CSCE 411, Fall 2025: Homework 3

Due: Friday, September 19, 11:59 pm CT

Problem 0. (-5 for leaving blank.) Write down all outside resources you consulted, and the names of any other students that you worked with in any way. By turning in this homework, you acknowledge that you have read and abide by all course policies on outside resources and collaboration as stated in the course syllabus. Use of Generative AI tools must be explicitly noted. *If you did not use outside resources, AI tools, or collaborators, state that explicitly.*

ChatGPT and Abdul Bari

Problem 1. (15 points total) Consider the following algorithm, which recursively computes each term of the Fibonacci sequence **without dynamic programming or memoization**.

```
FIB Input: Integer $n \geq 0$
Output: $n$-th Fibonacci number $F(n)$
if $n == 0$ then
    Return: 0   \\Base case: Fib(0) = 0
if $n == 1$ then
    Return: 1   \\Base case: Fib(1) = 1
else
    Return: FIB$(n-1)$ + FIB$(n-2)$   \\Recursive case
end if
```

In class, we stated that such a computation may be especially time-consuming.

(a) (5 points) Let $T(n)$ be the runtime of the above algorithm for computing the n-th Fibonacci number. Write down the recurrence relation for $T(n)$.

$$T(n) = \begin{cases} 1, & n = 0 \text{ or } n = 1, \\ T(n-1) + T(n-2) + O(1), & n \geq 2. \end{cases}$$

(b) (10 points) Prove that the runtime of the algorithm is $2^{\Omega(n)}$.

For all integers $n \geq 0$, the runtime $T(n)$ of $Fib(n)$ satisfies

$$T(n) \;\geq\; 2^{(n-1)/2} < \varphi^n.$$

Proof by induction

Base cases: For $n = 0$, we have

$$T(0) = 1 \;\geq\; 2^{(0-1)/2} = 2^{-1/2}.$$

For $n = 1$, we have

$$T(1) = 1 \;\geq\; 2^{(1-1)/2} = 1.$$

So the base cases hold.

Inductive step: Assume that for all $k < n$ we have

$$T(k) \;\geq\; 2^{(k-1)/2}.$$

For $n \geq 2$, the recurrence for the algorithm gives

$$T(n) \;\geq\; T(n-1) + T(n-2).$$

Applying the inductive hypothesis:

$$T(n) \;\geq\; 2^{(n-2)/2} + 2^{(n-3)/2}.$$

Factor out $2^{(n-3)/2}$:

$$T(n) \;\geq\; 2^{(n-3)/2}\left(2^{1/2} + 1\right).$$

Since $2^{1/2} + 1 = \sqrt{2} + 1 \approx 2.414 > 2$, it follows that

$$T(n) \;\geq\; 2^{(n-3)/2} \cdot 2 \;=\; 2^{(n-1)/2}.$$

Thus, by induction, $T(n) \geq 2^{(n-1)/2}$ for all $n \geq 0$.

HINT: You may use any approach to solve this problem. There are many elegant ways to approach this problem that are within the scope of the techniques discussed in class. A less elegant approach might use the fact that Binet's formula is a closed-form solution for the n-th Fibonacci number, so that $F(n) = \frac{\varphi^n - \psi^n}{\sqrt{5}}$, where $\varphi = \frac{1+\sqrt{5}}{2} \approx 1.618$ is the golden ratio and $\psi = \frac{1-\sqrt{5}}{2} \approx -0.618$ is its conjugate.

Problem 2. (20 points) You are a chef preparing meals for a food delivery service. You have n different recipes to choose from, so that each recipe $i \in [n]$ can earn p_i dollars in profit and requires t_i time to prepare. You can prepare each recipe as many times as you want, as long as the total time does not exceed T. You goal is to find the maximum profit you can earn in T minutes.

Formally, the input is:

- T: The available time you have (in minutes).
- n: The number of available recipes.
- $t_i \geq 0$: The time the i-th recipe takes to prepare, in minutes ($1 \leq i \leq n$).
- $p_i \geq 0$: The profit of the i-th item, in dollars ($1 \leq i \leq n$).

(a) (10 points) Let $S(t)$ be the maximum profit that can be obtained in t minutes. You may assume $t \geq 0$ is an integer and $t \geq \max_{i\in[n]} t_i$. What is the recurrence relation for $S(t)$? Justify your answer.

$$S(T) = \begin{cases} 0, & t < \min_{i\in[n]} t_i, \\ \max_{i\in[n]} \big(S(t - t_i) + p_i\big), & t \geq \max_{i\in[n]} t_i\,. \end{cases}$$

I first think of the last recipe prepared (recipe i). Then before cooking that recipe, you had $t - t_i$ minutes left. So, the best profit in that time is $S(t - t_i)$. After cooking that recipe, you add its profit p_i. In this situation any recipe could be the last recipe so we check all possible last recipes and see the profit in each case. Then we keep the maximum profit. If we run out of time to cook any recipe, out profit is zero (base case).

(b) (10 points) Consider an asymptotically optimal dynamic programming algorithm to compute $S(T)$, the maximum profit that you can make. What is the runtime of the algorithm? Justify your answer.

$\Theta(nT)$

For each time t form 1 to T we need to examine all the recipes to compute the maximum. Checking each recipe takes $O(1)$, so computing one $S(t)$ costs $O(n)$. Now we multiple by the number of subproblems which is T getting $\Theta(nT)$.

Problem 3. (15 points)

A robot starts at the origin $(0, 0)$ on a Cartesian plane. The robot can only move either:

- One unit to the right $(+1, 0)$, denoted `R`, or
- One unit up $(0, +1)$, denoted `U`.

Given a destination point (x, y) where x, y are non-negative integers, determine the number of distinct paths the robot can take to reach (x, y) from $(0, 0)$.

Example: If $(x, y) = (2, 2)$, the possible paths are:

`RRUU, RURU, RUUR, URRU, URUR, UURR`

Thus, there are 6 possible ways.

(a) (5 points) Let $N(x, y)$ be the number of distinct paths the robot can take to reach (x, y) from $(0, 0)$. Write the recurrence relationship for $N(x, y)$, including the base cases.

$$N(x,y) = \begin{cases} 1 & \text{if } x = 0 \text{ or } y = 0, \\ N(x-1,y) + N(x,y-1) & \text{if } x > 0 \text{ and } y > 0. \end{cases}$$

(b) (5 points) What is the runtime of an asymptotically optimal dynamic programming implementation of an algorithm to solve the robot counting problem, i.e., an algorithm that outputs $N(x, y)$ for integer inputs $x, y \geq 0$?

$\Theta(xy)$

(c) (5 points) Using your recurrence relationship, compute $N(6, 4)$, the number of distinct paths the robot can take to reach $(6, 4)$ from $(0, 0)$. Show your work.

$$\text{Base cases: } N(x,0) = 1,\ N(0,y) = 1$$

$$N(1,1) = N(0,1) + N(1,0) = 1 + 1 = 2$$
$$N(1,2) = N(0,2) + N(1,1) = 1 + 2 = 3$$
$$N(1,3) = N(0,3) + N(1,2) = 1 + 3 = 4$$
$$N(1,4) = N(0,4) + N(1,3) = 1 + 4 = 5$$

$$N(2,1) = N(1,1) + N(2,0) = 2 + 1 = 3$$
$$N(2,2) = N(1,2) + N(2,1) = 3 + 3 = 6$$
$$N(2,3) = N(1,3) + N(2,2) = 4 + 6 = 10$$
$$N(2,4) = N(1,4) + N(2,3) = 5 + 10 = 15$$

$$N(3,1) = N(2,1) + N(3,0) = 3 + 1 = 4$$
$$N(3,2) = N(2,2) + N(3,1) = 6 + 4 = 10$$
$$N(3,3) = N(2,3) + N(3,2) = 10 + 10 = 20$$
$$N(3,4) = N(2,4) + N(3,3) = 15 + 20 = 35$$

$$N(4,1) = N(3,1) + N(4,0) = 4 + 1 = 5$$
$$N(4,2) = N(3,2) + N(4,1) = 10 + 5 = 15$$
$$N(4,3) = N(3,3) + N(4,2) = 20 + 15 = 35$$
$$N(4,4) = N(3,4) + N(4,3) = 35 + 35 = 70$$

$$N(5,1) = N(4,1) + N(5,0) = 5 + 1 = 6$$
$$N(5,2) = N(4,2) + N(5,1) = 15 + 6 = 21$$
$$N(5,3) = N(4,3) + N(5,2) = 35 + 21 = 56$$
$$N(5,4) = N(4,4) + N(5,3) = 70 + 56 = 126$$

$$N(6,1) = N(5,1) + N(6,0) = 6 + 1 = 7$$
$$N(6,2) = N(5,2) + N(6,1) = 21 + 7 = 28$$
$$N(6,3) = N(5,3) + N(6,2) = 56 + 28 = 84$$
$$N(6,4) = N(5,4) + N(6,3) = 126 + 84 = 210$$

$$N(6,4) = 210$$

$x\backslash y$	0	1	2	3	4
0	1	1	1	1	1
1	1	2	3	4	5
2	1	3	6	10	15
3	1	4	10	20	35
4	1	5	15	35	70
5	1	6	21	56	126
6	1	7	28	84	210