Algorithms Midterm practice problems

Problems

1. You've been hired to store a sequence of n books on shelves in a library. The order of the books is fixed by the cataloging system and cannot be changed; each shelf must store a contiguous interval of the given sequence of books. You are given two arrays $H[1..n]$ and $T[1...n]$, where $H[i]$ and $T[i]$ are respectively the height and thickness of the i^{th} book in the sequence. All shelves in this library have the same length L; the total thickness of all books on any single shelf cannot exceed L. You can adjust the height of each shelf to match the tallest book on that shelf. Describe and analyze an efficient algorithm to assign books to shelves to minimize the total sum of heights of the shelves.

   (a) Suppose all the books have the same height h and the shelves have height larger than h, so every book fits on every shelf. Describe and analyze a greedy algorithm to store the books in as few shelves as possible. [Hint: The algorithm is obvious, but why is it correct?]

   (b) That was a nice warmup, but now here's the real problem. In fact the books have different heights, but you can adjust the height of each shelf to match the tallest book on that shelf. (In particular, you can change the height of any empty shelf to zero.) Now your task is to store the books so that the sum of the heights of the shelves is as small as possible. Show that your greedy algorithm from part (a) does not always give the best solution to this problem.

   (c) Describe and analyze an efficient algorithm to assign books to shelves to minimize the total sum of heights of the shelves.

2. Suppose we are given an array $A[1..n]$ with the special property that $A[1] \geq A[2]$ and $A[n-1] \leq A[n]$. We say that an element $A[x]$ is a local minimum if it is less than or equal to both its neighbors, or more formally, if $A[x-1] \geq A[x]$ and $A[x] \leq A[x+1]$.

   We can obviously find a local minimum in $O(n)$ time by scanning through the array. Describe and analyze an algorithm that finds a local minimum in $O(\log n)$ time.

3. You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed, which you are able to make an accurate estimate of based on publicly available information on the internet. You need to decide which houses to rob, where the only constraint stopping you from robbing each of them is that adjacent houses have security systems connected and it will automatically contact the police if any three adjacent houses were broken into on the same night. (You may assume all houses have positive values.)

   (a) Give a (small) example of why choosing the largest value house first (in other words, being greedy) will not necessarily yield the most money.

   (b) Give an algorithm to compute the maximum amount of money you can gain.

4. The robber from our last problem is back! The thief has found himself a new place for his thievery however, and is again trying to maximize the amount he can steal. However, this time there is only one entrance to this area, called root. Besides the root, each house has one and only one parent house. After a tour, the smart thief realized that all houses in this place form a binary tree. (Clearly, our thief has recently taken data structures.) The thief also quickly realizes after inspecting the security system that it will automatically contact the police if any two directly-linked houses are broken into on the same night.

   Given as input a tree of houses, return the maximum amount of money the thief can rob in one night without alerting the police.

5. Your grandmother dies and leaves you her treasured collection of n radioactive Beanie Babies. Her will reveals that one of the Beanie Babies is a rare specimen worth 374 million dollars, but all the others are worthless. The valuable Beanie Baby is either slightly more or slightly less radioactive than the others, but you don't know which. Otherwise, as far as you can tell, they are all identical.

   You have access to a state-of-the-art Radiation Comparator at your job. The Comparator has two chambers. You can place any two disjoint sets of Beanie Babies in Comparator's two chambers; the Detector will then indicate which of the two subsets emits more radiation, or that the two subsets are equally radioactive. (The two subsets are equally radioactive if and only if they contain the same number of Beanie Babies, and they are all worthless.) The Comparator is slow and consumes a lot of power, and you really aren't supposed to use it for personal projects, so you really want to use it as few times as possible.

   Describe an efficient algorithm to identify the valuable Beanie Baby. How many times does your algorithm use the Comparator in the worst case, as a function of n?

6. Recall that a palindrome is any string that is the same as its reversal. For example, I, DAD, HANNAH, AIBOHPHOBIA (fear of palindromes), and the empty string are all palindromes.

   (a) Describe and analyze an algorithm to find the length of the longest substring (not subsequence!) of a given input string that is a palindrome. For example, BASEESAB is the longest palindrome substring of BUBBASEESABANANA ("Bubba sees a banana."). Thus, given the input string BUBBASEESABANANA, your algorithm should return the integer 8.

   (b) Describe and analyze an algorithm to find the length of the longest subsequence (not substring!) of a given input string that is a palindrome. For example, the longest palindrome subsequence of MAHDYNAMICPROGRAMZLETMESHOWYOUTHEM is MHYMRORMYHM, so given that string as input, your algorithm should output the number 11.

7. After the Revolutionary War, Alexander Hamilton's biggest rival as a lawyer was Aaron Burr. (Sir!) In fact, the two worked next door to each other. Unlike Hamilton, Burr cannot work non-stop; every case he tries exhausts him. The bigger the case, the longer he must rest before he is well enough to take the next case. If a case arrives while Burr is resting, Hamilton snatches it up instead.

   Burr has been asked to consider a sequence of n upcoming cases. He quickly computes two arrays `profit`$[1 \ldots n]$ and `skip`$[1 \ldots n]$, where for each index i,

   - `profit`$[i]$ is the amount of money Burr would make by taking the i^{th} case, and
   - `skip`$[i]$ is the number of consecutive cases Burr must skip if he accepts the i^{th} case

   That is, if Burr accepts the i^{th} case, he cannot accept cases $i+1$ through $i+$ `skip`$[i]$. Design and analyze an algorithm that determines the maximum total profit Burr can secure from these n cases, using his two arrays as input.

8. Suppose that you are given an $n \times n$ checkerboard and a checker. You must move the checker from the bottom edge of the board to the top edge of the board according to the following rule. At each step you may move the checker to one of three squares:

   1) the square immediately above
   2) the square that is one up and one to the left (but only if the checker is not already in the leftmost column)
   3) the square that is one up and one to the right (but only if the checker is not already in the rightmost column)

   Each time you move from square x to square y, you receive $p(x, y)$ dollars. You are given a list of the values $p(x, y)$ for each pair (x, y) for which a move from x to y is legal. Do not assume that $p(x, y)$ is positive.

   Give an algorithm that figures out the set of moves that will move the checker from somewhere along the bottom edge to somewhere along the top edge while gathering as many dollars as possible. You algorithm is free to pick any square along the bottom edge as a starting point and any square along the top edge as a destination in order to maximize the number of dollars gathered along the way. What is the running time of your algorithm?

9. Consider the interval scheduling problem: Given a set of intervals $1..n$, each specified by a left endpoint in $L[1..n]$ and a right endpoint in $R[1..n]$, find the largest subset of intervals such that no two intersect. One natural greedy heuristic for this problem is that of *shortest interval first*: choose the shortest interval (so minimum over all i of $R[i] - L[i]$ to be in the subset, delete any intervals that overlap with it, and recurse on the smaller set.

   (a) Show that this strategy is not optimal; in other words, there exists a set of intervals such that the shortest interval first heuristic fails to return the largest possible set of non-overlapping subsets. (Hint: you don't need very many intervals.)

   (b) Prove that this heuristic gives a two approximation: in other words, if the optimal solution uses o intervals, and greedy uses g, then $g \geq .5o$ (or equivalently, $o \leq 2g$).