COMPSCI 320SC 2021 Midterm Test

University ID: _____

Student Name: _____

Student Signature: _____

Time Finished: _____

Attempt *all* questions by yourself.

Write clearly and *show all your work*!

Marks for each question are shown after each part of the questions.

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

An extra 30 minutes for typesetting/scanning is added to Canvas for uploading PDF answers.

Make sure your Name and UPI is clearly displayed on your submitted PDF. (This is inside the rendered document, not the filename!)

Good luck!

Question #:	1	2	3	Total
Possible marks:	10	10	10	30
Awarded marks:				

1. Stable matching problem of n men and n women **[10 marks]**

(a) As similar to the setting of Assignment 1, we consider odd integers as man ID and even integers as woman ID. Consider the following tables as preferences of men and women. What is the man-optimal stable matching provided by the Gale-Shapley algorithm? How many proposals does the algorithm use?

Table 1: Men's preferences

1	2	4	6	8
3	4	6	2	8
5	6	2	4	8
7	2	4	6	8

Table 2: Women's preferences

2	3	5	7	1
4	5	7	1	3
6	7	1	3	5
8	1	3	5	7

(***4 marks***)

The stable matching is: 1 - 8; 3 - 2; 5 - 4; 7 - 6 **(3 marks)**
If one pair is wrong then deduct 1 mark, 2 pairs are wrong, then deduct 2 marks.
There are 13 proposals which is the worst-case input. **(1 marks)**

(b) To find a stable matching, the Gale-Shapley algorithm has to implement the method $prefer(w, m, m')$ that checks whether a woman w prefers a man m to the current partner m'. A simple way of implementing $prefer(w, m, m')$ is called $index(w, m, m')$ that scans the preference of w to find the index of m and m'. If m appears earlier in the list than m', return True; otherwise, return False.

It is easy to see that $index(w, m, m')$ runs in $O(n)$ time. It is suggested that for every woman w we create the inverse of preference list of men to speed up the Gale-Shapley algorithm. However, creating the inverse of a preference list takes $O(n)$ time, which is similar to $index(w, m, m')$.

Explain why the trick of using inverse of preference list for each woman can improve the running time of Gale-Shapley algorithm. You will get no marks for writing down a big-Oh expression with no explanation.

(***2 marks***)

Since we create the preference list for n women before running the algorithm, the complexity of creating n inversed preference is $O(n^2)$, which is similar to the worst case number of proposals. **(1 marks)**
After creating the inversed preferences, the cost of checking $prefer(w, m, m')$ is $O(1)$, which makes GS algorithm running in $O(n^2)$ time. **(1 marks)**

(c) Given a setting of the stable matching problem where there is a man m who is the first choice of *all* women. Prove or give a counterexample: In any stable matching, m must be matched with his first choice.

(***4 marks***)

Assume that the preference of m is $\{w_1, w_2, \ldots\}$. If m is paired with w_2, then the pair $m - w_1$ is unstable since m prefers w_1 to w_2 and w_1 prefers m to her partner. Therefore, m must be paired with his first choice.

2. Greedy Algorithms **[10 marks]**

Consider the undirected and weighted graph in Figure 1 below. On the undirected graph, the graph can be traversed in either directions. For example, you can traverse from Node 1 to Node 2 and from Node 2 to Node 1 with the same weight of 3.

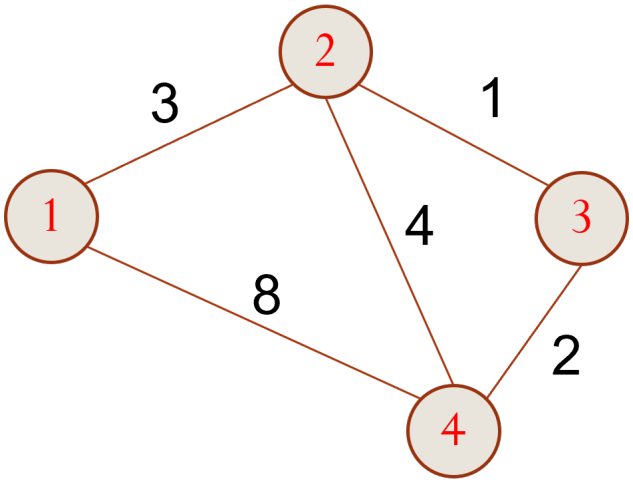

Figure 1: Graph 1.

(a) Draw the shortest path from Node 1 to Node 4 using the Dijkstra's algorithm on Graph 1. List the order in which nodes are added to the set of explored nodes. (***2 marks***)

The shortest path from Node 1 to Node 4 including the nodes in order is: $1 - 2 - 3 - 4$ with weight of 6. A correct shortest path with correct orders is enough for full mark.

(b) Draw the minimum spanning tree (MST) using the Prim's algorithm on Graph 1, starting at Node 1. List the order in which edges are added to the tree. (***2 marks***)

From 1, we take edge $1 - 2$.
From 1, 2, we take edge $2 - 3$.
From 1, 2, 3, we take edge $3 - 4$.
We stop algorithm.
The MST is $1 - 2 - 3 - 4$ with weight of 6.
A correct MST with correct edge orders is enough for full mark.

(c) On Graph 1, we observe that we can find the shortest path from any node to another node using only the MST. Is this observation correct for any undirected and weighted graph? Explain your answer. You will get no marks without any explanation.

(***6 marks***)

The statement is wrong. For example, consider the same graph with the weight 1 - 4 is 5.
The shortest path from 1 to 4 is 5 while it is 6 using the MST.
There is no partial mark if the explanation is not correct.

3. Divide and Conquer **[10 marks]**

(a) Master theorem: Consider the following "divide-and-conquer" function:

```
function printer(int n)
  for i = 1 to n do
    for j = 1 to ⌊√n⌋ do
      print "Hello Amigo"
    end for
  end for
  if n > 0 then
    printer(⌊2n/3⌋)
  end if
```

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

i. Provide a recurrence equation for $T(n)$. ***(3 marks)***

$$T(n) = \begin{cases} 0, & \text{if } n = 0 \\ T(\lfloor n/1.5 \rfloor) + \Theta(n^{1.5}) & \text{if } n > 0 \end{cases}$$

The first term can be replaced by $\lfloor 2n/3 \rfloor$.
The second term can be replaced by $\Theta(n\sqrt{n})$ or $O(n^{1.5})$.

ii. Solve the recurrence asymptotically for general n. ***(2 marks)***

Apply the master recurrence theorem with $a = 1$, $b = 1.5$, $c = 1.5$, we have $T(n) = \Theta(n^{1.5})$ or $O(n^{1.5})$.

You may want to make use of the following *master recurrence theorem*: Assume $T(n) = aT(n/b)+g(n)$, where $g(n)$ is $\Theta(n^c)$, is the total time for a divide and conquer algorithm. Then:

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

(b) Counting number of medians in an sorted array

A median item in an array (with duplicates) of n elements is the item x that is ranked $\lfloor n/2 \rfloor$ in a sorted version of the array, where items are ranked 1 to n. Given a sorted array, we want to count the number of occurences of the median in the array. Examples are given below.

Input: [3, 4, 4, 4, 5, 6, 7]
Ouput: 3
Input: [3, 3, 4, 4, 4, 4, 7]
Ouput: 4
Input: [3, 3, 3, 4, 6, 6, 6, 6]
Ouput: 1

Describe a divide-and-conquer algorithm that solves this problem in $O(\log n)$ time where n is the size of the array. (***5 marks***)

Given a sorted array $A[1...n]$, let $t = \lfloor n/2 \rfloor$.
We do a binary search in $A[1..t]$ finding the lowest index i of $A[t]$ and another binary search in $A[t..n]$ finding the highest index j of $A[t]$.
The answer/output is $j - i + 1$. As both binary searches run in $O(\log n)$ the total running time is $O(\log n)$.