

All answers should be as simple as possible.

1. The following code prints the data on a linked list in reversed order. Make a table that shows, as a function of n (the length of the list), how often each step is done.
Step 1. Set p to the address of *head* (the first element of the linked list). Set q to *link*(p) (the next element of the list). Set n to 0. (In many programming languages, *link*(p) would be written as $p \rightarrow \textit{link}$.)
Step 2. While q is not **nil** do Step 3.
Step 3. Set n to $n + 1$, set p to q , set q to *link*(p).
Step 4. For i from n down to 1 do Steps 5 to 7.
Step 5. p to the address of *head*. Set q to *link*(p). For j from 1 to i do Step 6.
Step 6. Set p to q , set q to *link*(p).
Step 7. Print *data*(p).

The basic idea of this algorithm is to first go down the list to count how long it is. Then with i going from n down to 1, the algorithm goes down the list i places and prints out the item it finds.

2. The following code solves the same problem as the previous code. Make a table that shows as a function of n , the length of the list, how often each step is done.
Step 1. Set s to the address of *head*. Set q to *link*(s). Set p to **nil**.
Step 2. While q is not **nil** do Step 3.
Step 3. Set r to *link*(q), set *link*(q) to p , set p to q , set q to r .
Step 4. While p is not **nil** do Step 5.
Step 5. Print *data*(p). Set r to *link*(p), set *link*(p) to q , set q to p , set p to r .
Step 6. Set *link*(s) to q .

The basic idea of this algorithm is to reverse the list (not a good idea if some other program is using it at the same time) and then to go down the reversed list printing out the items and rereversing the list back to its original order.

3. Candidate A has m supporters, candidate B has n supporters. Each supporter has the same probability (p) of voting for his candidate (and probability $1 - p$ of not voting). For each part give both general formulas and also particular values for the case $m = 2$, $n = 1$, $p = 0.5$.
 - a. What is the probability that A has more votes than B ?
 - b. What is the probability that B has more votes than A ?
 - c. What is the probability that A and B have the same number of votes?
4. Consider the problem of matching up items from list A with items from list B where two items can match if they have the same value. The algorithms listed below are under consideration for the problem. For each algorithm determine the leading term in its running time in terms of $|A|$ and $|B|$ (the lengths of the two lists). Give a rank order of the algorithms for the case where $|A| < |B|$ and where both are large. In this problem you only need to figure out the leading term in the running time so that you can produce this rank ordering. Also, you can say that two algorithms take approximately the same time when their leading terms are close to equal. Here are the ideas for the algorithms. Assume that efficient subalgorithms are used for the various ideas in the algorithms.
 - a. For each item in list A , do linear search of list B until you find a match (or the lack of a match). Pull matching items out of each list (assume that you can do this in constant time), and repeat until all items on list A are processed.
 - b. Like case a, except the roles of A and B are interchanged.
 - c. Like case a, except that list B is sorted (include the time needed for sorting) and then binary search is used to find matches.
 - d. Like case b, except that list A is sorted (include the time needed for sorting) and then binary search is used to find matches.
 - e. Sort both lists, then do a single traversal of the lists, removing matching items as they are found.
5. Consider the number of permutations of n items where each item is in a two-cycle.
 - a. How many such permutations are there for $n = 1, 2, 3$, and 4?
 - b. How many such permutations are there for general n ?

One approach for solving this problem is to first develop a recurrence equation for the solution.