CSE 100: Algorithm Design and Analysis
Midterm 3

Spring 2023

Note: This is a **closed** book examination. You have 75 minutes to answer as many questions as possible. The number in the square bracket at the beginning of each question indicates the number of points given to the question. Write all of your answers directly on this paper.

Please make your answers concise as well as precise. If there is something in the question that you believe is open to interpretation, then please go ahead and interpret, but state your assumptions in your answer. Or you can ask the TAs proctoring the exam for clarification. There is no penalty for attempting to answer a question with a wrong answer, so please answer as many questions as you can. Partial credit will be given even if the answer is not fully correct. If you run out of space you can ask for extra papers. Please write down your name on *every* page as some papers may fall off although it rarely happens. 9 pages in total including this cover. The maximum points you can earn is 135.

Problem	Points earned	Out of
1		8
2		8
3		14
4		14
5		14
6		14
7		14
8		14
9 (Bonus)		35
Sum		Max 135

Name _____

1. (8 points). If the statement is correct, choose 'true,' otherwise 'false.'

   (a) (2 points) A fixed-length code is prefix-free.
   True False **Sol.** True

   (b) (2 points) In the Interval Selection Problem, suppose intervals have weights and our goal is to find a subset of non-overlapping intervals whose total weight is maximized. The Earliest Ending algorithm finds the optimum solution.
   True False **Sol.** False

   (c) (2 points) In the tree representation of an optimal prefix-free code, there is no node that has exactly one child.
   True False **Sol.** True

   (d) (2 points) The Huffman code algorithm can be implemented using priority queue. If we use binary heap to implement a min-priority queue, the Huffman code algorithm can be implemented in $O(n \log n)$ time.
   True False **Sol.** True

2. (a) (4 points) In the Interval Selection Problem (See problem 4), we learned that some greedy algorithms are not optimal. Give an example that shows that the algorithm that repeatedly selects the shortest interval (and discards the intersecting intervals) is not optimal. **Sol.** See lecture slides. e.g. (1, 10), (11, 20), (8, 12)

   (b) (4 points) Suppose we use the following code for compression: $a : 0$, $b : 1$, $c : 01$, $d : 10$. What goes wrong? **Sol.** Creates ambiguities.

3. (14 points) Recall that the Fibonacci sequence is 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, $\cdots$. Formally, $f(0) = 0, f(1) = 1$ and $f(i) = f(i-1) + f(i-2)$ for all $i \geq 2$, where $f(i)$ denotes the ith Fibonacci number.

   Below is a pseudo-code of bottom-up DP for computing the nth Fibonacci number. Give a *pseudo-code* for top-down with memoization. If you don't know how, you can describe it in words—but you will then earn at most 6 points.

```
bottom up:
int F(n)
  Array A[0 ... n]
  A[0] = 0, A[1] = 1
  for i = 2; i <= n ; i++
     A[i] = A[i-1] + A[i-2]
  return A[i]

top down:
```

   Sol. See discussion ch15

4. (14 points) Interval Selection Problem (ISP). In the ISP, we are given n intervals $(s_1, f_1), (s_2, f_2), ..., (s_n, f_n)$, and are asked to find a largest subset of intervals that are mutually disjoint. For simplicity, let's assume that all s, f values are distinct. We refer to s_i and f_i as interval i's start and finish times, respectively. Also assume that intervals are ordered in increasing order of their finish times. Prove that there exists an optimal solution that includes the first interval, that is, the interval that ends the earliest.

   Sol. See lecture slides

5. (14 points) In the rod cutting problem, we are given as input, $p[0], p[1], ..., p[n]$, where $p[i]$ denotes the price of a rod/piece of length i. We are interested in cutting a given rod of length n into pieces of integer lengths in order to maximize the revenue. The followings are pseudocodes of the bottom-up DP for the problem. The first pseudocode correctly computes the maximum revenue one can get out of rod of length n, $r[n]$. You're asked to *modify/complete the first pseudocode and give the second pseudocode*, so it prints out an optimum cut. To get all points, you should give pseudocodes. You can choose to describe the changes in words (as long as your description is precise), but you will only get partial points. If you're not comfortable with the notation below, you can define your own—you can even write everything from scratch if it's easier for you.

```
EXTENDED-BOTTOM-UP-CUT-ROD(p, n)
1. Let r[0 .. n] and s[0 .. n] be new arrays
2. r[0] = 0
3. for j = 1 to n
4.    q = - infinity
5.    for i = 1 to j
6.         if q < p[i] + r[j-i]
7.              q = q[i] + r[j-i]
8.
9     r[j] = q
10. return r

PRINT-CUT-ROD-SOLUTION(p, n)
```

Sol. See lecture slides.

6. (14 points) In the Matrix chain multiplication problem, we are given as input a sequence of n matrices, $A_1, A_2, ..., A_n$ where A_i is $p_{i-1} \times p_i$. We would like to fully parenthesize the product $A_1 A_2 \cdots A_n$ such that the number of multiplications is minimized. For simplicity, let us assume that we are only interested in the minimum number of multiplications needed to compute $A_1 A_2 \cdots A_n$.

   We observed the following recursion, where $m[i, j]$ denotes the min number of multiplications needed to compute $A_{i...j} := A_i A_{i+1} \cdots A_j$.

$$m[i,j] = \begin{cases} \min_{i \le k \le j-1} m[i,k] + m[k+1,j] + p_{i-1} p_k p_j & \text{if } i < j \\ 0 & \text{if } i = j \end{cases}$$

   Sol. See lecture slides.

   (a) (6 points) Describe an algorithm that output the minimum number of multiplications needed to compute $A_1 A_2 \cdots A_n$. Your algorithm must be a *bottom-up* DP. Make sure that you include how to set up the DP table entries, in which order you compute the entries, and which value is returned as the optimum.

   (b) Analyze the (asymptotic) running time:

      i. (3 points) What is the number of the DP entries/subproblems?

      ii. (3 points) What is the running time for computing each entry?

      iii. (2 points) What is the running time for computing the optimum?

7. (14 points) Huffman Code. Suppose we have a text consisting only of a, b, c, d, e, f where each character appears with the following frequency:

a	b	c	d	e	f	total
6	18	8	2	5	11	50

(a) (10 points) Show the code built by the Huffman algorithm, *both as a tree and as a list* (character, codeword). When combining two trees, *the tree with lowest root frequency becomes the left child* and the tree with the second-lowest root frequency becomes the right child. Left children are associated with the bit 0, right children with the bit 1.

Sol. a:100
b:11
c: 00
d: 1010
e: 1011
f: 01

Tree: omitted

(b) (2 points) How many bits are required to represent the input using this Huffman code? (If you don't want to do calculation, you just need to show the process, e.g. X * Y + ...). **Sol.** 3 * 6 + 2 * 18 + 2 * 8 + 4 * 2 + 4* 5 + 2 * 11 = 120

(c) (2 points) How many bits are required to represent the input using a fixed-length code? Of course, you want to use as few bits as possible.

Sol. 50 * 3 = 150

8. (14 points) LCS DP table. In the LCS problem, we are given as input two sequences, $X = \langle x_1, x_2, \ldots, x_m \rangle$ and $Y = \langle y_1, y_2, \ldots, y_n \rangle$ and would like to find a longest subsequence common to both. Towards this end, we defined $c[i, j]$:= length of LCS of X_i and Y_j, where $X_i := \langle x_1, x_2, ...x_i \rangle$ and $Y_j := \langle y_1, y_2, ...y_j \rangle$. *Fill out the the following empty LCS DP table of entries $c[i, j]$ and give an LCS between X and Y.*

	j	0	1	2	3	4	5
i		y_j	C	B	B	D	A
0	x_i						
1	C						
2	B						
3	A						
4	C						

Sol. 000000
011111
012222
012223
012223

An LCS between X and Y: CBA

9. (Bonus; this will be graded rigorously) (35 points) [Rod-Cutting: Every unit-piece is unique] In the rod cutting problem we learned in the class, all unit pieces of the given rod were assumed to be uniform. That is, each piece's price was only based on its length. However, it is not the case here. So, we're given a rod of length n where the unit pieces are indexed by $1, 2, ...n$ from the left to the right. Let $c[i, j]$ denote a piece consisting of i, $i + 1$, ... jth unit pieces. As input, we're given n, and $p[i, j]$, which denotes the price of $c[i, j]$, for all $1 \leq i \leq j \leq n$. Here, $p[i, j]$ could be negative. So, you may want to discard some pieces. *Give an algorithm* that *computes the maximum revenue* you can obtain by cutting the given rod and selling (some of) them and *analyze the running time*. Cutting is assume to be free.

   e.g. Suppose $n = 3$, and $p[1, 1] = 3, p[2, 2] = -1, p[3, 3] = 4, p[1, 2] = 2, p[2, 3] = 3, p[1, 3] = 6$. Then, we can cut the rod into three unit pieces. By selling only the first and the third, we can get the max revenue of 7.

This page is intentionally left blank.