G_C(z) = G_A(z)G_B(z)$, then $G'_C(z) = G'_A(z)G_B(z) + G_A(z)G'_B(z)$, so $G'_C(1) = G'_A(1)G_B(1) + G_A(1)G'_B(1) = G'_A(1) + G'_B(1)$, and $A_C = A_A + A_B$. Likewise, $G''_C(z) = G''_A(z)G_B(z) + G'_A(z)G'_B(z) + G'_A(z)G'_B(z) + G_A(z)G''_B(z)$, so $G''_C(1) = G''_A(1)G_B(1) + G'_A(1)G'_B(1) + G'_A(1)G'_B(1) + G_A(1)G''_B(1) = G''_A(1) + 2G'_A(1)G'_B(1) + G'_A(1)G'_B(1) + G''_B(1)$. Now $V = G''(1) + G'(1) - G'(1)^2$, so $V_C = G''_C(1) + G'_C(1) - G'_C(1)^2 = G''_A(1) + 2G'_A(1)G'_B(1) + G''_B(1) + G'_A(1) + G'_B(1) - [G'_A(1)+G'_B(1)]^2 = G''_A(1)+G''_B(1)+G'_A(1)+G'_B(1)-G'_A(1)^2+G'_B(1)^2 = V_A + V_B$.

6.3–2. $C'(z) = [d\,A(B(z))/dB(z)][d\,B(z)/dz] = A'(z)B'(z)$, so $C'(1) = A'(1)B'(1)$ and $A_C = A_A A_B$. $C''(z) = d[d\,A(B(z))/dB(z)][d\,B(z)/dz]/dz = d[d\,A(B(z))/dB(z)]/dz\,[d\,B(z)/dz] + [d\,A(B(z))/dB(z)][d^2\,B(z)/dz^2] = [d^2\,A(B(z))/dB(z)^2][d\,B(z)/dz][d\,B(z)/dz] + [d\,A(B(z))/dB(z)][d^2\,B(z)/dz^2] = A''(B(z))B'(z)^2 + A'(B(z))B''(z)$, so $C''(1) = A''(1)B'(1)^2 + A'(1)B''(1)$, and $V_C = V_A A_B^2 + V_B A_A$.

6.3–3. The generating function for one roll is $(x + x^2 + x^3 + x^4 + x^5 + x^6)/6$. The generating function for n rolls is $[(x + x^2 + x^3 + x^4 + x^5 + x^6)/6]^n$. The average for one roll is $(1+2+3+4+5+6)/6 = 3\frac{1}{2}$ and the average for n rolls is $3\frac{1}{2}n$. The variance for one roll is $(1 + 4 + 9 + 16 + 25 + 36)/6 - 49/4 = 2\frac{11}{12}$ and the variance for n rolls is $2\frac{11}{12}n$.

6.3–4. The generating function for one roll is $(1+x+x^2+\cdots+x^m)/m$, which, in closed form, is $(1-x^{m+1})/(1-x)$. The generating function for n rolls is $[(1-x^{m+1})/(1-x)]^n$. The average for one roll is $(1+2+\cdots+m)/m = (m+1)/2$ and the average for n rolls is $n(m+1)/2$. The variance for one roll is $(1+4+\cdots+m^2)/m-(m+1)^2/4 = (m^2-1)/12$ and the variance for n rolls is $n(m^2-1)/12$.

6.3–5. $G(z) = (1 + z + z^2)/3$. The generating function for generation n can be obtained from $G_n(z) = [1 + G_{n-1}(z) + G_{n-1}(z)^2]/3$. (It is too complicated to write out directly.) The average and variance can be obtained by differentiating the recurrence. $G'_n(z) = [G'_{n-1}(z) + 2G_{n-1}(z)G'_{n-1}(z)]/3$ and $G''_n(z) = [G''_{n-1}(z) + 2G'_{n-1}(z)^2 + 2G_{n-1}(z)G''_{n-1}(z)]/3$, so $G'_n(1) = G'_{n-1}(1) = 1$ and $G''_n(1) = G''_{n-1}(1) + \frac{2}{3}G'_{n-1}(1)^2 = \frac{2}{3}n$. Therefore $A_n = 1$ and $V_n = \frac{2}{3}n$.

6.3–6. Set $z = 0$ in the recurrence for $G_n(z)$ obtain $G_n(0) = [1 + G_{n-1}(0) + G_{n-1}(0)^2]/3$. This recurrence does not appear to be easy to solve. The first few terms are $G_0(0) = 0$, $G_1(0) = 1/3$, $G_1(0) = 13/27$.

6.3–7. If there is a limiting probability, then it must be the case that for large n, $G_n(0)$ is nearly the same as $G_{n-1}(0)$, so the limiting probability can be obtained by solving the equation $G_\infty(0) = [1 + G_\infty(0) + G_\infty(0)]/3$, to obtain $G_\infty(0) = 1$. So the elephant is almost sure to eventually have no descendants.

7.1.1–1. You can solve the equation to obtain $T_n = 2^n$, but the solution has no relation to the full history problem [eq. (1)], because the solution to the full history equation agrees with that of first order equation only for $n \geq 1$.

7.1.1–2. The sum is zero for $n = 1$, so a suitable boundary condition is $t_1 = 1$. For $n \geq 2$, the recurrence is equivalent to $t_n = t_{n-1} + 1$. The solution is $t_n = n$.

7.1.1–3. The sum is zero for $n = 1$, so a suitable boundary condition is $t_1 = 1$. For $n \geq 2$, the recurrence is equivalent to $t_n = 3t_{n-1} + 1$. The solution is $t_n = (3^n - 1)/2$.

7.1.1.1–1. $p_{n,k} = \sum_{0 \leq i < n} p_i \sum_j p_{i,j} p_{n-i-1,k-j-n-1}$. The j index says how many comparisons are used in the first part (which has i elements). For the total number of comparisons to be k, the second part (which has $n - i - 1$) elements must use $k - j - n - 1$ comparisons because $n + 1$ comparisons are used for splitting.

7.1.1.2–1. $(n + 1)U_n - nU_{n-1} = 2 + U_{n-1}$ or $U_n = 2/(n + 1) + U_{n-1}$. The solution is $U_n = 2[1/(n + 1) + 1/n + \cdots + 1/2] + U_0 = 2H_{n+1} - 2$.

7.1.2.1–1. Consider two cases: (1) the root and (2) all other nodes. Case (1): A tree with no left subtree of the root consists of a root and a right subtree with $n - 1$ nodes. There are C_{n-1} such trees. Thus, there are $C_n - C_{n-1}$ trees where the root has a nonnull left subtree. Case (2): Suppose there are i nodes on the left subtree and $n - i - 1$ node on the right subtree. For such trees and a fixed right subtree, there are L_i nodes in the left subtrees that have nonnull subtrees. For each left subtree there are C_{n-i-1} right subtrees. For right subtrees we have L_{n-i-1} nodes with nonnull left subtrees (for each left subtree of the root) and a factor of C_i to account for the number of left subtrees of the root. Totaling the cases gives $L_n = C_n - C_{n-1} + \sum_i L_i C_{n-i-1} + \sum_i L_{n-i-1} C_i$ which reduces to the required answer after a change of variable on the last summation.

7.1.2.1–2. Multiplying the recurrence by x^n, summing, and using $L_0 = 0$ gives $L(x) = \sum_{n \geq 1} \left(\sum_{0 \leq i \leq n-1} 2L_i C_{n-i-1} + C_n - C_{n-1} \right) x^n =$
$x\left(\sum_{n \geq 0} L_n x^n \right) \left(\sum_{n \geq 0} C_n x^n \right) + \sum_{n \geq 0} C_n x^n - 1 - x \sum_{n \geq 0} C_n x^n = 2xL(x)C(x) +$
$C(x) - 1 - xC(x)$. Solving for $L(x)$ gives the final result.

7.1.2.1–3. $L(x) = 1/(2x\sqrt{1 - 4x}) - 3/(2\sqrt{1 - 4x})$. Expanding the square roots with the binomial theorem gives (after much algebra) $L(x) = \sum_{n \geq 0} (n - 1)\binom{2n}{n}/[2(n + 1)]x^n$, $L_n = (n - 1)\binom{2n}{n}/[2(n + 1)] = (n - 1)C_n/2$.

7.1.3–1. $f(x, z) = e^{xz - x^2/2}$.

7.1.3–2. $z/(e^z - 1)$. See Knuth [9, Section 1.2.11.2].

7.1.3.1–1. $A(t, v) = atv + 2\sum_i \binom{t}{i} p^i (1 - p)^{t-i} A(t - i, v - 1)$. Eq. (94) becomes $G_v(z) =$
$a \sum_{0 \leq i < v} (v - i) 2^i (1 - p)^i z e^z = aze^z \{[2(1-p)+v][2(1-p)]^v - (v+1)[2(1-p)]\}/(1 -$
$2p)^2$, so $A(t, v) = at\{[2(1-p)+v][2(1-p)]^v - (v+1)[2(1-p)]\}/(1-2p)^2$. When v is large and $2p$ is much less than 1, this is about v times larger than the previous case.

7.1.3.1–2. Transform the recurrence to $\lg T_n = \lg T_{n-1} + \lg T_{n-2}$ with boundary conditions $\lg T_1 = 0$ and $\lg T_2 = 1$. The solution is $\lg T_n = F_{n-1}$, so $T_n = 2^{F_{n-1}}$, where F_n is the n^{th} Fibonacci number.

7.1.3.1–3. Letting $F_n = T_n/T_{n-1}$, $F_n = nF_{n-1}$, so $F_n = an!$, and $T_n = an!T_{n-1} = a^n n!(n-1)!(n-2)!\cdots 1$.

7.1.3.1–4. Let $T_n = \frac{1}{2}(1 - S_n)$. $T_n = [1 - (1 - 2T_1)^{2^{n-1}}]/2$. See Bender and Orszag [21, p 53].

7.1.4–1. $C_n = \frac{12}{7}[(n+1)H_n + 1] + c_1(n+1) + c_2(-1)^n\binom{5}{n}$. See Green and Knuth [6, Section 2.1.2.2].

7.1.4–2. See Purdom and Brown, SIAM J. Comp. **14**(1985) pp 943–953.

7.2.1–1. g_n satisfies the equation $g_{n+1}g_n + (a+A)g_{n+1} + (b+A)g_n = 0$ and h_n satisfies the equation $(b+A)h_{n+1} + (a+A)h_n + 1 = 0$, so $h_n = [-(a+A)/(b+A)]^n C$, $g_n = [-(b+A)/(a+A)]^n C'$, $f_n = [-(b+A)/(a+A)]^n C' + A$.

7.2.1–2. Let $L_n = \lg T_n$. $\lg T_n = \lg T_{n-1} + \lg T_{n-2}$, so $L_n - L_{n-1} - L_{n-2} = 0$, with $L_0 = 0$ and $L_1 = 1$. Therefore $L_n = F_n$, the n^{th} Fibonacci number, and $T_n = 2^{F_n}$.

7.2.1–3. Let $L_n = \ln T_n$. $\ln T_n - 2\ln T_{n-1} + \ln T_{n-2} = \ln n$, or $L_n - 2L_{n-1} + L_{n-2} = \ln n$. The solution to the homogenous equation is $L_n = a + bn$. Variation of parameters gives $L_n = a + bn - \sum_{2\le i\le n}[(i-1)\ln i] + n\sum_{2\le i\le n}(\ln i)$. The last sum is $nn!$. The general solution is $T_n = \exp\left(a + bn + (n-1)n! - \sum_{1\le i\le n} i\ln i\right)$.

7.2.1–4. $aS_{n+1} + b = 2(aS_n + b)(1 - aS_n - b)$, or $aS_{n+1} + 2a^2S_n^2 + 2a(2b-1) + b(2b-1) = 0$. For $a = -1/2$ and $b = 1/2$, this reduces to $S_{n+1} = S_n^2$. The solution to this equation is $S_n = S_0^{(2^n)}$, so the solution to the original problem is $T_n = (1 - S_0^{2^n})/2$, or $T_n = [1 - (1 - 2T_0)^{(2^n)}]/2$.

7.2.1.1–1. $\int_n^\infty 2^{-x}\ln x\,dx < [(\ln n)/n]\int_n^\infty 2^{-x}x\,dx$
$= [(\ln n)/n]\left[n/\ln 2 + 1/(\ln 2)^2\right]e^{-n\ln 2} < 2\lg n 2^{-n}$, so $P_n = O(nK^{2^n})$.

7.2.2–1. A tree of height n consists of either two subtrees of height $n-1$, a left subtree of height $n-1$ and a right subtree of height $n-2$, or a left subtree of height $n-2$ and a right subtree of height $n-1$, which gives the recurrence. There is one such tree of height zero (consisting of just a root) and three of height one [the three are (1) the complete binary tree with three nodes, (2) the tree with a root and a leaf on the left, and (3) the tree with a root and a leaf on the right].

7.2.2–2. Let $X_n = T_n + T_{n-1}$. $T_n + T_{n-1} = (T_{n-1} + T_{n-2})^2 + T_{n-1} - T_{n-2}^2$, so $X_n = X_{n-1}^2 + T_{n-1} - T_{n-2}^2 = X_{n-1}^2 + 2T_{n-2}T_{n-3}$. This leads to $X_n = \lfloor\theta^{2^n}\rfloor$ and $T_n = \lfloor\theta^{2^n}\rfloor - \lfloor\theta^{2^{n-1}}\rfloor + \cdots \pm 1$, where $\theta \approx 1.43684$. See Knuth [11, Section 6.2.3] for additional details.

7.2.2–3. Write the recurrence as $T_0T_1\cdots T_{n-2} = T_{n-1} - r$, and replace $T_0T_1\cdots T_{n-2}$ with $T_{n-1} - r$ in the original recurrence to obtain $T_n = (T_{n-1} - r)T_{n-1} + r$. Let $T_n = X_n + \frac{1}{2}r$. Then $X_n + \frac{1}{2}r = (X_{n-1} - \frac{1}{2}r)(X_{n-1} + \frac{1}{2}r) + r$, so $T_n \approx \frac{1}{2}r\theta^{2^n}$ for the appropriate value of θ (which depends on r). For $r = 2$, $\theta = \sqrt{2}$, and for $r = 4$, $\theta = \frac{1}{2}(1 + \sqrt{5})$. See Greene and Knuth [6, pp 33-34] for more details.

7.2.3–1. The "main" operator has one or more items before it and one or more after. Thus, it can be at position i, where $1 \le i \le n-1$. If the main operator is a position i, then the items before it can be associated W_i ways and the items after it can be associated W_{n-i} ways. There is just one way to associate one item. Letting

$W_i = Y_{i-1}$ gives $Y_{n-1} = \sum_{1\le i\le n-1} Y_{i-1}Y_{n-i-1}$. Replacing n by $n+1$ and i by $i+1$ gives $Y_n = \sum_{0\le i\le n-1} Y_iY_{n-i-1}$, with $Y_0 = 1$, so $Y_n = C_n$, the n^{th} Catalan number, and $W_n = C_{n-1}$.

7.2.3–2. Number the vertices consecutively. Any two adjacent vertices are members of some triangle. Consider the triangle formed by vertices 1, 2, and some third vertex (say i). The line from 2 to i divides the polygon into two parts. The part that is opposite vertex 1 has $i-1$ vertices. Likewise, the line from 1 to i divides the polygon into two parts, and the part opposite vertex 2 has $n-i+1$ vertices. The triangle 1, 2, i, the part with $i-1$ vertices, and the part with $n-i+2$ vertices just cover the polygon. There are T_{i-1} ways to triangulate the $i-1$ part and T_{n-i+2} ways to triangulate the $n-i+2$ part. Thus, $T_n = \sum_{3\le i\le n} T_{i-1}T_{n-i+2} = \sum_{2\le i\le n-1} T_iT_{n-i+1}$. When $n = 2$ or 3 there is just one way to triangulate the polygon (do nothing).

7.2.3–3. Define $S_{n-2} = T_n$. Then $S_{n-2} = \sum_{2\le i\le n-1} S_{i-2}S_{n-i-1} = \sum_{0\le i\le n-3} S_iS_{n-i+1}$. Replacing n with $n+2$ gives $S_n == \sum_{0\le i\le n-1} S_iS_{n-i-1}$, with $S_0 = 1$. Therefore $S_n = C_n$ (the n^{th} Catalan number), and $T_n = C_{n-2}$.

7.2.3–4. $\phi_0 = 0$ because with zero flips the coin cannot land heads up. $\phi_1 = p$ because with one flip, if the coin lands heads up, then it has for the first time landed heads up once more than tails up. Now consider general $n \ge 2$. If the first flip is heads, then the case does not contribute to ϕ_n. If the first flip is tails then you must first get back to even and then get ahead by one. The probability of getting a tail, then catching up in i flips and then getting ahead by one for the first time at the n^{th} flip is $(1-p)\phi_i\phi_{n-i-1}$ since these are independent events. The contribution for each value of i is disjoint, so the recurrence is obtained by summing over i.

7.2.3–5. Let $G(x) = \sum_{n\ge0} \phi_n x^n$. Then $\sum_{n\ge0} \phi_n x^n = (1-p)x\big(\sum_{i\ge0} \phi_i x^i\big)\big(\sum_{j\ge0} \phi_j x^j\big) + px$, so $G(x) = [1-\sqrt{1-4p(1-p)x^2}]/[2(1-p)x]$. (Using the plus sign in front of the square root would lead to divergence for x near zero.) The average is given by $G(1)' = 3p/[2(1-p)]$.

7.2.3–6. For even n, $\phi_n = 0$; for odd n, $\phi_n = 2p^{(n+1)/2}(1-p)^{(n-1)/2}\binom{n-1}{(n-1)/2}/(n+1)$. This can be written with Catalan numbers as $\phi_n = p^{(n+1)/2}(1-p)^{(n-1)/2}C_{(n-1)/2}$.

7.3–1. If n is even, then $\lfloor n/2\rfloor = n/2$, and n and $n/2$ have the same number of one bits. Therefore if $D_{n/2} = n - d_{n/2}$, then by eq. (178) $D_n = n+n-d_{n/2} = 2n-d_n$. If n is odd, then $\lfloor n/2\rfloor = (n-1)/2$, and n has one more one bit than $\lfloor n/2\rfloor$. Therefore if $D_{(n-1)/2} = n-1-d_{(n-1)/2}$, then by eq. (178) $D_n = n+(n-1)-d_{(n-1)/2} = 2n-d_n$. By definition $D_0 = 0$, so $D_0 = 2n-d_n$ for $n = 0$. Putting the pieces together, we have $D_0 = 2n-d_n$, and we have $D_n = 2n-d_n$ implies $D_{2n} = 2(2n)-d_{2n}$ and $D_{2n+1} = 2(2n+1)-d_{2n+1}$, so by induction the result is true.

7.3–2. When n is even, eq. (181) is $T_n = 1+T_{n/2}$. For $n = 1$, eq. (183) gives $T_1 = 3-2 = 1$, which is the correct value. Using in eq. (183) on the right (for $n \ge 2$) gives $1+\lfloor\lg n\rfloor - 1+3-2^{\lfloor\lg n\rfloor}/(n/2) = \lfloor\lg n\rfloor+3-2^{\lfloor\lg n\rfloor+1}/n$, so the formula works for even n (provided we can also show that it works for odd n). For odd n where n is not one less than a power of 2, $T_{\lfloor n/2\rfloor}$ and $T_{\lceil n/2\rceil}$ have almost the same value, so the right side of eq. (181) is $1+\lfloor\lg n\rfloor - 1+3-2^{\lfloor\lg n\rfloor}\{[(n+1)/2]/[(n+$

$1)/2] + [(n-1)/2]/[(n-1)/2]\}/n = \lfloor \lg n \rfloor + 3 - 2^{\lfloor \lg n \rfloor + 1}/n$, so the works for odd values of n that are not one less than powers of 2. When n is odd and one less than a power of 2, we can write n as $2^{k+1} - 1$, and the right side of eq. (181) is $1 + [2^k/(2^{k+1}-1)](k+1) + [(2^k-1)/(2^{k+1}-1)][k+1-2/(2^k-1)] = 1 + \{[(k+1)2^{k+1} - k - 1]/(2^{k+1}-1)\} - 2/(2^{k+1}-1) = k + 2 - 2/(2^{k+1}-1)$, which is the same as the value for $T_{2^{k+1}-1}$ given by eq. (183).

7.3–3. $T_n = n\lceil \lg n \rceil - 2^{\lceil \lg n \rceil} + 1$. Notice that this problem is similar to the recurrence in eq. (181).

7.3–4. The answer varies about like $n^{\lg 3}$. Number evaluation shows that in the range $512 \le n \le 1024$, $3.44n^{\lg 3} \le T_n \le 3.73n^{\lg 3}$. The smallest coefficient occurs at $n = 549$ where $T_n = 75738$ and the largest occurs at $n = 880$ where $T_n = 172856$. At $n = 512$, $T_n = 69500$ and at $n = 1024$, $T_n = 209537$.

7.3–5. $n!$ is the product of the numbers from 1 to n. Every other one is evenly divisible by 2, so there are $\lfloor n/2 \rfloor$ of the factors that are evenly divisible by 2. Every fourth factor is also divisible by 4, and every 2^i factor is evenly divisible by 2^i. This gives the summation. The recurrence can be obtained by noticing that $\lfloor n/2 \rfloor$ of the factors are divisible by 2, and that after a factor of 2 is divided out of each even factor, the even numbers are transformed into the integers from 1 to $\lfloor n/2 \rfloor$, $T_{\lfloor n/2 \rfloor}$ gives the number of additional factors of 2 that can be divided out. The summation would have a solution of $2n$ if it was not for the floor function. The floor function causes the answer to be a little smaller. If you think of dividing by two as shifting the binary representation of the number to the right one place, then you can see that the difference between $n/2^i$ and $\lfloor n/2^i \rfloor$ is that in the latter case the bits shifted below the binary point are lost. This does not manner when the bits are zero, but it does when they are one. The total effect of the "lost" bits in the summation is d_n, the number of one bits in n.

7.4–1. Use proof by induction to show that $f_k(x) = x\ln(x/k)$. Base case: $f_1(x) = \min_{0 \le y \le x}\{y \ln y\} = x \ln x$ because $y \ln y$ is an increasing function of y. Assume $f_k(x) = x\ln(x/k)$ for $k < k_*$. Then $f_{k_*}(x) = \min_{0 \le y \le x}\{y \ln y + f_{k_*-1}(x-y)\} = \min_{0 \le y \le x}\{y \ln y + (x-y)\ln[(x-y)/(k_*-1)]\}$. At the minimum, the derivative with respect to y is zero, so $(d/dy)\{y \ln y + (x-y)\ln[(x-y)/(k_*-1)]\} = \ln y + 1 - \ln[(x-y)/(k_*-1)] - 1 = 0$. This gives $y = (x-y)/(k_*-1)$ or $y = x/k_*$ at the minimum. Plugging this value in gives $f_{k_*}(x) = (x/k_*)\ln(x/k_*) + [(k_*-1)x/k_*]\ln(x/k_*) = x\ln(x/k_*)$. Therefore if the result is true for all $k < k_*$ then it is also true for $k = k_*$.

7.4–2. The minimum is at $x_i = 1/n$. The value of the minimum is $1/n$.

7.4.1–1. A complete restricted three way tree has $3 \cdot 2^{i-1}$ nodes on level i for $i \ge 1$. One third of these are terminal nodes. A tree that is as close to balanced as possible with $n = 2^{k+1} - 1$ terminal nodes has $3 \cdot 2^{k-1}$ of its terminal nodes on the bottom level. A tree with $n = 2^{k+1} - 1 + 1$ terminal nodes has $3 \cdot 2^{k-1} - 1$ terminal nodes on the next to the bottom level (which was the bottom level when there was one less node) and 2 nodes on the bottom level. A tree with $n = 2^{k+1} - 1 + 2$ terminal nodes has $3 \cdot 2^{k-1} - 1$ terminal nodes one the next to bottom level and 3 nodes on the bottom level. Then the pattern of increases repeats. A tree with $n = 2^{k+1} - 1 + 3$ terminal nodes has $3 \cdot 2^{k-1} - 2$ terminal nodes on the next to the bottom level and 5 nodes on the bottom level. A

tree with $n = 2^{k+1} - 1 + j$ terminal nodes has $3 \cdot 2^{k-1} - \lceil j/2 \rceil$ terminal nodes on the next to bottom level and $\lceil 3j/2 \rceil$ nodes on the bottom level. The path length for such a tree is $\sum_{0 \le i < k} i2^{i-1} + (3 \cdot 2^{k-1} - \lceil j/2 \rceil)k + \lceil 3j/2 \rceil (k+1) = k2^{k+1} - 2^k + jk + \lceil 3j/2 \rceil + 1$ Since $k = \lfloor \lg(n+1) \rfloor - 1$ and $j = n - 2^{k+1} + 1$, $C_n = k2^{k+1} - 2^k + jk + \lceil 3j/2 \rceil + 1 = \lfloor \lg(n+1) \rfloor 2^{\lfloor \lg(n+1) \rfloor + 1} - 2^{\lfloor \lg(n+1) \rfloor + 2} + (n - 2^{\lfloor \lg(n+1) \rfloor})(\lfloor \lg(n+1) \rfloor - 1) + \lceil 3(n - 2^{\lfloor \lg(n+1) \rfloor})/2 \rceil + 1$.

7.4.2–1. See Reingold and Tarjan [127].

7.4.3.1–1.

$$
\begin{aligned}
Y_{00} &= 0 \\
Y_{i0} &= Y_{i-1,0} + D_{a_i} \quad \text{for } i \ge 1, \\
Y_{0j} &= Y_{0,j-1} + I_{b_j} \quad \text{for } j \ge 1, \\
Y_{ij} &= min\{Y_{i-1,j-1} + C_{a_i,b_j}, Y_{i-1,j} + D_{a_i}, Y_{i,j-1} + I_{b_j}\} \quad \text{for } i, j \ge 1.
\end{aligned}
$$

These can be solved in time $O(mn)$ by solving in order of increasing i and for each value of i in order of increasing j.

7.4.3.1–2.

Algorithm 7a.1 **Edit:** Input: Cost matrices I, D, C; strings $a_1 \ldots a_m$ and $b_1 \ldots b_n$. Output: Matrices Y and A where Y_{ij} gives the cost of converting $a_1 \ldots a_i$ to $b_1 \ldots b_j$ and A_{ij} has the action for last step of the transformation of the first string into the second string. The actions are: *delete* i (delete the character that was originally in position i from the first string), *insert* s_j at i (insert the symbol s_j at the end of the characters that have so far been added at the original position i), and change i to s_j (change the character that was originally at position i to s_j). No entry for A_{ij} indicates that no change is needed. From this information, you can start with A_{mn} and construct the entire transformation step by step.

Step **1.** $Y_{00} \leftarrow 0$.

Step **2.** For $i \leftarrow 1$ to m do $Y_{i0} \leftarrow Y_{i-1,0} + D_{a_i}$ and $A_{i0} \leftarrow$ *delete* i.

Step **3.** For $j \leftarrow 1$ to n do $Y_{0j} \leftarrow Y_{0,j-1} + I_{b_j}$ and $A_{0j} \leftarrow$ *insert* s_j at 0.

Step **4.** For $i \leftarrow 1$ to m do

Step **5.** For $j \leftarrow 1$ to n do

Step **6.** $Y_{ij} \leftarrow Y_{i-1,j-1} + C_{a_i,b_j}$ and $A_{i,j} \leftarrow$ *change* i to s_j.

Step **7.** If $Y_{ij} > Y_{i-1,j} + D_{a_i}$ then $Y_{ij} \leftarrow Y_{i-1,j} + D_{a_i}$ and $A_{ij} \leftarrow$ *delete* i.

Step **8.** If $Y_{ij} > Y_{i,j-1} + I_{b_j}$ then $Y_{ij} > Y_{i,j-1} + I_{b_j}$ and $A_{ij} \leftarrow$ *insert* s_j at i.

Step **9.** End for.

Step **10.** End for.

7.4.3.1–3. The time is given by the recurrence $T_{i,j} = t_1 + (j-i)t_2 + \sum_{i \le k < j}(T_{i,k} + T_{k+1,j})$ with the boundary condition $T_{i,i} = t_3$. Since the solution just depends on the difference between i and j, we can write $S_{j-i} = T_{i,j}$ and obtain the recurrence $S_i = t_1 + it_2 + 2\sum_{0 \le j < i} S_j$ with the boundary condition $S_0 = t_3$. Elimination of history gives $S_{i+1} - S_i = t_2 + 2S_i$, or $S_i = t_2 + 3S_{i-1} = t_2(3^{i-1} - 1)/2 + 3^{i-1}S_1$. Since $S_1 = t_1 + 2t_0$, $T_{1,n} = S_{n-1} = t_2(3^{n-2} - 1)/2 + (t_1 + 2t_0)3^{n-2}$. The algorithm is so slow ($O(3^n)$) because it does not remember the answers to subproblems and must frequently recompute them.

7.4.3.1–4. This algorithm does each subproblem only once. The time $T_{i,j}$ equals t_0 for $i = j$ and $t_1 + (j-i)t_2$ for $i < j$ plus the time for subproblems. This leads to $T_{1,n} = \sum 1 \le i \le j \le n S_{i,j}$, where $S_{i,j} = t_1 + (j-i)t_2$ for $i < j$ and $S_{i,i} = t_0$. Let $R_{j-i} = S_{i,j}$. Then $T_{1,n} = \sum 1 \le i \le j \le n R_{j-i} = \sum 1 \le i \le n \sum_{i \le j \le n} R_{j-i} = \sum 1 \le i \le n \sum_{0 \le j \le n-i} R_j = \sum_{0 \le j \le n-1} \sum 1 \le i \le n - j R_j = \sum_{0 \le j \le n-1}(n - j)R_j = nt_0 + t_1[n(n+1)/2] + t_2[(n^3 - n)/6] = O(n^3)$. Fast is so much faster because it does not have to rework subproblems.

7.4.3.1–5. Fast must work all the subproblems for $1 \le i \le j \le n$, but Faster may skip some of them due to the if test in step 6. It is difficult to analyze because it is difficult to tell how many subproblems will be skipped. The time for Fast does not depend on the input, but the time for Faster does. Furthermore the time for Faster depends on the data in a rather complicated way.

7.4.3.2–1. Let $C_{i,j}$ be the number of comparisons needed to find items i through j when they form a minimum cost search tree. Then $C_{i,i} = 0$, because you can find the item with no searches ($C_{i,i} = 1$ if you need to verify that the item is in the tree). If a tree is built for nodes i through j with node k at the root and with minimum cost search trees for the left and right subtrees, then $C_{i,j} = p_i + p_{i+1} + \cdots + p_j + C_{i,k-1} + C_{k,j}$. The minimum cost subtree has for the root that node k which minimizes this expression. These equations can be solved in time $O(n^3)$ by first computing $C_{i,j}$ for $j = i$, $1 \le i \le n$, then for $j = i + 1$, $1 \le i \le n - 1$, etc.

8.1.1–1. $T_n = T_{n-1} + S_{n-1}$, $S_n = T_{n-1}$. The solution is $T_n = c_1\lambda_1^n + c_2\lambda_2^n$, $S_n = c_1\lambda_1^{n-1} + c_2\lambda_2^{n-1}$, where $\lambda_1 = (-1 + \sqrt{3}i)/2$ and $\lambda_2 = (-1 - \sqrt{3}i)/2$ and i is the square root of -1.

8.1.1–2. The general solution of $T_n = 2T_{n-1}$ is $T_n = 2^n T_0$. The general solution of $T_n = 3T_{n-1} - 2T_{n-2}$ is $T_n = c_1 + c_2 2^n$. The general solution of the original recurrence can be obtained from the general solution of the new recurrence by setting the parameters in the new solution appropriately, i.e., $c_1 = 0$ and $c_2 = T_0$.

8.1.1–3. From eq. (11) we get that $T_n = c_1 + c_2(-1)^n + c_3 2^n + c_4(-2)^n$. From eq. (8) we get that $S_n - S_{n-1} = 4c_2(-1)^n + \frac{3}{2}c_3 2^n + \frac{1}{2}c_4(-2)^n$, so $S_n = c_5 + 2c_4(-1)^n + 3c_3 2^n + \frac{1}{3}c_4(-2)^n$. We have 5 parameters, which is one too many. Fitting these answers to eq. (6) with $n = 2$ gives $c_2 + c_3 + 3c_4 = 0$, or $c_2 = -c_3 - 3c_4$. Plugging this in gives $T_n = c_1 - (c_3 + 3c_4)(-1)^n + c_3 2^n + c_4(-2)^n$, $S_n = c_5 + 2c_4(-1)^n + 3c_3 2^n + \frac{1}{3}c_4(-2)^n$. This problem can also be solved by a simple extension of the method of Section 8.1.3., which requires solving the equation

$$\det \begin{vmatrix} 1 - \lambda^{-1} & -\lambda^{-1} + \lambda^{-2} \\ -2\lambda^{-1} + 2\lambda^{-2} & 1 - \lambda^{-1} \end{vmatrix} = 0.$$

8.1.1.1–1. The big O term in eq. (34) is largest for small t, since $t \geq 0$, eq. (34) implies that $t > \ln M_t / \ln[(3+\sqrt{5})/2] + O(1)$. Plugging this lower bound on t into the right side of eq. (34) gives

$$t = \frac{\ln M_t}{\ln \frac{3+\sqrt{5}}{2}} - \frac{\ln \frac{5+\sqrt{5}}{10}}{\ln \frac{3+\sqrt{5}}{2}} + O\left(\left(\frac{3-\sqrt{5}}{3+\sqrt{5}}\right)^{\ln M_t / \ln[(3+\sqrt{5})/2]+O(1)}\right)$$
$$= \frac{\ln M_t}{\ln \frac{3+\sqrt{5}}{2}} - \frac{\ln \frac{5+\sqrt{5}}{10}}{\ln \frac{3+\sqrt{5}}{2}} + O\left(M_t^{\ln[(3-\sqrt{5})/(3+\sqrt{5})]/\ln[(3+\sqrt{5})/2]}\right).$$

8.1.1.1–2. M_t is given by $M_{t+2} - 5M_{t+1} + M_t = 0$ with boundary conditions $M_0 = 1$, $M_1 = 4$. $M_t = [(21+3\sqrt{21})/42][(5+\sqrt{21})/2]^t + [(21-3\sqrt{21})/42][(5-\sqrt{21})/2]^t$, $P_t = (\sqrt{21}/21)[(5+\sqrt{21})/2]^t - (\sqrt{21}/21)[(5-\sqrt{21})/2]^t$.

8.1.3.1–1. Adding $1/\lambda$ times the last row to the next to the last row gives a next to last row with $1 + 1/\lambda$ in the first column, $-\lambda$ in the next to last column, and zero in the other columns. Continuing the process of subtracting rows from the one above gives an i^{th} from the bottom row that has $1 + 1/\lambda + \cdots + 1/\lambda^i$ in the first column, $-\lambda$ in the i from the last column and zero in all the other columns. The first column has λ less than this in the first column because it started with $1 - \lambda$ rather than with 1, so the first row of the first column becomes $-\lambda + 1 + 1/\lambda + \cdots + 1/\lambda^{n-1}$. The resulting matrix has no nonzero entries above the main diagonal, so the value of the determinant is the product of the entries on the diagonal, $(-1)^n(\lambda^n - \lambda^{n-1} - \lambda^{n-2} - \cdots - 1)$. Setting this to zero and dividing by $(-1)^n$ gives the final answer.

8.1.3.1–2. Eq. (106) is the significant equation. From Table 8.1, we get that the algorithm uses time $24 \cdot 3 + 13 \cdot 5 + 7 \cdot 9 + 4 \cdot 17 + 2 \cdot 31 + 1 \cdot 57 + 1 \cdot 105 = 492$ time units, where a time unit is the time to process a block of length one. Eq. (106) says that this time is approximately $0.590N \lg N \approx 415.9$ in the same time units, so in this case the formula is too low by about 15 percent.

8.1.3.1–3. See Knuth [11, Section 5.4.2].

8.1.4–1. If $0 < k < n$ then, there will be k running at time t if (1) there were k at time t and nothing happened: $(1 - \lambda\,dt)^k(1 - \mu dt)P_k(t) + o(dt) = P_k(t) - k\lambda P_k(t)\,dt - \mu P_k(t)\,dt + o(dt)$, (2) there were $k-1$ at t and one was fixed: $(1 - \lambda\,dt)^k(\mu dt)P_k(t) + o(dt) = \mu P_k(t)\,dt + o(dt)$, (3) there were $k+1$ at t and one broke: $(1-\lambda\,dt)^k(k+1)\lambda\,dt(1-\mu dt)P_k(t)+o(dt) = (k+1)\lambda P_{k+1}(t)\,dt+o(dt)$, (4) other cases: $o(dt)$. Adding the cases gives $P_k(t+dt) = P_k(t) + [\mu P_{k-1}(t) - (k\lambda - \mu)P_k(t) + (k+1)\lambda P_{k+1}(t)]\,dt + o(dt)$. Subtracting $P_k(t)$, dividing by dt and taking the limit as dt goes to zero gives

$$\frac{d\,P_k(t)}{dt} = \mu P_{k-1}(t) - (k\lambda - \mu)P_k(t) + (k+1)\lambda P_{k+1}(t).$$

For $k = 0$ case (1) does not apply, so we get

$$\frac{dP_0(t)}{dt} = -\mu P_0(t) + \lambda P_1(t).$$

For $k = n$ case (3) does not apply, so we get

$$\frac{dP_n(t)}{dt} = \mu P_{n-1}(t) - n\lambda P_n(t).$$

8.1.4–2. The equations for the steady state solution are

$$\begin{aligned} -\mu P_0 + \lambda P_1 &= 0 \\ \mu P_{k-1} - (k\lambda + \mu)P_k + (k+1)\lambda P_{k+1} &= 0 \\ \mu P_{n-1} - k\lambda P_k &= 0. \end{aligned}$$

The solutions are $P_k = \rho^k/[k!(\sum_{0\le i\le n}\rho^i/i!)]$, where $\rho = \mu/\lambda$.

8.1.4–3. See Fisz [28, Section 8.5.C].

8.1.4–4. See Fisz [28, Section 8.5.C].

8.2.1–1. Consider a set with k elements that are selected from the integers from 1 to n and that satisfy the restriction. If the element n is in the set, then $n-1$ is not and the remaining $k-1$ elements are from the set of integers from 1 to $n-1$ and they obey the restriction. If n is not in the set, then there are k numbers in the range 1 to $n-1$ and they obey the restriction. These two cases include all the possibilities, each one counted once.

8.2.1–2. $f(n,k) = \binom{n-k+1}{k}$.

8.2.1–3. Replacing n with $n-1$ gives $F_{n-1,i} = (n-1)F_{n-2,i} + F_{n-2,i-1}$. Letting $G_{ni} = F_{n-1,i}$ gives $G_{ni} = (n-1)G_{n-1,i} + G_{n-1,i-1}$, so $G_{ni} = \sum_i a_i \left[{n \atop i}\right]$, or $F_{ni} = \sum_i a_i \left[{n+1 \atop i}\right]$, is a general solution.

8.2.1–4. $F_{ni} = \left[{n+1 \atop i+1}\right]$.

8.2.2–1. The indices for each term on the right side are lexicographically less than the indices for the term on the left side, so this can be proved by induction. Assuming the proposed solution works for the terms on the right side, you can show that the left side is given by the proposed solution. Plugging the proposed solution into the right side and factoring out common factors gives $(i_1 + i_2 + \cdots + i_n - 1)!/(i_1!i_2!\ldots i_n!)(i_1 + i_2 + \cdots + c_n) = \binom{i_1+i_2+\cdots+c_n}{i_1,i_2,\ldots,i_n}$, which is what $F_{i_1,i_2,\ldots,i_n}$ was claimed to equal.

8.2.2.1–1. See Knuth [11, p. 427].

8.2.2.1–2. Let $n' = n$, $i' = n - i$ to obtain $F_{n,n-i} = g(i)F_{n-1,n-i} + F_{n-1,i-1}$. Let $F_{n,n-i} = G_{n,i}$ to obtain $G_{n,i} = G_{n-1,i} + g(i)G_{n-1,i-1}$. Let $h(i)H_{ni} = G_{ni}$. For $h(i) = \prod_{1\le k\le i} g(k)$, this gives $H_{n,i} = H_{n-1,i} + H_{n-1,i-1}$, so $H_{ni} = \sum_j a_j \binom{n}{i+j}$, $G_{ni} = \sum_j a_j \binom{n}{i+j} \prod_{1\le k\le i} g(k)$, and
$F_{ni} = \sum_j a_j \binom{n}{n-i+j} \prod_{1\le k\le n-i} g(k) = \sum_j b_j \binom{n}{i+j} \prod_{1\le k\le n-i} g(k)$, where the a's and b's are arbitrary constants.

8.2.2.1–3. First work Exercise 4 or 5. Then use the definitions to show that $\binom{n}{0}_q = 1$ and $\binom{0}{k}_q = \delta_{k0}$. Finally do a proof by induction. From the recurrence, you have that if $\binom{n-1}{k}_q$ and $\binom{n-1}{k-1}_q$ are polynomials, then so is $\binom{n}{k}$.

8.2.2.1–4. $[(q^n-1)/(q-1)][(q^{n-1}-1)/(q^2-1)]\ldots[(q^{n-i+1}-1)/(q^i-1)]-[(q^{n-1}-1)/(q-1)][(q^{n-2}-1)/(q^2-1)]\ldots[(q^{n-i}-1)/(q^i-1)]=[q^n-1-(q^{n-i}-1)][(q^{n-1}-1)/(q-1)][(q^{n-2}-1)/(q^2-1)]\ldots[(q^{n-i+1}-1)/(q^{i-1}-1)]=q^{n-i}[(q^{n-1}-1)/(q-1)][(q^{n-2}-1)/(q^2-1)]\ldots[(q^{n-i}-1)/(q^i-1)]$.

8.2.2.1–5. $[(q^n-1)/(q-1)][(q^{n-1}-1)/(q^2-1)]\ldots[(q^{n-i+1}-1)/(q^i-1)]-[(q^{n-1}-1)/(q-1)][(q^{n-2}-1)/(q^2-1)]\ldots[(q^{n-i+1}-1)/(q^{i-1}-1)]=[(q^n-1)/(q^i-1)-1][(q^{n-1}-1)/(q-1)][(q^{n-2}-1)/(q^2-1)]\ldots[(q^{n-i+1}-1)/(q^{i-1}-1)]=q^i[(q^{n-1}-1)/(q-1)][(q^{n-2}-1)/(q^2-1)]\ldots[(q^{n-i}-1)/(q^i-1)]$.

8.2.2.1–6. Proof by induction. Base case $n=0$: $1=\binom{0}{0}_q q^0x^0$. If it is true up to $n-1$, then the left side is
$(1+q^{n-1}x)\sum_{0\le i\le n-1}\binom{n-1}{i}_q q^{i(i-1)/2}x^i = 1+q^{n-1+(n-1)(n-2)/2}x^n$
$+\sum_{0\le i\le n-1}\left[\binom{n-1}{i}_q q^{i(i-1)/2}-\binom{n-1}{i}_q q^{n-1+(i-1)(i-2)/2}\right]$
$=1+q^{n(n-1)/2}x^n+\sum_{0\le i\le n-1}\left[\binom{n-1}{i}_q-\binom{n-1}{i}_q q^{i(i-1)/2}\right]q^{i(i-1)/2}x^i=$
$\sum_{0\le i\le n}\binom{n}{i}_q q^{i(i-1)/2}x^i$, so it is also true up to n.

8.2.2.1–7. Notice that $(q^n-1)/(q-1)=q^{n-1}+q^{n-2}+\cdots+1==[1+(q-1)]^{n-1}+[1+(q-1)]^{n-2}+\cdots+1=n+O(q-1)$. Dividing top and bottom by $(q-1)^k$ leads to $\binom{n}{i}_q=[n(n-1)\cdots(n-i+1)+O(q-1)]/[1\cdot2\cdots k+O(q-1)]=\binom{n}{i}+O(q-1)$. The last term vanishes when the limit as $q\to1$ is taken.

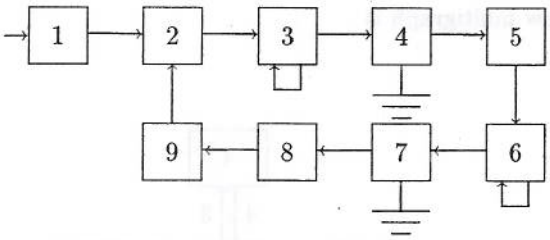
8.2.2.1–8. Let $F_{ni}=n!T_{ni}/i!$. Then $F_{ni}=(n-i)F_{n-1,i-1}+F_{n-1,i}$. Using $i'=n-i$, $n'=n$ gives $F_{ni}=iF_{n-1,n-i-1}+F_{n-1,n-i}$. Letting $H_{ni}=F_{n,n-i}$ gives $H_{ni}=iH_{n-1,i}+H_{n-1,i-1}$, so H_{ni} is a linear combination of Stirling numbers of the second kind. This gives $T_{ni}=\sum_k a_k(n!/i!)\left\{{n-k\atop n-i}\right\}$. The boundary condition $T_{2i}=\delta_{i1}$ (a two node tree always has exactly one node of degree one) gives $T_{ni}=(n!/i!)\left\{{n-2\atop n-i}\right\}$.

8.2.2.1–9. Transform the indices using eqs. (142–143) with $a=0$, $b=1$, $c=1$, $d=1$, $p=0$, $q=1$, $r=1$, and $s=0$ to obtain $F_{ni}=2F_{n-1,i}+F_{n-1,i-1}$, or $\frac{1}{2}F_{ni}=F_{n-1,i}+\frac{1}{2}F_{n-1,i-1}$. Let $f(n)g(i)H_{ni}=F_{ni}$, with $f(n)=2^n$ and $g(i)=2^{-i}$. Then $H_{ni}=H_{n-1,i}+H_{n-1,i-1}$, so H_{ni} is a linear combination of binomial coefficients, i.e., $H_{ni}=\sum_j a_j\binom{n}{i-j}$. A useful boundary condition for this problem is $N_{k0}=\delta_{k0}$, which gives $F_{0k}=\delta_{k0}$. Since $2^{n-i}H_{ni}=F_{ni}$, $H_{0i}=\delta_{i0}$, so $a_j=\delta_{j0}$ and $H_{ni}=\binom{n}{i}$. Thus $N_{in}=F_{ni}=2^{n-i}\binom{n}{i}$.

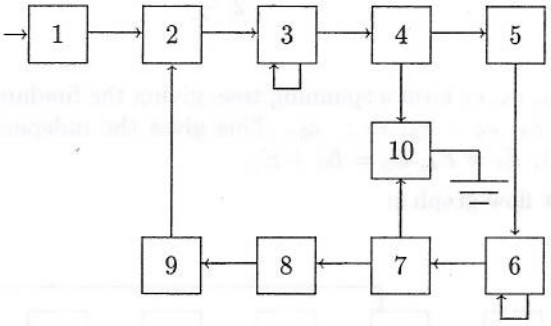
8.2.3–1. Replacing $m+1$ with m gives the recurrence $mF_{mn}=nF_{m-1,n-1}$, which is a one dimensional equation. The solution is $F_{m,n}=F_{m-n,0}/\binom{m}{n}$ for $m\ge n$ and $F_{m,n}=F_{0,n-m}\binom{n}{m}$ for $n\ge m$.

9.1–1. Subgraphs that are four single node graphs and the 10 graphs with edges $(\{1,2\})$, $(\{2,3\})$, $(\{2,4\})$, $(\{3,4\})$, $(\{1,2\},\{2,3\})$, $(\{1,2\},\{2,4\})$, $(\{2,3\},\{3,4\})$, $(\{1,2\},\{2,3\},\{2,4\})$, $(\{1,2\},\{2,3\},\{3,4\})$, $(\{1,2\},\{2,4\},\{3,4\})$. The last four graphs are spanning trees.

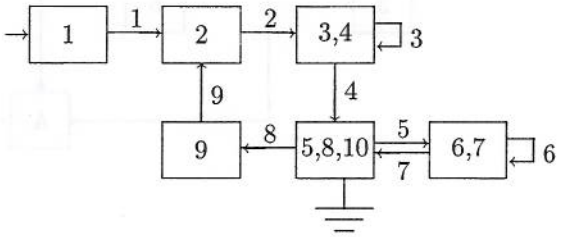
9.1–2.



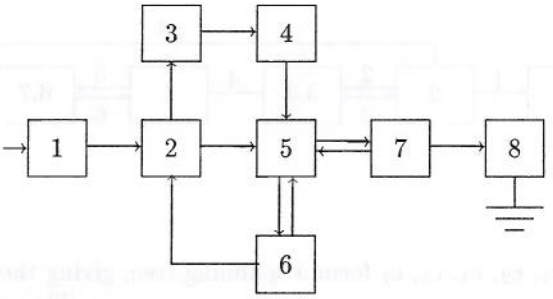
9.1.1–1. First transform the answer to Exercise 9.1–2 into



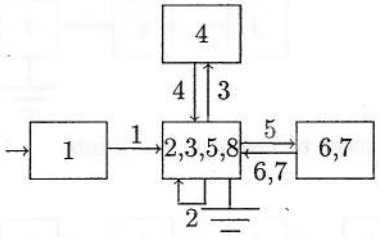
The edge flow multigraph is



9.1.2–1. The single exit flow graph is

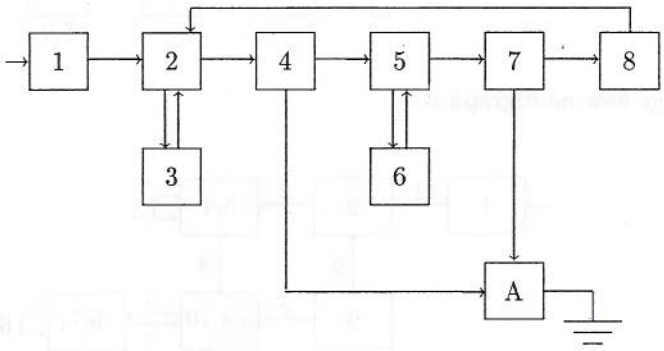


The edge flow multigraph is

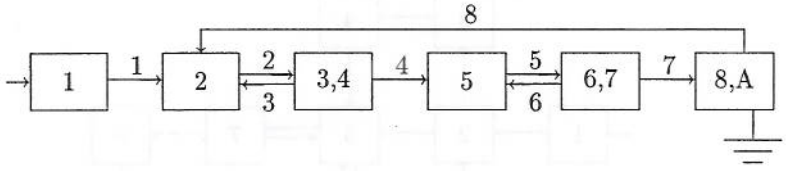


The edges e_1, e_3, e_5 form a spanning tree, giving the fundamental cycles $e_0 : e_1$; $e_2 : ;\ e_4 : e_3$; $e_6 : e_5$; $e_7 : e_5$. This gives the independent flow equations $E_1 = E_0 = 1$, $E_3 = E_4$, $E_5 = E_6 + E_7$.

9.1.2–2. The single exit flow graph is

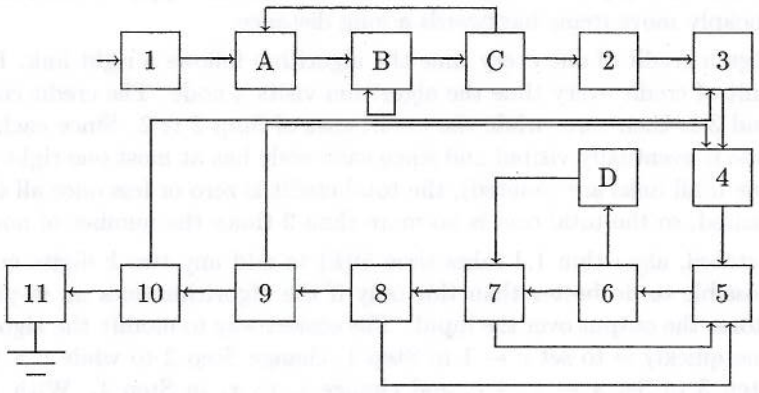


The edge flow multigraph is

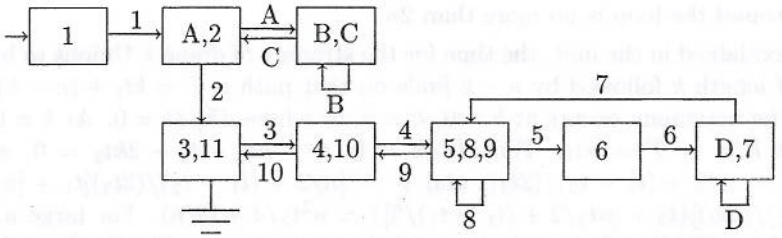


The edges e_1, e_2, e_4, e_5, e_7 form a spanning tree, giving the fundamental cycles $e_0 : e_1, e_2, e_4, e_5, e_7$; $e_3 : e_2$; $e_6 : e_5$; $e_8 : e_2, e_4, e_5, e_7$. This gives the independent flow equations $E_1 = E_0 = 1$, $E_2 = E_3 + E_8$, $E_4 = E_7 = E_8$, $E_5 = E_6 + E_8$.

9.1.2–3. The flow graph is



The edge flow multigraph is



The edges e_1, e_C, e_2, e_3, e_4, e_5, e_6 form a spanning tree, giving the fundamental cycles e_0 : e_1, e_2; e_A : e_C; e_B : ; e_D : , e_7 : e_5, e_6; e_8 : ; e_9 : e_4; e_{10} : e_3. This gives the independent flow equations $E_1 = E_2 = E_0 = 1$, $E_A = E_C$, $E_3 = E_{10}$, $E_4 = E_9$, $E_5 = E_6 = E_7$.

9.2.2–1. The number of inversions of A with respect to B counts the number of pairs (i, j) such that $i < j$ and $k_i > k_j$. Since the mapping between i and k_i is one-to-one, this is the same as the number of pairs (k_j, k_i) such that $k_j < k_i$ and $j > i$, which counts the number of inversions of B with respect to A.

9.2.2–2. Deleting creates no inversions. It removes x. The time is k and the credit time is $k - x$. In the text it is shown that $k - x \le i$, so eq. (8) applies to deletions as well as insertions, and the rest of the analysis is the same as that for insertions.

9.2.2–3. The analysis in the text applies so long as an item is moved forward. If an item is moved backwards, assume that the cost is proportional to how far backwards it is moved. (This is a realistic assumption for algorithms that do not have any additional pointers, but it is not realistic for an algorithm that keeps a pointer to the end of the list.) Consider moving an item backwards j positions. The effect of the Move to Front algorithm is as given on page 382; it creates $k - 2x - 1$ inversions. The move backwards increases the number of inversions by j. The cost of algorithm A accessing the i^{th} item and moving it backwards is $i + j$. The amortized cost for algorithm M is no more than $2(k - x) - 1 + j$. Since $k - x \le i$, the cost is no more than $2i - 1 + j$, so we still have that $C_M \le 2C_A - 1$, and the rest of the analysis is the same as that that follows eq. (8). Notice the importance

of the having j in the cost. The analysis does not apply to algorithms that can cheaply move items backwards a long distance.

9.2.2–4. Assign a credit of one every time the algorithm follows a right link. Pay back one unit of credit every time the algorithm visits a node. The credit cost of Steps 1 and 3 is then zero, while the credit cost of Step 2 is 2. Since each node in the tree is eventually visited and since each node has at most one right link (exactly one if nil links are counted), the total credit is zero or less once all the nodes are visited, so the total cost is no more than 2 times the number of nodes.

9.2.2–5. As stated, algorithm 1.4 takes time $\Theta(k)$ to add any two k digits numbers. It is possible to do better than this only if the algorithm does an in-place add and stores the output over the input. The easiest way to modify the algorithm to add one quickly is to set $c \leftarrow 1$ in Step 1, change Step 2 to while $c > 0$ do, change Step 3 to Set $s \leftarrow x_i + c$, and change z_i to x_i in Step 4. With the modified algorithm, $\lfloor n/2 \rfloor$ of the numbers in the range 0 to $n-1$ cause the loop to be done once, $\lfloor n/4 \rfloor$ cause it to be done twice, etc., so the total number of times around the loop is no more than $2n$.

9.2.3–1. As explained in the hint, the time for the strategy of doing k Unions to build a path of length k followed by $n-k$ finds on that path is $T = kt_1 + (n-k)(t_2 + kt_3)$. The maximum occurs at $k = 0$, $k = n$, or where $dT/dk = 0$. At $k = 0$, $T = nt_2$, at $k = n$, $T = nt_1$. For $dT/dk = 0$, $t_1 + nt_3 - t_2 - 2kt_3 = 0$, which gives $k = n/2 + (t_1 - t_2)/(2t_3)$, and $T = [n/2 + (t_1 - t_2)/(2t_3)]t_1 + [n/2 - (t_1 - t_2)/(2t_3)]\{t_2 + [nt_3/2 + (t_1 - t_2)/2]\} = n^2 t_3/4 + O(n)$. For large n, the place where $dT/dk = 0$ gives the largest value since it grows like $O(n^2)$ while the other two values are linear in n. Any other strategy that used k Unions would take less time, because it would take the same time for Unions, but less time for Finds.

9.2.3.1–1. Let h_n be the height of the tallest tree that can be built with n calls to **Union with Weights**. Then $h_n = 1 + \max 0 \le i \le \lfloor i/2 \rfloor \{h_i\}$ with $h_0 = 0$. Since h_n is nondecreasing, $h_n = 1 + h_{\lfloor i/2 \rfloor} = 1 + \lfloor \lg n \rfloor$. It is better to do all the Unions first. For i Unions and $n-i$ finds, the time is $\max_i\{i + (1 + \lfloor \lg i \rfloor)(n-i)\} = n + \max_i\{(1 + \lfloor \lg i \rfloor)(n-i)\}$. Setting $i = n/(\lg n)$ gives a lower bound of $n \lg n - n \lg \lg n + (n \lg \lg n)/(\lg n)$. Setting $i = n$ in $\lg i$ and $i = 0$ in $n - i$ gives an upper bound of $n + n \lg n$, so the worst-case time is $n \lg n + O(n)$.

9.3.1–1. The i^{th} component of the vector $\mathbf{F}^{-1}(\mathbf{F}(x))$ is given by
$(1/n) \sum_{0 \le j < n} (\sum_{0 \le k < n} x_k \omega^{jk}) \omega^{-ij} = (1/n) \sum_{0 \le j < n} \sum_{0 \le k < n} x_k \omega^{(k-i)j} =$
$(1/n) \sum_{0 \le k < n} x_k \sum_{0 \le j < n} \omega^{(k-i)j}$. Now, $\sum_{0 \le j < n} \omega^{(k-i)j} = \delta_{ik}$ because for $i \ne k$,
$\sum_{0 \le j < n} \omega^{(k-i)j} = (1 - \omega^{(k-i)n})/(1 - \omega^{k-i})$ and for $i = k$,
$(1/n) \sum_{0 \le k < n} x_k \sum_{0 \le j < n} \omega^{(k-i)j} = \sum_{0 \le j < n} \omega^{(k-i)j} = \sum_{0 \le j < n} 1 = n$. So, the i^{th} component is $(1/n) \sum_{0 \le k < n} x_k \sum_{0 \le j < n} n\delta_{ik} = x_i$.

9.3.1–2. The m^{th} component of $(\mathbf{F}A) \cdot (\mathbf{F}B)$ is $\sum_{0 \le k < n} \sum_{0 \le j < n} a_k \omega^{mk} b_j \omega^{mj} =$
$\sum_{0 \le k < n} \sum_{0 \le j < n} a_k b_j \omega^{m(j+k)}$. Let $j' = j + k$ and $k' = k$ ($j = j' - k'$, $k = k'$) to obtain $\sum_{0 \le k < n} \sum_{k \le j < n+k} a_k b_{j-k} \omega^{mj}$ for the m^{th} component. When B has period n, $b_{j-n} = b_j$, this can be written as
$\sum_{0 \le k < n} \left(\sum_{k \le j < n} a_k b_{j-k} \omega^{mj} + \sum_{0 \le j < k} a_k b_{j-k} \omega^{mj} \right) =$

$\sum_{0\le k<n}\sum_{0\le j<n} a_k b_{(j-k) \bmod n}\omega^{mj}$. The m^{th} component of $\mathbf{F}(A \circledast B)$ is $\sum_{0\le j<n}\sum_{0\le k<n} a_k b_{(j-k) \bmod n}\omega^{mj}$.

9.3.1–3. See Knuth [10, Section 4.3.3.A].

9.3.2–1.

Algorithm 9a.2 **Add one:** Input: Arrays A and B with elements indexed 0 to $k-1$ which represent a number m in forward and in reverse binary. Output: Arrays A and B which represent the number $m+1$ in forward and reverse binary.

Step **1.** $j \leftarrow 0$.

Step **2.** While $A_j = 1$ do

Step **3.** Set $A_j \leftarrow 0$, $B_{k+1-j} \leftarrow 0$, and $j \leftarrow j+1$.

Step **4.** Set $A_j \leftarrow 1$ and $B_{k+1-j} \leftarrow 1$.

9.3.2–2. Step 1 is done $1+\lg n$ times. Step 2 is done $(n+1)(1+\lg n)$ times. Step 3 is done $1+\lg n$ times. Step 4 is done $n+1$ times. Each step can be done in constant (amortized) time if care is taken. (See the remarks following the algorithm.)

9.3.2–3. In the inner loop both algorithms must do one addition, one subtraction, one adding of indices, four indexing operations, and one calculation of *odd*. If the calculation of *odd* is done by tabler look up, it will use one multiplication, one masking operation, and one indexing operation. The FFT also uses some bit operations, which will be quick on most machines. The RFFT also needs to do n subroutine calls which pass $2n + 2\sum_{1\le i\le k} 2^{n-i}2^i = 2n + 2n\lg n$ parameters. On many machines the time for the subroutine calls will be some what larger than the time for the calculation. All in all one should expect RFFT to take between 150 percent and 300 percent of the time of FFT.

9.3.3–1. See Aho, et. al. [1, Section 7.3].

9.3.3–2. $\omega^{n/2} \equiv -1 \pmod{\omega^{n/2}+1}$, so $\sum_{0\le j<k} x_j\omega^{jn/2} = \sum_{0\le j<k} x_j(-1)^j$.

9.3.3–3. The value of n is 8, the value of ω is 4 (2 is not big enough), the value of m is $\omega^{n/2}+1 = 257$. The inverse transform needs $\omega^{-1} = 193$ and $1/n = 225$. The following tables summarize the calculation of the Fourier transforms and the inverse transform.

	i	1	2	3	t		i	1	2	3	t		i	1	2	3	t	1/8
x_0	1	1	4	10	10	y_0	5	5	12	26	26	z_0	3	7	7	40	40	5
x_1	2	2	6	255	56	y_1	6	6	14	255	139	z_1	74	209	33	231	128	16
x_2	3	3	255	223	223	y_2	7	7	255	223	223	z_2	128	0	7	15	15	34
x_3	4	4	255	30	97	y_3	8	8	255	30	52	z_3	161	81	128	256	223	60
x_4	0	1	49	56	255	y_4	0	5	117	139	255	z_4	4	256	15	128	231	61
x_5	0	2	66	42	42	y_5	0	6	134	95	95	z_5	135	196	195	159	159	52
x_6	0	3	210	97	30	y_6	0	7	150	52	30	z_6	129	256	240	223	256	32
x_7	0	4	195	66	66	y_7	0	8	135	248	248	z_7	177	241	197	0	0	0

In the table i indicates initial data (or x_iy_i for z), the numbers on the top line indicate the various values of j in the FFT Algorithm, t indicates the results of Step 4 of the FFT, and 1/8 indicates the effect of dividing by 8 (multiplying by

225) when computing the inverse transform. The value $\omega = 4$ was used during the transform and the value $\omega^{-1} = 193$ was used during the inverse transform. The 1/8 column has the final answer.

9.3.3–4. $\omega = 4$, $m = 15$, $n = 2$. $4^2 \equiv 1 \pmod{15}$, $4^1 \equiv 4 \not\equiv 1 \pmod{15}$, $4^0 + 4^1 \equiv 5 \not\equiv 0 \pmod{15}$.

9.4.3–1. $O(k^3)$. The i^{th} multiplication multiplies a number of size $O(i)$ words by a one-word number. There are k such multiplications.

9.4.4–1. $w = 4,616,628/98,636,952 = 384,719/8,219,746$, $x = 455,172/98,636,952 = 37,931/8,219,746$, $y = 3,158,742/98,636,952 = 526,457/16,439,492$, $z = 5,504,568/98,636,952 = 229,357/4109873$. Mod 20011, the determinant is 14,098 and $w = 14,098$, $x = 14,930$, $y = 17,015$, $z = 1543$. Mod 20021, the determinant is 18,814 and $w = 11,798$, $x = 14,710$, $y = 15,445$, $z = 18,814$.

10.1–1. n elements must be output, so time $O(n)$ is required. (Also the answer depends on all $2n$ input elements, so the amount of input also shows that time $O(n)$ must be used.)

10.2.1–1. $\lg(5!) \approx 6.9$ which rounds up to 7.

10.2.1–2. See Knuth [11, p. 184].

10.2.2–1. The obvious algorithm for merging is to compare the first two elements of each list and output the smaller element. Repeat this process until one list is empty. After one list becomes empty, output the rest of the other list. The remaining length of the other list will be between 1 and m, so this algorithm uses $2m - 1$ comparisons in the worst case.

10.2.2–2. Consider a set with $n + 1$ distinct elements, one of which is q. There are n ways to select n distinct elements one of which is q, and there is one way to select n elements none of which is q. Thus, there are $n + 1$ cases, and any correct algorithm must be able to isolate the one case where q is not in the set. This gives $n + 1 \le 2^k$, where k is the number of comparisons, or $k \ge \lg(n + 1)$. Since k is an integer, $k \ge \lceil \lg(n + 1) \rceil$.

10.2.2–3. Consider a set with $n+k$ distinct elements, k of which are the same as the elements of the k queries. There are $\binom{n}{i}$ ways to select i particular query elements and $n-i$ nonquery elements. For each i there are $\binom{k}{i}$ possible choices for the i query elements. This is a total of $\sum_i \binom{k}{i}\binom{n}{i} = \binom{n+k}{k}$ cases. Any correct algorithm must be able to isolate the one case where none of the queries are in the set. This gives $\binom{n+k}{k} \le 2^k$, where k is the number of comparisons, or $k \ge \lg \binom{n+k}{k}$. Since k is an integer, $k \ge \left\lceil \lg \binom{n+k}{k} \right\rceil$.

10.2.3–1 to 3. See Knuth [11, Section 5.3.3].

10.3.2–1. If $M(n) = \Theta(n^2)$ then the analysis in the section implies that there is some upper and lower bound on the limit, but it does not give much information about what the value of the limit is. If $M(n)$ increases more rapidly than n^2 then you can obtain the following limits. Eq. (13) gives $M(n) \le T(3n)+O(n^2)$. The n^3 algorithm for transitive closure gives $T(3n) \le 27T(n)$, so $M(n) \le 27T(n) + O(n^2)$. If the $O(n^2)$ grows more slowly than the other terms, then $\lim_{n\to\infty} T(n)/M(n) \le 1/27$. Eq. (25) gives $\lim_{n\to\infty} T(n)/M(n) \ge 3$ under the same assumptions.

10.3.2–2. The transitive closure of A_{22} and the transitive closure of $A_{11} \vee A_{12}[I \vee (A_{22})^+]A_{21}$.

10.3.2–3. No. The time would be $O(n^{\lg 3})$

10.3.2–4. Since the matrix A can be embedded in an array of no more than twice the original size, $T(n) \le T(2^k) \le 3M(2^k) + O(2^{2k})$ where $2^{k-1} \le n \le 2^k$. Since the time for matrix multiplication is no more than n^3, $M(2^k) \le 2^3 M(n)$, so $T(n) \le 24M(n) + O(n^2)$.

10.3.2–5. Let's consider the extreme case where $n = 2^{k-1}+1$. In this case, the algorithm that uses the actual size rather than size 2^k is probably almost as fast the algorithm are on problems of size 2^k. In other words, for large problems it is faster by a factor of between 4 and 8 (depending on how fast Boolean Matrix Multiplication can be done). Let $M_{XYZ}(n)$ where X, Y, and Z can be 0 or 1, be the time to multiply a matrix of size $n+X$ by $n+Y$ with a matrix of size $n+Y$ by $n+Z$. Let $A_{XY}(n)$ be the time required to add to matrices of size $n+X$ by $n+Y$, which is $Cn^2 + O(n)$. The time for copying matrices is similar to the time for adding. Then the time for the algorithm that uses the actual size is $T(2^{k-1}+1) = T(2^{k-2}+1) + T(2^{k-2}) + M_{011}(2^{k-2}) + M_{010}(2^{k-2}) + M_{010}(2^{k-2}) + M_{001}(2^{k-2}) + M_{100}(2^{k-2}) + M_{101}(2^{k-2}) + Kn^2 + O(n)$, while the time for the algorithm that uses powers of 2 is (for problems of size 2^{k-1} $T(2^{k-1}) = 2T(2^{k-2}) + M_{000}(2^{k-2}) + M_{000}(2^{k-2}) + M_{000}(2^{k-2}) + M_{000}(2^{k-2}) + M_{000}(2^{k-2}) + Kn^2 + O(n)$. The difference of the two recurrences is $T(2^{k-1}+1) - T(2^{k-1}) = T(2^{k-2}+1) - T(2^{k-2}) + M_{011}(2^{k-2}) - M_{000}(2^{k-2}) + M_{010}(2^{k-2}) - M_{000}(2^{k-2}) + M_{010}(2^{k-2}) - M_{000}(2^{k-2}) + M_{001}(2^{k-2}) - M_{000}(2^{k-2}) + M_{100}(2^{k-2}) - M_{000}(2^{k-2}) + M_{101}(2^{k-2}) - M_{000}(2^{k-2}) + O(n)$, which is a recurrence in $T(2^k+1) - T(2^k)$. The size of difference of $M_{110} - M_{000}$, etc. should be small compared to the size of M. (For n^3 matrix multiplication algorithm, the difference is $O(n^2)$.) If $M_{110} - M_{000}$, etc. have about the same size as M_{000} then the two approaches give similar times.

10.3.2–6. Eq. (21) only gives an upper bound on the value of $T(n)$. (If $T(n)$ turns out to grow only as rapidly as n^2, the big O term would permit an equal sign.)

10.3.2–7. See Lawler [12, Section 4.2].

10.4–1. $H_{i,-p(n)} \vee H_{i,-p(n)+1} \vee \cdots \vee H_{i,p(n)}$ for $0 \le i \le p(n)$, $\neg H_{ij} \vee \neg H_{ik}$ for $0 \le i \le p(n)$, $-p(n) \le j < k \le p(n)$.

10.4–2. The first group has $p(n)+1$ clauses of $2p(n)+1$ literals. The second group has $[p(n)+1]2p(n)[2p(n)+1]/2$ clauses of 2 literals.

10.4–3. $S_{ij1} \vee S_{ij2} \vee \cdots S_{ij|\Gamma|}$ for $0 \le i \le p(n)$, $-p(n) \le j \le p(n)$ and $\neg S_{ijk} \vee \neg S_{ijk'}$ for $0 \le i \le p(n)$, $-p(n) \le j \le p(n)$, $1 \le k < k' \le |\Gamma|$.

10.4–4. $2p(n)^2 + p(n)$ clauses of length $|\Gamma|$ and $(2p(n)^2 + p(n))|\Gamma|(|\Gamma|-1)/2$ clauses of length 2.

10.4–5. Let s be the number of states in the machine. There are $p(n)(2p(n)+1)|\Gamma|s$ clauses of each of the following three types: $\neg Q_{ik} \vee \neg H_{ij} \vee \neg S_{ijl} \vee Q_{i+1,k'}$, $\neg Q_{ik} \vee \neg H_{ij} \vee \neg S_{ijl} \vee H_{i+1,j-1}$ (or $\neg Q_{ik} \vee \neg H_{ij} \vee \neg S_{ijl} \vee H_{i+1,j+1}$), $\neg Q_{ik} \vee \neg H_{ij} \vee \neg S_{ijl} \vee S_{i+1,j,l'}$. The total is $3p(n)(2p(n)+1)|\Gamma|$.

10.4–6. $S_{iml} \vee \neg S_{i+1,m,l}$, $\bar{S}_{iml} \vee S_{i+1,m,l}$, for $0 \le i \le p(n)$, $-p(n) \le m \le p(n)$ with $m \ne j$, $1 \le l \le |\Gamma|$.

10.4–7. $4p(n)^2|\Gamma|$.

10.4.1–1. The proof that the problem is in NP is given in the problem statement. To prove that the problem is NP hard, consider any 3-satisfiability problem. Let the i^{th} clause be $l_{i1} \vee l_{i2} \vee l_{i3}$. The 3-satisfiability problem has a solution if and

if only the problem $\wedge_i(Q_1(l_{i1}, a_i) \wedge Q_2(l_{i2}, a_i) \wedge Q_3(l_{i3}, a_i)$ has a solution. (It is understood that when l is a negative literal, $Q_j(l, a)$ is shorthand for $Q'_j(l', a)$ where l' is the variable of l.) If there is some setting of the variables that satisfies the 3-satisfiability problem, then setting the corresponding variables for the Q's the same way. Set a_i to j if j is the smallest index such that l_{ij} is true. (There must be such a value for j because each clause in the 3-satisfiability problem must have at least one true literal.) If there is some setting of the variables in the problem with the Q, then for each literal l_{ia_i} set the corresponding variable in the 3-satisfiability problem in the way that corresponds to making the literal true. Any variables that are not assigned a value by this process can be given any value.

10.4.1–2. The following procedure works in polynomial time on predicates where each clause has no more than two literals.

Algorithm 10a.3 2-Sat: Input: A predicate P that is the conjunction of clauses where each clause consists of no more than two literals. Output: "Yes" if the predicate is satisfiable and "No" if it is not. Initially, no variable has its value set.

Step **1.** Simplify the predicate by dropping all clauses with a true literal and by dropping all false literals from clauses. If any clause becomes empty, output "No". If the predicate becomes empty, output "Yes".

Step **2.** If the predicate has any clause of length one, set it to the value that makes it true and go to Step 1.

Step **3.** If any variable appears in only positive literal (literals without a not sign) or only in negative literals, set it to the value that makes the clauses true and go to Step 1.

Step **4.** At this point every clause has length 2 and every variable appears in both positive and negative clauses. Pick any unset variable and consider both setting it to true and to false. For each value do Steps 1–3. (Use a procedure that has just those three steps.) If either value leads to the output "Yes" for the subcalculation, then set the variable to that value and go to Step 1. (For greatest efficiency, do the two calculations in parallel and use the one that finishes first. Also retain the results of the variable settings done in Steps 2 and 3, and just repeat Step 4 instead of going to Step 1.)

The reason that this algorithm can work correctly in polynomial time is that at Step 4 when a variable is set, it forces the setting of the other variables that depend on it (either immediately or after other variables are set) so we can tell if a value is going to work without having to consider two values of other variables. In 3-satisfiability things are not so simple. When one variable is set, there are two variable, either of which can be set in an attempt to satisfy the predicate.

10.4.2–1. See Garey and Johnson [4, p 54].

10.4.2–2. See Garey and Johnson [4, p 54].

10.4.3–1. The number of nodes in the constructed graph is $e + K$ where e is the number of edges in the original graph and K is the bound on the vertex cover. The number of edges is $16e + Kv$.

11.1–1. $F(x) = 0$ for $x < 2$, $F(x) = 1/36$ for $2 \le x < 3$, $F(x) = 3/36$ for $3 \le x < 4$, $F(x) = 6/36$ for $4 \le x < 5$, $F(x) = 10/36$ for $5 \le x < 6$, $F(x) = 15/36$ for $6 \le x < 7$, $F(x) = 21/36$ for $7 \le x < 8$, $F(x) = 26/36$ for $8 \le x < 9$, $F(x) = 30/36$ for $9 \le x < 10$, $F(x) = 33/36$ for $10 \le x < 11$, $F(x) = 35/36$ for $11 \le x < 12$, $F(x) = 1$ for $x \ge 12$.

11.1–2. $F(x) = (\frac{1}{2})^n \sum_{0 \le i \le x} \binom{n}{i}$.

11.1–3. $F(x) = 1 - (k/N)^{\lfloor x \rfloor - 1}$ for $x \ge 1$, $F(x) = 0$ for $x < 1$.

11.1–4. $F(x) = 1 - \binom{k}{\lfloor x \rfloor - 1} / \binom{N}{\lfloor x \rfloor - 1}$.

11.2–1. The null hypothesis is that either method is equally likely two games of a pair. Under the null hypothesis, the probability that one method would win 70 or more games out of 95 (two tail test) is $2^{-95}(\sum_{0 \le i \le 25} \binom{95}{i} + \sum_{70 \le i \le 95} \binom{95}{i} = 4 \times 10^{-4}$.

11.2–2. To be 95 percent sure that the coin comes up heads more often than tails we need the smallest n such that $\sum_{n/2 \le i \le n} \binom{n}{i} (2/3)^i (1/3)^{n-i} \ge 0.95$.
$\sum_{n/2 \le i \le n} \binom{n}{i} (2/3)^i (1/3)^{n-i} = 3^{-n} \sum_{n/2 \le i \le n} \binom{n}{i} 2^i$. The answer is that 22 coin flips are needed. 22 flips gives 99.42 percent confidence, while 21 flips only gives 94.7 percent confidence. Suppose you want to test a coin for bias at the 95 percent confidence level. If we do a two tailed test with n flips and call the coin biased if we get either k or less heads or $n - k$ or more heads, then the confidence level of the test is $2^{-n} \sum_{k < i < n-k} \binom{n}{i}$. If the probability for the coin being tested is p, then the probability that the coin being tested will test okay is $\sum_{k < i < n-k} \binom{n}{i} p^i (1-p)^{n-i}$. The number of flips we need to have a 95 percent chance of detecting that the coin is biased depends on how sure we want to be that we are correct. If we want to be 95 percent sure of being correct while having a 95 percent chance of detecting a coin that comes up heads 95 percent of the time, we need to choose n and k so that $2^{-n} \sum_{k < i < n-k} \binom{n}{i} > 0.95$ and $3^{-n} \sum_{k < i < n-k} \binom{n}{i} 2^i < 0.05$. If we assume that the binomial distribution is approximately normal (with mean pn and standard deviation $\sqrt{p(1-p)n}$) then the first condition is equivalent to $n/2 - k \approx (1.92/2)\sqrt{n}$ or $k \approx \frac{1}{2}n + 0.98\sqrt{n}$. The second condition is equivalent to $(\frac{2}{3}n - \frac{1}{2}n - 0.98\sqrt{n})/(\frac{1}{3}\sqrt{2n}) \approx 1.96$. This gives $n \approx 133$ and $k \approx \frac{1}{2}n + 11$. To obtain an exact answer it would be necessary to explore various integers near these values. Informally, the reason the second calculation requires such a large n is that n must be larger enough that $n/2$ and $2n/3$ are far enough apart that a point in between can be about $\sqrt{n}$ away from each one. The first calculation just required that $n/2$ be about $\sqrt{n}$ away from $2n/3$.

11.3–3. The answers change and become much harder to calculate exactly. It is a good idea to approximate the binomial distribution with a normal distribution. The mean for n flips of the coin is $0.51n$ and the standard deviation is $\sqrt{0.51 \cdot 0.49n} \approx 0.50\sqrt{n}$. For the first part of the problem we need $(0.51n - 0.50n)/(0.50\sqrt{n}) \approx 1.65$ or $n \approx 83$. For the second part we need $(0.51n - 0.50n - 0.98\sqrt{n})/(0.50\sqrt{n}) \approx 1.96$ or $n \approx 38416$. With slightly biased dice you have to be really patient to obtain good evidence that the dice are biased.

11.4–1. The mean is $\int_{-\infty}^{+\infty} t dF(t) = \int_0^q \mu t e^{-\mu t}\, dt + q e^{-\mu q} = -te^{-\mu t}|_0^q + \int_0^q e^{-\mu t}\, dt + q e^{-\mu q} = e^{-\mu q}(-q - 1/\mu) + 1/\mu + q e^{-\mu q} = (1 - e^{-\mu q})/\mu$. The average of t^2 is $\int_{-\infty}^{+\infty} t^2 dF(t) = \int_0^q \mu t^2 e^{-\mu t}\, dt + q^2 e^{-\mu q} = -t^2 e^{-\mu t}|_0^q + 2\int_0^q \mu t e^{-\mu t}\, dt + q^2 e^{-\mu q} = e^{-\mu q}(-q^2 - 2q/\mu - 1/\mu^2) + 1/\mu^2 + 1/\mu + q^2 e^{-\mu q} = 1/\mu^2 - e^{-\mu q}(2q/\mu + 1/\mu^2)$. The variance is $1/\mu^2 - e^{-\mu q}(2q/\mu + 1/\mu^2) - (1 - e^{-\mu q})^2/\mu^2 = e^{-\mu q}/\mu^2(1 - 2q/\mu) + (-e^{-2\mu q})/\mu^2$.

11.4–2. $f(t) = \mu e^{-\mu t}$ for $t \geq 0$, and $f(t) = 0$ for $t < 0$.

11.4–3. $\phi(t) = \int_{-\infty}^{+\infty} e^{itx}\, dF(x) = \sum_{0 \leq j \leq n} \binom{n}{j} p^j (1-p)^{n-j} e^{itj} = [(1-p) + pe^{it}]^n = [1 + p(e^{it} - 1)]^n$.

11.4–4. $\kappa_j = (d^j/dt^j) \ln \phi(-it)|_{t=0}$. For this problem, $\kappa_j = (d^j/dt^j) \ln[1 + p(e^t - 1)]^n = n(d^j/dt^j) \ln[1 + p(e^t - 1)]|_{t=0}$. The first few values of κ are $\kappa_1 = np$, $\kappa_2 = np(1-p)$, $\kappa_3 = np(1-p)(1-2p)$, and $\kappa_4 = np(1-p)(1 - 6p + 6p^2)$.

11.5–1. $\frac{1}{2}$.

11.8–1. The theory and experiment differ for E_0 by $0.006/0.032 \approx 0.2$ standard deviations. One would expect a larger deviation about 84 percent of the time. Nothing is suspicious about this. The theory and experiment differ for W_1 by $0.1006/0.0011 \approx 91$ standard deviations. The probability of this happening by chance is about 1 in 10^{1802}, which is to say that it is virtually impossible. The authors notice that something was wrong but they thought the random number generator they used might be to blame when in fact it was their calculation. People find it easy to put the blame in the wrong place.

11.9–1. For a given x and μ, let $p = \mu/x$. With this p the mean is indeed μ and the probability that the distribution gives a value greater than or equal to x is μ/x.

11.9–2. The mean is $p(x - y) + y$ and the variance is $(1-p)y^2 + px^2 - [p(x-y) + y]^2 = p(1-p)(x-y)^2$. Now x occurs with probability p, and x is $(1-p)(x-y)$ above the mean. The value of σ^2/x^2 is $p(1-p)(x-y)^2/x^2$. The value of y in the range 0 to x that makes this as large as possible is $y = 0$. With this value of y we get a ratio of $p(1-p)$. As p approaches zero this approaches the Chebyshev bound.

11.9–3. Direct calculation: $(1/2)^{100} \sum_{i \leq 80} \binom{100}{i} \approx 5.00 \times 10^{-10}$. Normal distribution: the deviation is 6 standard deviations, which gives $\text{Prob}(X \geq 80) \approx 9.87 \times 10^{-10}$. Markov bound: $\text{Prob}(X \geq 80) \leq 50/80 = 0.625$. Chebyshev bound: $\text{Prob}(X \geq 80) \leq 25/6400 \approx 3.81 \times 10^{-3}$. Chernoff bound: solve $80 = 100e^a/(1 + e^a)$ to obtain $a = \ln 4$ and $\text{Prob}(X \geq 80) \leq e^{-80 \ln 4}(\frac{1}{2} + \frac{1}{2}e^{\ln 4})^{100} = 2^{260}5^{100} \approx 4.3 \times 10^{-9}$.

11.10.1–1. The value of e_i is given by $\sum_t e p_{it}(e)$. Multiply eq. (108) by e and sum over e, t_1, and t_2. Multiply eq. (107) by e and sum over e. Add the two results together to obtain

$$
\begin{aligned}
e_i = (1-b) + \frac{1}{2}b(1-p)\Bigg(\sum_{e,t_1,t_2} e \Bigg(& p_{i-1,t_1}\left(\frac{e-1}{2}\right)\sum_m p_{i-1,t_2}(m) \\
& + p_{i-1,t_2}\left(\frac{e-1}{2}\right)\sum_m p_{i-1,t_1}(m)\Bigg) \\
& + bp\sum_m p_{i-1,t_1}(m)p_{i-1,t_2}(e-m-1)\Bigg) \\
= (1-b) + \frac{1}{2}b(1-p)\Bigg(& \sum_{e,t_1} e p_{i-1,t_1}\left(\frac{e-1}{2}\right)\sum_{m,t_2} p_{i-1,t_2}(m) \\
& + \sum_{e,t_2} e p_{i-1,t_2}\left(\frac{e-1}{2}\right)\sum_{m,t_1} p_{i-1,t_1}(m)\Bigg) \\
& + bp\sum_{m,t_1}\sum_{e,t_2} p_{i-1,t_1}(m) e p_{i-1,t_2}(e-m-1).
\end{aligned}
$$

The first sum over m and t_1 is 1, as is the sum over m and t_2. In the first sum over e, we need the change of variables $e' = (e-1)/2$ $(e = 2e'+1)$. Normally this transformation would not be permitted because e' is noninteger when e is even, but the $p_{i-1,t_1}((e-1)/2)$ factor is zero for even e, so no problem arises. The same transformation is also needed for the second sum over e. The last sum over e needs the change of variables $e' = e - m + 1$ $(e = e' + m - 1)$. Applying all this gives

$$
\begin{aligned}
e_i &= (1-b) + b(1-p)\sum_{e,t_1}(2e+1)p_{i-1,t_1}(e) \\
&\quad + bp\sum_{m,t_1}\sum_{e,t_2} p_{i-1,t_1}(m)(e+m-1)p_{i-1,t_2}(e) \\
&= (1-b) + b(1-p) + 2b(1-p)e_{i-1} + bp\sum_{e,t_2} e p_{i-1,t_2}(e) \\
&\quad + bp\sum_{m,t_1} m p_{i-1,t_1}(m) - bp\sum_{m,t_1} p_{i-1,t_1}(m)\sum_{e,t_2} p_{i-1,t_2}(e) \\
&= (1-b) + b(1-p) + 2b(1-p)e_{i-1} + 2bpe_{i-1} - bp \\
&= 1 + 2be_{i-1}
\end{aligned}
$$

11.10.1–2. Proceed as before, except multiply by e^2 instead of by e.

$$
\begin{aligned}
e_i &= (1-b) + \frac{1}{2}b(1-p)\Bigg(\sum_{e,t_1,t_2} e^2 \Bigg(p_{i-1,t_1}\left(\frac{e-1}{2}\right) \sum_m p_{i-1,t_2}(m) \\
&\qquad + p_{i-1,t_2}\left(\frac{e-1}{2}\right) \sum_m p_{i-1,t_1}(m) \Bigg) \\
&\quad + bp \sum_m p_{i-1,t_1}(m) p_{i-1,t_2}(e-m-1) \Bigg) \\
&= (1-b) + \frac{1}{2}b(1-p)\Bigg(\sum_{e,t_1} e^2 p_{i-1,t_1}\left(\frac{e-1}{2}\right) \sum_{m,t_2} p_{i-1,t_2}(m) \\
&\qquad + \sum_{e,t_2} e^2 p_{i-1,t_2}\left(\frac{e-1}{2}\right) \sum_{m,t_1} p_{i-1,t_1}(m) \Bigg) \\
&\quad + bp \sum_{m,t_1} \sum_{e,t_2} p_{i-1,t_1}(m) e^2 p_{i-1,t_2}(e-m-1). \\
&= (1-b) + b(1-p) \sum_{e,t_1} (2e+1)^2 p_{i-1,t_1}(e) \\
&\quad + bp \sum_{m,t_1} \sum_{e,t_2} p_{i-1,t_1}(m)(e+m-1)^2 p_{i-1,t_2}(e) \\
&= (1-b) + b(1-p)(4s_{i-1} + 4e_{i-1} + 1) + bp \sum_{e,t_2} e^2 p_{i-1,t_2}(e) \\
&\quad + bp \sum_{m,t_1} m^2 p_{i-1,t_1}(m) + bp \sum_{m,t_1} p_{i-1,t_1}(m) \sum_{e,t_2} p_{i-1,t_2}(e) \\
&\quad - 2bp \sum_{e,t_2} e p_{i-1,t_2}(e) - 2bp \sum_{m,t_1} m p_{i-1,t_1}(m) \\
&\quad + 2bp \sum_{m,t_1} m p_{i-1,t_1}(m) \sum_{e,t_2} e p_{i-1,t_2}(e) \\
&= (1-b) + b(1-p)(4s_{i-1} + 4e_{i-1} + 1) + 2bps_{i-1} + bp - 4bpe_{i-1} + 2bpe_{i-1}^2 \\
&= 1 + 2b(e_{i-1} + pe_{i-1}^2) + 2b(2-p)s_{i-1}
\end{aligned}
$$

11.11–1. The equations are $4a + 49b = 56$ and $49a + 809b = 850$. The solution is $a = 3654/835 \approx 4.4$, $b = 656/835 \approx 0.79$. The variance of the error is $2988/835 \approx 3.6$.

11.11–2. See Purdom and Stigler [124].

A–1. $\sum_{1 \le i \le j \le n} x_i x_j$.

A–2. $\sum_{1 \le i \le j \le k \le n} x_i x_j x_k$.