Name _____

CSCI 332, Fall 2024

Final Exam—Practice 1

Note that this exam has six sections. They are:

1. Stable Matching
2. Algorithm Analysis
3. Graph Algorithms
4. Greedy Algorithms
5. Divide and Conquer Algorithms (no problem on this practice)
6. Dynamic Programming Algorithms

You may use a 3x5 handwritten notecard of notes during the test but no other resources. If you need more space than what is given, develop your solution on scratch paper before copying your final answer to the exam paper.

Good luck!

Section 1 (Stable Matching)

Recall the *stable matching problem*: given n men, n women, and preference lists ranking each man for each woman and each woman for each man, find a matching that contains no unstable pairs.

1. (2 points) How many matchings (not necessarily stable) are there, as a function of n?

2. (5 points) Decide whether the following statement is true or false. If it is true, give an example. If it is false, give a short explanation why no such instance can exist.

   True or false? There is an instance of the Stable Matching Problem with a stable matching containing a pair (m, w) such that m is ranked last on the preference list of w and w is ranked first on the preference list of m.

3. (5 points) Consider the following preference lists for 4 men and 4 women.

m_1: w_3, w_4, w_2, w_1	w_1: m_1, m_3, m_2, m_4
m_2: w_1, w_4, w_2, w_3	w_2: m_4, m_3, m_2, m_1
m_3: w_2, w_3, w_1, w_4	w_3: m_2, m_1, m_4, m_3
m_4: w_3, w_1, w_2, w_4	w_4: m_3, m_4, m_1, m_2

   What is the outcome of the Gale-Shapley algorithm on this input? Assume that men propose to women.

4. (3 points) Given the above preference, give another stable matching that is not the same as the one you found above. Here is another copy of the preference lists in case it is helpful.

m_1: w_3, w_4, w_2, w_1	w_1: m_1, m_3, m_2, m_4
m_2: w_1, w_4, w_2, w_3	w_2: m_4, m_3, m_2, m_1
m_3: w_2, w_3, w_1, w_4	w_3: m_2, m_1, m_4, m_3
m_4: w_3, w_1, w_2, w_4	w_4: m_3, m_4, m_1, m_2

5. (5 points) Describe a worst-case input, in terms of runtime, for the Gale-Shapley algorithm.

6. (3 points) Give a function $f(n)$ such that the worst-case runtime of the Gale-Shapley algorithm on an input of size n is $\Theta(n)$.

7. (2 points) Describe a best-case input, in terms of runtime, for the Gale-Shapley algorithm.

Section 2 (Algorithm Analysis)

8. (6 points) For each of the following statements, circle T if it is true and F if it is false.

   - $\sqrt{n}$ is $O(\log n)$: T or F
   - n^2 is $\Omega(n^2)$: T or F
   - n^3 is $\Omega(n^2)$: T or F
   - $n \log n$ is $\Theta(n^2)$: T or F
   - There is an algorithm with worst-case runtime that is $O(n^2)$ and best-case runtime that is $O(n^3)$: T or F
   - There is an algorithm with worst-case runtime that is $\Theta(n^2)$ and best-case runtime that is $\Omega(n^3)$: T or F

9. (4 points) Suppose you have algorithms with the following runtimes. (Assume these are exact running times as a function of the input size n, not a asymptotic running times.) How much slower do these algorithms get when you double the input size? (You can find this by dividing the runtime on $2n$ by the runtime on n). You should simplify your answer as much as possible. If n values do not cancel, you can describe what the value approaches as n gets large.

   (a) $3n^2$

   (b) $\log n$

10. (3 points) In words, what is the definition of the worst-case runtime for an algorithm?

11. (3 points) Give a function $f(n)$ such that the worst-case runtime of the following algorithm is $\Theta(f(n))$. Recall that $\lfloor x \rfloor$ takes the *floor* of x, meaning that it rounds down to the nearest integer.

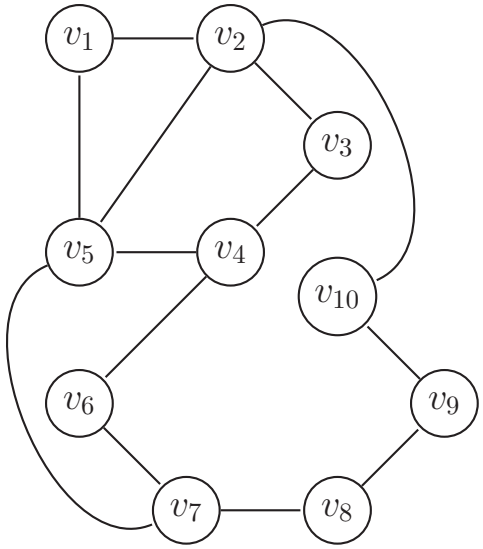
```
Algorithm-1(array A of length n):
  Result = 0
  For i in 1 to n:
      For j in i + ⌊n/2⌋ to n:
          Add A[j] to Result
  Return Result
```

12. (4 points) Give a function $f(n)$ such that the worst-case runtime of the following algorithm is $\Theta(f(n))$.

```
Algorithm-2(array A of length n):
  Result = 0
  For i in 1 to n:
      For j in 2^i to n:
          Add A[j] to Result
  Return Result
```

Section 3 (Graph Algorithms)

13. (3 points) Given the following undirected graph, give the nodes contained in the sets for each layer that would be generated by running breadth-first search starting at v_1. You may not need all of layers L_0 through L_6.



$L_0 =$

$L_1 =$

$L_2 =$

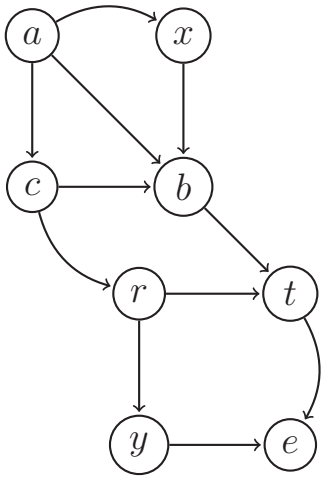
$L_3 =$

$L_4 =$

$L_5 =$

$L_6 =$

14. (3 points) Given the following directed graph, give a topological ordering of the nodes.



15. (3 points) Decide whether the following statement is true or false. If it is true, just write True. If it is false, give a counterexample.

True or false? Every connected tree has exactly $n - 1$ edges.

16. (3 points) Finish the following algorithm to find a topological ordering of a graph. Assume that the input is a directed graph with no cycles.

```
Topo-Order(length n list of nodes and length m list of edges):
  T = [], an empty list to hold the topological order of the nodes
  While there is a node that hasn't been added to T yet:
      Look through edge list to find v, a node with no incoming edges
```

17. (8 points) Fill in the blanks to complete the proof that in any binary tree the number of nodes with two children is exactly one less than the number of leaves. (Recall that a binary tree is a rooted tree where every node has no more than two children.)

Proof: Let T be a _____.

Inductive hypothesis: Suppose that every binary tree _____
_____
_____

There are two cases to consider:

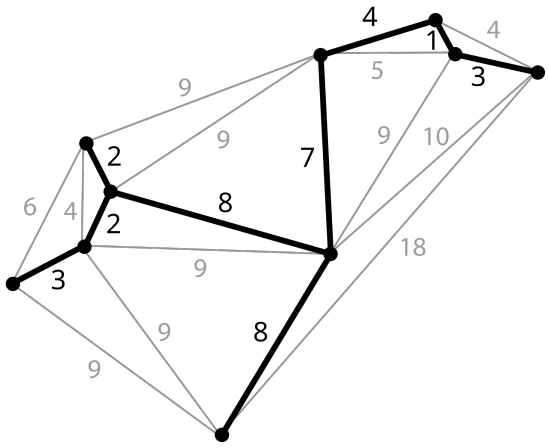
- T has 1 node. Then, _____
  _____
  _____.
- T has $n > 1$ nodes. In this case, if we remove a leaf node v and form T', we have formed a tree with fewer nodes than T in which each node has at most two children, and by the inductive hypothesis, the number of nodes with two children is exactly one less than the number of leaves. Now we must consider two cases:
  - Suppose v was the only child of its parent u in T. _____
    _____
    _____
    _____
    _____
    _____.
  - Suppose v was one of two children of its parent u in T. _____
    _____
    _____
    _____
    _____
    _____.

□

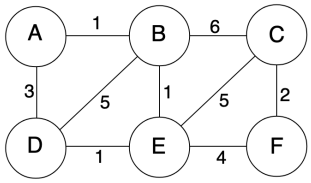
Section 4 (Greedy Algorithms)

In this problem, we will work on the *minimum spanning tree problem*. In this problem, we are given a connected graph with positive weights on the edges and we want to find the cheapest (lowest weight) set of edges that touches every node.

For example, in the graph below, the heavy edges form a minimum spanning tree of weight 38.



18. (5 points) What is the weight of the minimum spanning tree on this graph?



19. (5 points) A minimum spanning tree will never contain a cycle. Fill in the proof by contradiction of this fact. Hints: use the fact that every edge has positive weight. If you're totally lost, use the graph above to make an example to help you understand.

Proof: Let G be a connected graph and let T be a minimum spanning tree for G containing cycle with edges $e_1, e_2, \ldots, e_n, e_1$, each with weights $w_1, w_2, \ldots, w_n, w_1$, respectively.

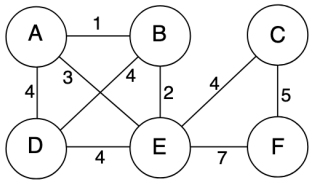
...a contradiction, meaning that the original assumption that T contained a cycle is false.

□

20. (5 points) Now that we know that minimum spanning trees never contain cycles, we are ready to give a greedy algorithm for the problem.

```
MST(graph G):
    T = [], an empty list to hold the edges in the minimum spanning tree
    While there is an edge that would not create a cycle when added to T:
        Choose the lowest-weight such edge and add it to T
    Return T
```

What order are the edges added to T for the following graph? (It may not be unique—just give one valid order).



21. (5 points) Give an example of a real-world problem that could be solved by computing a minimum spanning tree. (What do the nodes represent? What do the edges represent? What do the weights represent? How is a minimum spanning tree a solution?)

Section 5 (Divide and Conquer)

No problem on this practice test :)

Section 6 (Dynamic Programming)

You've been tasked with decorating the yard for the holidays. There are n trees indexed from 1 to n that may be decorated, and each tree i can hold d_i decorations. You like to put up as many decorations as possible. However, your landlord thinks that it is gaudy to decorate two adjacent trees, and so if you decorate tree i with d_i decorations, you cannot place any decorations on tree $i+1$. However, you may leave two adjacent trees un-decorated if you would like. Your goal is to design an algorithm to choose which trees to decorate in order to maximize the decorations that you place.

22. (4 points) Suppose you have 8 trees which can hold $d_1 = 5, d_2 = 8, d_3 = 9, d_4 = 12, d_5 = 7, d_6 = 4, d_7 = 2, d_8 = 10$ decorations. What is the maximum number of decorations you can place, and which trees should you decorate to do so?

23. (4 points) Give a recurrence relation for the maximum decorations using only trees up to tree i.

$$OPT(i) = \begin{cases} \_\_\_\_\_ & \text{if } i = 0 \\ \_\_\_\_\_ & \text{if } i = 1 \\ \_\_\_\_\_ & \text{if } i > 1. \end{cases}$$

24. (4 points) How total calls to OPT would be made to compute $OPT(3)$ for the input above? You may find it useful to draw the recursion tree but you are not required to do so.

25. (6 points) Give a polynomial time, recursive algorithm to compute the max decorations using your recurrence relation above. You should fill in the following two functions so that Max_Decorations returns the maximum number of decorations over any possible valid combinations of trees input.

Max_Decorations($d_1, d_2, \ldots, d_n$):

Compute_OPT(j):

26. (2 points) Analyze the runtime of your algorithm. That is, give a function $f(n)$ such that the worst-case runtime of Max_Decorations on an input of size n is $\Theta(f(n))$.