

**Dalhousie University**  
**CSCI 3110 — Design and Analysis of Algorithms I**  
**Fall 2014**  
**Final Examination**  
**December 5**  
**3:30pm-6:30pm**

Student Name:	_____
Student ID Number:	_____
FCS Username (CSID):	_____
Signature:	_____

Instructions (Read Carefully):

1. Aids allowed: one 8.5" by 11" piece of paper with anything written or printed on it (both sides). No textbooks, computers, calculators, or other aids.
2. This exam booklet has 11 pages, including this page. Ensure that you have a complete paper.
3. Understanding the exam questions is part of the exam. Therefore, questions will **not** be interpreted. Proctors will confirm or deny errors or ambiguities only. If you are unsure of your own understanding, clearly state reasonable assumptions that will not trivialize the questions.
4. The blank sheet given with this exam book is a piece of scratch paper. Do not submit.

Question	1	2	3	4	5	6	7	8	9	Total
Marks	15	12	20	13	10	7	8	7	8	100
Scores										
Marker										

1. [15 marks] True-false: 3 marks each. No justification necessary.

(a)  $\lg n = O(n)$ .

(b) Dijkstra's algorithm returns false when the input graph has a negative weight cycle reachable from the source vertex.

(c) Kruskal's algorithm does not work when some edges of the graph have negative weights.

(d) If  $f(n) = \Theta(g(n))$  then  $g(n) = \Theta(f(n))$ .

(e)  $P \subseteq NP$ .

2. [12 marks] Multiple-choice — **no justification necessary**. Circle the *single* best answer. 4 marks each.

(a) What is the relationship between  $f(n) = 2^{(\lg n)/2}$  and  $g(n) = n/\lg n$ ?

- a.  $f(n) = O(g(n))$
- b.  $f(n) = \Theta(g(n))$
- c.  $f(n) = \omega(g(n))$
- d.  $f(n) = \Omega(g(n))$

(b) Which of the following four problems is NP-complete?

- a. The maximum subarray problem
- b. The subset-sum problem
- c. The longest common subsequence problem
- d. The single-source shortest path problem

(c) In class, we learned a randomized algorithm for the selection problem, and proved that its expected running time is  $O(n)$ . What does this result imply?

- a. There is no  $O(n)$ -time deterministic algorithm for this problem
- b. The running time of this algorithm is  $O(n)$  only when the input distribution is uniform
- c. Any deterministic algorithm that solves this problem requires sorting
- d. The running time of this algorithm, when computed as an expectation over the distribution of the random number generator used in this algorithm, is  $O(n)$

3. [20 marks] (Order of Growth)

(a) [4 marks] Complete the formal definition of  $\Theta(g(n))$  by filling in the blank below:

$\Theta(g(n)) = \{f(n) : \text{there exist positive constants } c_1, c_2 \text{ and } n_0 \text{ such that}$

\_\_\_\_\_ }

(b) [4 marks] Prove that  $100 \lg n + 100\sqrt{n} = o(n)$ .

(c) [12 marks] For each of the following recurrences, use the “master theorem” and give the solution using big- $\Theta$  notation. Simply write down your answer and no justification is necessary.

If the “master theorem” does not apply to a recurrence, indicate this, but you need not show your reasoning or give a solution.

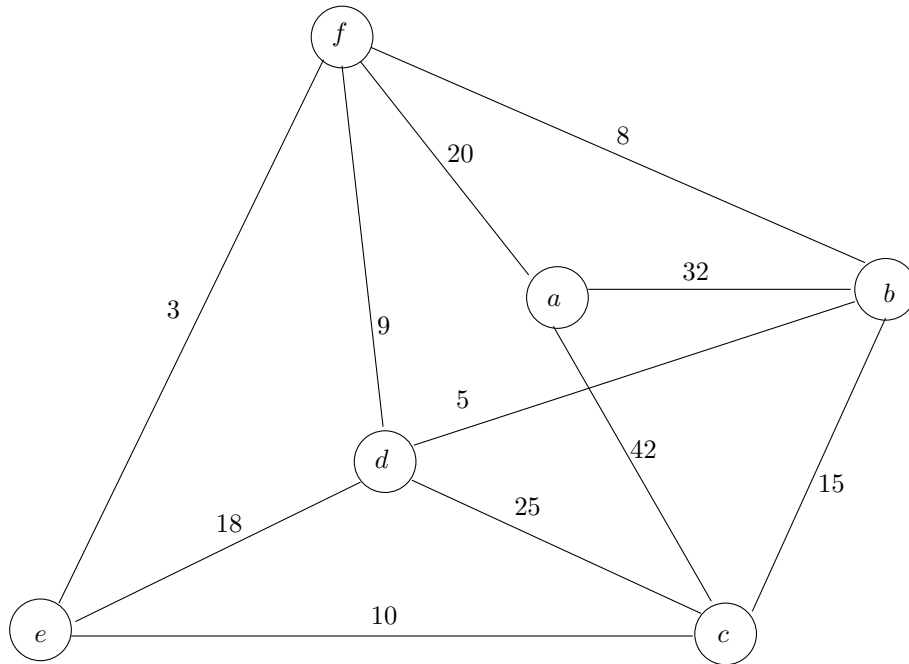
–  $T(n) = 2T(n/2) + \Theta(n^3)$

–  $T(n) = 4T(\lceil n/2 \rceil) + n^2$

–  $T(n) = 9T(n/3) + \Theta(n^2/\lg n)$

–  $T(n) = 2T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n$

4. [13 marks] Consider the weighted, connected and undirected graph  $G$  below:



- (a) [3 marks] The number of edges in a minimum spanning tree of  $G$  is (just write down the answer, no justification is needed):

- (b) [5 marks] Write down the edges of a minimum spanning tree of  $G$  constructed using Kruskal's algorithm, *in the order that they are selected by Kruskal's algorithm*, in the line below.

You can use  $(a,b)$  to denote an edge between vertices  $a$  and  $b$ .

You need not actually draw the tree or show your steps.

---

- (c) [5 marks] Choose  $a$  as the *starting vertex* for Prim's algorithm, and write down the edges of a minimum spanning tree of  $G$  constructed using Prim's algorithm, *in the order that they are selected by Prim's algorithm*, in the line below.

Use  $(a,b)$  to denote an edge between vertices  $a$  and  $b$ .

You need not actually draw the tree or show your steps.

---

5. [10 marks] (Computational Problems and Complexity)

(a) [3 marks] What is the main difference between the complexity classes NP-hard and NP-complete?

(b) [3 marks] Consider the following decision problem:

Given three integers  $u$ ,  $v$  and  $w$ . Is there a common divisor of  $u$  and  $v$  that is greater than  $w$ ?

Is this problem in the complexity class  $P$ ? Justify your answer.

(b) [4 marks] Let  $A$ ,  $B$  and  $C$  be three decision problems. Use the definition of polynomial-time reduction to prove that if  $A \leq_P B$  and  $B \leq_P C$ , then  $A \leq_P C$ .

6. [7 marks] Suppose that Alice has been using a communication channel to transmit messages to Bob. However, for some reason, this channel becomes noisy. As a result, some characters in the message could be lost during the transmission, so that only a selected portion of the original message is successfully transmitted.

For example, Alice sent the following message to Bob:

WHO IS HARRY POTTER'S BEST FRIEND?

As some character are lost during transmission, Bob could receive the following message:

HO HARY POT'S BST END

We can model this phenomenon as a problem for strings. Given a string  $X[1..n]$ , we say that string  $Y[1..m]$  is a *subsequence* of  $X$  if there are a set of indices  $\{i_1, i_2, \dots, i_k\}$ , such that  $y_1 = x_{i_1}, y_2 = x_{i_2}, \dots, y_k = x_{i_k}$ , and  $i_j < i_{j+1}$  for  $j = 1, 2, \dots, k - 1$ . That is, we follow the definition of subsequence given in class. Then, the message that Bob received from Alice is a subsequence of the message that Alice sent.

In a case of transmission via a noisy channel, it could be useful to know if the received message is indeed a subsequence of the message sent. Therefore, give an  $O(m+n)$ -time algorithm for determining whether a given string,  $Y[1..m]$ , is a subsequence of a given string,  $X[1..n]$ . Describe your algorithm in English. You can provide pseudocode if it helps you describe your algorithm, but it is not required. Analyze the running time of your algorithm.

You are NOT required to prove the correctness of your algorithm.

Hint: design a greedy algorithm.

7. [8 marks] Assume that you are given an array  $A[1..n]$  of distinct elements. A *local minimum* of  $A$  is defined as follows:

- $A[1]$  is a local minimum if it is smaller than  $A[2]$ ;
- $A[n]$  is a local minimum if it is smaller than  $A[n - 1]$ ;
- An element  $A[i]$  with  $1 < i < n$  is a local minimum if it is smaller than both  $A[i - 1]$  and  $A[i + 1]$ .

Thus an array may have one or more local minimum elements. Your task is to design an algorithm that can find one local minimum of  $A$ .

(a) [3 marks] Consider an algorithm that scans the array from left to right, while doing some comparisons until it sees a local minimum. Give an array of  $n$  elements that will make this algorithm scan the entire array.

(b) [2 marks] Describe in concise English how to solve the above problem efficiently using divide-and-conquer in  $O(\lg n)$  time. You need not give pseudocode, but you can if it helps with your explanation.

(c) [2 marks] Justify the correctness of your algorithm briefly.

(d) [1 marks] Analyze the running time of your algorithm.



8. [7 marks] Suppose that you are hired by a company, which set up its computer network a year ago to link together its  $n$  offices spread across the globe. You have reviewed the work done at that time, and noted that they modeled their network as a connected, undirected graph,  $G$ , with  $n$  vertices, one for each office, and  $m$  edges, one for each possible connection. Each edge is assigned a weight, which was equal to the annual rent that it costs to use that edge for communication purposes. Then they computed a minimum spanning tree,  $T$ , for  $G$  to decide which of the  $m$  edges in  $G$  to lease.

Now it is time to renew the leases for connecting the vertices in  $G$ . You noticed that the rent for each possible connection is unchanged, with the exception of one connection used in  $T$ , which has been increased. That is, in the graph, one edge,  $e$ , in  $T$  now has a greater weight than before, while the weights of all other edges remain unchanged.

Your task is to design an  $O(n+m)$ -time algorithm to update  $T$ , to find a new minimum spanning tree  $T'$  for  $G$  given the change in weight for the edge  $e$ .

- (a) [2 marks] Describe in concise English how to solve the above problem. You need not give pseudocode, but you can if it helps with your explanation.

- (b) [3 marks] Justify the correctness of your algorithm.

- (c) [2 marks] Analyze the running time of your algorithm.

9. [8 marks] Given an array  $A[1..n]$  of integers, the *maximum product sum* is the largest sum that can be formed by multiplying adjacent elements in the array. Each element can be matched with at most one of its neighbours.

For example, if  $A = \{1, 2, 3, 1\}$ , then the maximum product sum is  $1 + (2 \times 3) + 1 = 8$ . If  $A = \{2, 2, 1, 3, 2, 1, 2, 2, 1, 2\}$ , then the maximum product sum is  $(2 \times 2) + 1 + (3 \times 2) + 1 + (2 \times 2) + 1 + 2 = 19$ .

More precisely, for a set of array indices  $S = \{i_1, i_2, \dots, i_k\}$ , with  $1 \leq i_1 < i_2 < \dots < i_k < n$ , and  $i_j - i_{j-1} > 1$  for any  $j \in [2, k]$ , the *product sum* is defined as  $\sum_{i \in S} (A[i]A[i+1]) + \sum_{i \notin S} A[i]$ . Then, the maximum product sum is the largest among product sums for all possible  $S$ .

- (a) [2 marks] Compute the maximum product sum of  $A = \{1, 4, 3, 2, 3, 4, 2\}$ .

- (b) [6 marks] Describe a dynamic programming algorithm that computes the maximum product sum for any given array  $A$ .

Provide an English description AND pseudocode.

Analyze the running time of your algorithm.

No justification of correctness is required.

*An extra page for your solution to Question 9.*