COMPSCI 320SC 2025 Midterm Test

Put the answers in the space below the questions. Write clearly and *show all your work*!

Marks for each question are shown below and just before each answer area.

This 60 minute test is worth 10% of your final grade for the course.

Question #:	1	2	3	4	Total
Possible marks:	5	5	5	5	20
Awarded marks:					

1. Recall the *coin change problem*: Given a set of coins of denominations $1 \leq c_1 < c_2 < \ldots < c_n$ and the amount x, what is the smallest number of coins needed to pay x? In the lecture, we considered a *greedy* algorithm for this problem which sequentially takes the largest possible coin.

   (a) Suppose $c_1 = 1, c_2 = 5, c_3 = 7, c_4 = 11$. Find a value x for which the greedy algorithm is not optimal. Write down how many coins of each domination the greedy algorithm uses, and give example of an optimal solution. ***(1 mark)***

   There are many such values. For example, when $x = 10$ the greedy algorithm uses coins $7, 1, 1, 1$, whereas the optimal solution is $5, 5$.

   (b) Suppose $c_i = 2^{i-1}$ for every $i \in \{1, \ldots, n\}$. Prove that the greedy algorithm is optimal. ***(4 marks)***

   We prove that the greedy algorithm is optimal by induction on x. For $x = 1$, both the greedy and the optimal solution use coin $c_1 = 1$.
   Suppose that the greedy algorithm is optimal whenever $x \leq k$, for some integer $k \geq 1$. We show that it is then also optimal when $x = k + 1$. Let $j \in \{1, \ldots, n\}$ be the largest index such that $c_j \leq x$. That is, c_j is the first coin chosen by the greedy algorithm. If we can show that an optimal solution also uses coin c_j, then we are done as by the induction the greedy solution is optimal for $x - c_j$.
   Note that an optimal solution uses each coin $c_1, \ldots, c_{n-1}$ at most once. Indeed, if an optimal solution would use some c_i twice, for $i \in \{1, \ldots, n-1\}$, then we could get a better solution by replacing $2 \times c_i$ with c_{i+1} (since $2 \cdot c_i = c_{i+1}$). Suppose, towards a contradiction, that an optimal solution does not use the coin c_j. Then it also cannot use any coin larger than c_j, as c_j is the largest coin which is at most x. That means the whole amount x has to be paid using $c_1, \ldots, c_{j-1}$, and each at most once, which is not possible as $c_1 + \ldots + c_{j-1} < c_j \leq x$.

2. You are given n jobs, labelled $1, \ldots, n$. Each job i is specified by two nonnegative integers (c_i, b_i):

   - c_i is the cost you must pay to complete job i.
   - b_i is the minimum amount of money you must have on hand before you can start job i.

   Assume that $b_i \geq c_i$ for every $i \in \{1, \ldots, n\}$.

   The goal is to determine the smallest initial amount of money you need in order to complete all n jobs, where you are free to choose the order in which the jobs are performed.

   (a) Suppose we perform the jobs in the decreasing order of b_i. Give an example where such an ordering requires larger amount of money than an optimal one. You need to justify your answer. ***(2 marks)***

   Suppose we have two jobs, $(5, 10)$ and $(1, 9)$. Let x be the amount of money we have at the beginning. Then we need $x \geq 10$ in order to start the first job, and $x - 5 \geq 9$ in order to start the second. This implies we need $x = 14$. However, if we first do the second job and then the first one, we need $x \geq 9$ to start the second job and then $x - 1 \geq 10$ to start the first one. This results in $x = 11$, which is better.

   (b) Suppose we perform the jobs in the increasing order of c_i. Give an example where such an ordering requires larger amount of money than an optimal one. You need to justify your answer. ***(2 marks)***

   Suppose we have two jobs, $(1, 1)$ and $(5, 10)$. Let x be the amount of money we have at the beginning. Then we need $x \geq 1$ in order to start the first job, and $x - 1 \geq 10$ in order to start the second. This implies we need $x = 11$. However, if we first do the second job and then the first one, we need $x \geq 10$ to start the second job and $x - 5 \geq 1$ to start the first one. This results in $x = 10$, which is better.

   (c) Describe how to obtain an optimal ordering of jobs. You do not need to prove that it is optimal. ***(1 mark)***

   Do the jobs in the decreasing order of $b_i - c_i$.

3. Consider the following divide-and-conquer "algorithm":

```
function printer(int n)
    if n == 1 return
    printer(⌊n/2⌋)
    for j = 1 to 4 do
        printline "hello world"
```

Let $T(n)$ denote the number of lines of output generated by a call of printer(n).

(a) Provide a recurrence equation for $T(n)$. ***(2 marks)***

$$T(n) = \begin{cases} 0 & \text{if } n = 1 \\ T(\lfloor n/2 \rfloor) + 4 & \text{otherwise} \end{cases}$$

(b) Solve the recurrence exactly for $n = 2^k$, being a power of 2. ***(2 marks)***

$$T(2^k) = 4 \cdot \lg n$$

(c) Recall the Master Divide-and-Conquer Theorem.

Assume $t(n) = a \cdot t(n/b) + g(n)$, where $g(n) \in \Theta(n^c)$, is the total time for a divide-and-conquer algorithm. Then:

$$t(n) \in \begin{cases} \Theta(n^c) & \text{if } a < b^c \\ \Theta(n^c \log n) & \text{if } a = b^c \\ \Theta(n^{\log_b a}) & \text{if } a > b^c \end{cases}$$

Apply this theorem (show variable values) for this algorithm printer(n). ***(1 mark)***

$a = 1$, $b = 2$, and $c = 0$. So $\Theta(\log n)$.

4. Okar is a notoriously clever burglar of houses. He realizes that to avoid being caught by the police, he can only steal from houses that are at distance greater than two from each other on a street. (We assume if he robs a house at address x, then cannot burgle a house at addresses $x-2$, $x-1$, $x+1$ and $x+2$ of the road).

   Input is a list of house (positive) content values, $v_1, v_2, \ldots, v_n$, starting at address 1 and ending at address n. Okar want to maximize the total value of homes that they burgle, without getting caught.

   (a) Consider the following input. What is best total value looted? What are the possible optimal set of houses that achieve this value?

Address:	1	2	3	4	5	6	7	8	9	10
Values:	7	1	2	9	3	1	4	7	9	5

(2 marks)

25 = 7+9+4+5 $\{v_1, v_4, v_7, v_{10}\}$ = 7+9+9 $\{v_1, v_4, v_9\}$

   (b) Describe in detail a dynamic programming algorithm to obtain the maximum value.

(3 marks)

Define the subproblem $M(i)$ to be the best total for houses in the range 1 to i, $1 \leq i \leq n$. with house i being robbed.
Clearly $M(1) = v_1$, $M(2) = v_2$, and $M(3) = v_3$
$M(i) = v_i + \max_{j<i-2} M(j)$
To get final answer we take $\max_{i=1}^{n} M(i)$.