Basic Algorithms CSCI-UA.0310

Homework 5

Due: March 8th, 4:00 PM EST

Instructions

Please answer each **Problem** on a separate page. Submissions must be uploaded to your account on Gradescope by the due date and time above.
Please note that no late submission will be accepted for this homework.

Problems to submit

Problem 1 (19 points)

Recall that the bottom-up dynamic programming algorithm for finding the nth Fibonacci number required $\Theta(n)$ extra space. Modify the algorithm to develop a linear-time bottom-up DP approach with $O(1)$ extra space. You have to write the pseudo-code of your algorithm.

Hint: Note that in order to compute $F(i)$, we only need to know $F(i-1)$ and $F(i-2)$, and do not require the previous Fibonacci numbers. Use this remark to replace the auxiliary array of size n used in the algorithm covered in the lecture by two auxiliary variables.

Problem 2 (9+3+15 points)

Recall *the rod cutting problem* discussed in the lecture: Consider a rod of length n inches and an array of prices $P[1 \dots n]$, where $P[i]$ denotes the price of any i inches long piece of rod. Now suppose we have to pay a cost of \$1 per cut.
Thus, for example, if we cut the rod into k pieces of lengths $n_1, n_2, \dots, n_k$, this means that we made $k-1$ cuts, which gives their final selling price as $P[n_1] + \cdots + P[n_k] - (k-1)$.
Let $\textsc{MaxPrice}(n)$ denote the maximum selling price we can get this way among all possible options we have to make the cuts (note that one possible option is to make no cut and sell the whole rod of length n).

(a) Find the recursion that $\textsc{MaxPrice}(n)$ satisfies. In other words, you should write $\textsc{MaxPrice}(n)$ in terms of $\textsc{MaxPrice}(n-1)$, $\textsc{MaxPrice}(n-2)$, $\dots$, $\textsc{MaxPrice}(0)$. Fully justify your answer.

(b) Identify the base case for your recursion in part (a) and find its corresponding value. Justify your answer.

(c) Write the pseudo-code for the bottom-up DP algorithm to compute $\textsc{MaxPrice}(n)$. Find and justify the time complexity of your algorithm in the form of $\Theta(.)$.

Problem 3 (9+3+15 points)

Consider the array $A[1 \dots n]$ consisting of n non-negative integers. There is a frog on the last index of the array, i.e. the nth index of the array. In each step, if the frog is positioned on the i^{th} index, then it can make a jump of size at most $A[i]$ towards the beginning of the array. In other words, it can hop to any of the indices $i, \dots, i - A[i]$.
The goal is to develop a DP-based algorithm to determine whether the frog can reach the 1st index of the array.
For $i = 1, 2, \dots, n$, define the subproblem $\textsc{CanReach}(i)$ to denote true or false depending on whether the

frog can reach the 1st index of the array when it is currently standing on the ith index (so $\textsc{CanReach}(i) =$ true, if it can reach the 1st index, and $\textsc{CanReach}(i) =$ false, if it cannot reach the 1st index).

(a) Find the recursion that $\textsc{CanReach}(n)$ satisfies. In other words, you should write $\textsc{CanReach}(n)$ in terms of some of $\{\textsc{CanReach}(n-1), \textsc{CanReach}(n-2), \ldots, \textsc{CanReach}(1)\}$. Fully Justify your answer.

Hint: What options does the frog have for the first jump? It can jump to any of the following indices: $n, \ldots, n - A[n]$.

(b) Identify the base case for your recursion in part (a) and find its corresponding value. Justify your answer.

(c) Write the pseudo-code for the bottom-up DP algorithm to compute $\textsc{CanReach}(n)$. Justify that the run-time of your algorithm is $O(n^2)$ in the worst-case.

Problem 4 (3+9+15 points)

Suppose we want to find the number of ways to make change for n cents with the use of dimes (10 cents), nickels (5 cents), and pennies (1 cent). Note that the ordering of coins in the change matters! (see the examples below)
For $i = 1, \ldots, n$, define the subproblem $\textsc{MakeChange}(i)$ to denote the number of ways to make change for i cents with the use of dimes (10 cents), nickels (5 cents), and pennies (1 cent).
Below, you can find the number of ways to make change for i cents for $i = 1, \ldots, 7$:

- $i = 1, \ldots, 4$: There is only one way to do that. We have to change only using pennies.
- $i = 5$: There are two ways: $1 + 1 + 1 + 1 + 1$ (use 5 pennies), or 5 (use one nickel).
- $i = 6$: There are three ways: $1 + 1 + 1 + 1 + 1 + 1$ (use 6 pennies), or $1 + 5$ (one penny and one nickel), or $5 + 1$ (one nickel and one penny).
- $i = 7$: There are four ways: $1 + 1 + 1 + 1 + 1 + 1 + 1$ (use 7 pennies), or $1 + 1 + 5$ (two pennies and one nickel), or $1+5+1$ (one penny, one nickel, and one penny), or $5+1+1$ (one nickel and two pennies).

(a) Find the values of $\textsc{MakeChange}(8)$ and $\textsc{MakeChange}(9)$.

(b) Find the recursion that $\textsc{MakeChange}(n)$ satisfies for $n \geq 10$. In other words, you should write $\textsc{MakeChange}(n)$ in terms of some of $\{\textsc{MakeChange}(n-1), \textsc{MakeChange}(n-2), \ldots, \textsc{MakeChange}(1)\}$. Fully justify your answer.

Hint: What options do we have for the first coin of the change?

(c) Write the pseudo-code for the bottom-up DP algorithm to compute $\textsc{MakeChange}(n)$. Find and justify the time complexity of your algorithm in the form of $\Theta(.)$.

Bonus problems

The following problems are optional and will not count towards your final grade. You are encouraged to submit your solutions to receive a feedback.

Bonus Problem 1

Develop an algorithm to compute the nth Fibonacci number in $O(\log n)$ time.

Bonus Problem 2

(a) Can you develop a bottom-up DP algorithm for Problem 3 with $O(n)$ space that runs in $O(n \log n)$ time in the worst case?
If so, write the pseudo-code of your algorithm.

(b) Can you improve the run-time of your algorithm in part (a) to $O(n)$?
If so, write the pseudo-code of your algorithm.