

- 1a. What is the value of $\binom{8}{3}$?
- 1b. What is the value of $\binom{9}{3}$?
2. Suppose you repeat a process n times. Each time you do the process, p is the probability of a successful outcome. What is most likely number of successes? (Remember that it will always be an integer.) Make clear what method you use to solve the problem.
3. Simplify $\sum_{1 \leq i \leq n} i \binom{i}{5}$.
4. The following sum does not have a closed form: $\sum_{0 \leq i \leq m} \binom{n}{i} p^i (1-p)^{n-i}$.
- 4a. Define $f(i)$ to be $\binom{n}{i} p^i (1-p)^{n-i}$. Compute and simplify
- $$x = f(m)/f(m-1).$$
- 4b. Show when m is not too big then x is an increasing (or decreasing, you decide which) function of m .
- 4c. Give a simple upper bound on $\binom{n}{i} p^i (1-p)^{n-i}$ in terms of x^i . (The fact that you can do this depends on the result from 4b, and it is true only under the conditions where 4b is true.)
- 4d. Use the result from 4b to compute an upper for the sum at the beginning of the problem. (This result depends on 4c.)

5. (Counts as two questions.) Consider the following algorithm, which finds the length of a list or else determines that the list ends in a cycle. The list starts at position h of the array N . An index of zero indicates the end of the list. The idea of the algorithm is that i goes down the list at unit speed, while j goes down the list at double speed. If j comes to the end of the list, then the list does not end in a cycle, and l measures the length of the list. On the other hand, if j runs into i , then the list ends in a cycle.

1. Set $i \leftarrow h, j \leftarrow h, l \leftarrow 0$.
2. If $j = 0$, stop (l is the length of the list).
3. Set $j \leftarrow N[j], l \leftarrow l + 1$.
4. If $j = i$, stop (the list ends in a cycle).
5. If $j = 0$, stop (l is the length of the list).
6. Set $j \leftarrow N[j], i \leftarrow N[i], l \leftarrow l + 1$.
7. If $j = i$, stop (the list ends in a cycle).
8. Go to step 2.

Define i_k to be the k th value of i generated by the algorithm, with $i_0 = h$. Thus, $i_0 = h, i_1 = N[h], i_2 = N[N[h]]$, etc. When possible, define a so that a is the smallest value such that $i_a = i_m$ for some $m > a$, and then define b to be the smallest value greater than a such that $i_a = i_b$. When there is no such a , define a to be the value such that $N[i_a] = 0$ and b to be zero. In other words, when the list ends in a cycle, a is the number of cells down the list where the cycle starts, and $b - a$ is the length of the cycle. When the list does not end in a cycle, a is the length of the list and b is zero.

How long does the algorithm take in terms of a and b ? To avoid excessive details, you only need to say how often Step 2 is done. For partial credit just give some simple upper and lower bounds on the number of times Step 2 is done.