CS 231: Fundamentals of Algorithms **Fall 2021**

Assignment #2

Your Name:

Collaborators:

Total time spent:

Problem 1. [4 points]
Decide whether you think the following statements are true or false. If it is true, give a short explanation. If it is false, give a counterexample.

(1) In every instance of the Stable Matching Problem, there is a stable matching containing a pair (m, w) such that m is ranked first on the preference list of w and w is ranked first on the preference list of m. **(2)** Consider an instance of the Stable Matching Problem in which there exists a man m and a woman w such that m is ranked first on the preference list of w and w is ranked first on the preference list of m. Then in every stable matching S for this instance, the pair (m, w) belongs to S.

Problem 2. [4 points]
(a) Suppose algorithm A has worst-case running time $O(n)$ and algorithm B has worst-case running time $O(n^2)$. What can you say about the relative performance of the algorithms?
(b) Repeat part a for the worst-case running times $\Theta(n)$ and $\Theta(n^2)$.

Problem 3. [6 points]
Consider the different graph representations discussed in class, and fill out the table below. In the following table, show the worst-case running times for the best algorithms for implementing the given operations. Justify your answer by writing a 2 sentence (maximum) explanation of how the operation would be performed. **Note:** Be careful what data structures you will be using for each implementation.

Operation	Adjacency Matrix	Adjacency List
Add a vertex		
Remove a vertex		
Find a vertex		
Add an edge		
Remove an edge		
Find an edge		

Problem 4. [6 points]
Some friends of yours work on wireless networks, and they're currently studying the properties of a network of n mobile devices. As the devices move around (actually, as their human owners move around), they define a graph at any point in time as follows: there is a node representing each of the n devices, and there is an edge between device i and device j if the physical locations of i and j are no more than 500 meters apart. (If so, we say that i and j are "in range" of each other.)

They'd like it to be the case that the network of devices is connected at all times, and so they've constrained the motion of the devices to satisfy the following property: at all times, each device i is within 500 meters of at least $n/2$ of the other devices. (We'll assume n is an even number.)

What they'd like to know is: Does this property by itself guarantee that the network will remain connected?

Here's a concrete way to formulate the question as a claim about graphs.

Claim: *Let G be a graph on n nodes, where n is an even number. If every node of G has degree at least $n/2$, then G is connected.*

(a) Decide whether you think the claim is true or false.
(b) Give a counter-example if it is false, or write a clear proof on why it is true.

Problem 5. [8 points]
Consider a version of the stable matching problem where there are n men and n women as before. Assume each man ranks the women (and vice versa), but now we allow ties in the ranking. In other words, we could have a man that is indifferent two women $w1$ and $w2$, but prefers either of them over some other woman $w3$ (and vice versa). We say a woman w prefers man $m1$ to $m2$ if $m1$ is ranked higher on the w's preference list and $m1$ and $m2$ are not tied.

With indifferences in the rankings, there could be two natural notions for stability. And for each, we can ask about the existence of stable matchings, as follows.

(a) Strong Instability. A strong instability in a perfect matching, S, consists of a man m and a woman w, such that each of m and w prefers the other to their partner in S. **Does there always exist a perfect matching with no strong instability?** Either give an example of a set of men and women with preference lists for which every perfect matching has a strong instability; or give an algorithm that is guaranteed to find a perfect matching with no strong instability.

(b) Weak Instability. A weak instability in a perfect matching, S, consists of a man m and a woman w, such that their partners in S are w' and m', respectively, and one of the following holds:
* m prefers w to w', and w either prefers m to m' or is indifferent between these two choices; or
* w prefers m to m', and m either prefers w to w' or is indifferent between these two choices.

In other words, the pairing between m and w is either preferred by both, or preferred by one while the other is indifferent. **Does there always exist a perfect matching with no weak instability?** Either give an example of a set of men and women with preference lists for which every perfect matching has a weak instability; or give an algorithm that is guaranteed to find a perfect matching with no weak

Problem 6. [10 points - Proof Problem]
There's a natural intuition that two nodes that are far apart in a communication network—separated by many hops—have a more tenuous connection than two nodes that are close together. There are a number of algorithmic results that are based to some extent on different ways of making this notion precise. Here's one that involves the susceptibility of paths to the deletion of nodes.

Suppose that an n-node undirected graph $G = (V, E)$ contains two nodes s and t such that the distance between s and t is strictly greater than $n/2$.

(a) [Proof Problem, submit separately] Prove that there must exist some node v, not equal to either s or t, such that deleting v from G destroys all $s - t$ paths. (In other words, explain that the graph obtained from G by deleting v contains no path from s to t.)

(b) Give an algorithm with running time $O(m + n)$ to find such a node v.

Problem 7. [6 points]
Consider the BFS implementation on pp. 90-91. When creating the list $L[i + 1]$, the algorithm has the following line:

```
For each node u in L[i]
```

This line does not specify the order in which the nodes of $L[i]$ should be considered.

(a) Argue that the layer assignments are independent of the specific order chosen in this line of the algorithm. In other words, the set $L[0]; L[1]; ...$ do not depend on the order.

(b) Show by example that the BFS tree T returned by the algorithm can depend on the particular order chosen in this line.

Problem 8. [6 points] - Coding question
In order to asymptotically analyze the running time of an algorithm, one has to think about how the data will be represented and manipulated in an implementation of that algorithm, so as to bound the number of computational steps it takes. In this problem, you will use arrays and lists to implement the different high-level operations in the Gale-Shapley algorithm presented in Chapter 1, and shown in Figure 1.

```
Initially all m ∈ M and w ∈ W are free
While there is a man m who is free and hasn't proposed to
every woman
   Choose such a man m
   Let w be the highest-ranked woman in m's preference list
      to whom m has not yet proposed
   If w is free then
      (m, w) become engaged
   Else w is currently engaged to m'
      If w prefers m' to m then
         m remains free
      Else w prefers m to m'
         (m, w) become engaged
         m' becomes free
      Endif
   Endif
Endwhile
Return the set S of engaged pairs
```

Figure 1: The Gale-Shapley algorithm.

Write a Java function to implement the following Gale-Shapley algorithm pseudocode with a worst-case running time complexity of $O(n^2)$ that follows the input and output specifications below.

Your function should have the following definition:

```
static public int[] GaleShapley(int[][] m-pref, int[][] w-pref)
```

Input Types:
Two 2D arrays of Integer; one for the men's preferences and the other for women's preferences. You could define them as int[][] m-pref, int[][] w-pref.

Output Types:
The output format is a single Integer array; in which the value in index i represents the women ID that is matched with the man with ID i.

Notes:
Since we are programming, the first man is referred to as man 0, first woman is referred to as woman 0. Also, assume all inputs are valid, you don't have to verify them. In other words, length of m-pref == length of w-pref, and the preference subarrays contain every person from the opposite list once.

Example 1:
m-pref = [[0, 1, 2], [0, 2, 1], [2, 1, 0]]
w-pref = [[1, 2, 0], [2, 0, 1], [0, 1, 2]]
Outputs [1, 0, 2], representing man 0 with woman 1, man 1 with woman 0, man 2 with woman 2

Example 2:
m-pref = [[0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5]]
w-pref = [[0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5]]
Outputs [0, 1, 2, 3, 4, 5]