CS570 Summer 2025: Analysis of Algorithms   Exam I

	Points		Points
Problem 1	20	Problem 5	16
Problem 2	9	Problem 6	10
Problem 3	19	Problem 7	14
Problem 4	12		
			Total 100

First name	
Last Name	
Student ID	

Instructions:

1. This is a 2-hr exam. Closed book and notes. No electronic devices or internet access.
2. A single double sided 8.5in x 11in cheat sheet is allowed.
3. If a description to an algorithm or a proof is required, please limit your description or proof to within 150 words, preferably not exceeding the space allotted for that question.
4. No space other than the pages in the exam booklet will be scanned for grading.
5. If you require an additional page for a question, you can use the extra page provided within this booklet. However please indicate clearly that you are continuing the solution on the additional page.
6. Do not detach any sheets from the booklet.
7. If using a pencil to write the answers, make sure you apply enough pressure, so your answers are readable in the scanned copy of your exam.
8. Do not write your answers in cursive scripts.
9. This exam is printed double sided. Check and use the back of each page.

Student ID:

1) 20 pts (2 pts each)
Mark the following statements as **TRUE** or **FALSE** by circling the correct answer. No need to provide any justification.

[TRUE/FALSE]
If f(n) = Θ(g(n)) and h(n) = Θ(g(n)), then f(n) - h(n) = Θ(g(n))

[TRUE/FALSE]
In Prim's algorithm, the greedy choice is to pick the edge with the smallest weight that connects a visited node to an unvisited one.

[TRUE/FALSE]
A strongly connected directed graph with n vertices and n edges must contain at least one cycle.

[TRUE/FALSE]
Master Theorem can be applied to the following recurrence equation

$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + n$$

[TRUE/FALSE]
Every directed acyclic graph (DAG) has a unique topological ordering of its vertices.

[TRUE/FALSE]
If all edge weights in a connected, undirected graph are distinct, and the graph has at least one cycle, then the heaviest edge in the graph cannot be part of any minimum spanning tree.

[TRUE/FALSE]
In a binary heap, the *decrease-key* operation has a worst-case time complexity of $O(\log n)$

[TRUE/FALSE]
In the Gale-Shapley algorithm, it is possible for some men to end up with their best valid partners when women propose.

[TRUE/FALSE]
If f(n) = Ω(n log n) and g(n) = O(n^2 log n), then f(n) = O(g(n)).

[TRUE/FALSE]
Suppose that a DFS on a DAG G starting from node s succeeds in finding all nodes in G. Then s must be the first node in every topological ordering of G.

Student ID:

2) 9 pts **Select all correct answers!** 3 pts each. No partial credit

I. Give the tight upper bound for the worst-case run time of the following function using the big O notation:

```
int count(int n) {
  int iter = 0;
  for (int i = n; i > 0; i /= 2)
     for (int j = 0; j < i; j++)
        iter += 1;
  return iter;
}
```

A) O(log n)
B) O(n)
C) O(n logn)
D) O(n^2)

II. Consider the following four functions that map the positive integers to the reals:
$f_1(n) = n \log n$, $f_2(n) = n^{\log n}$, $f_3(n) = 2^n$, $f_4(n) = n^{3/2}$.
Which of the following correctly ranks them from slowest to fastest growth as $n \to \infty$?

(A) $f_1 \prec f_4 \prec f_2 \prec f_3$
(B) $f_1 \prec f_2 \prec f_4 \prec f_3$
(C) $f_4 \prec f_1 \prec f_2 \prec f_3$
(D) $f_2 \prec f_1 \prec f_4 \prec f_3$

III. Which of the following is/are true about the Gale-Shapley algorithm (with men proposing)?
(A) Men end up with their worst valid partners
(B) Women end up with their worst valid partners
(C) Men end up with their best valid partners
(D) Women end up with their best valid partners

Student ID:

3) 19 pts
Suppose you have participated in a bike race with an unusual format. It will take place in a road network represented as an undirected graph G = (V,E). The race starts at vertex *s* and ends at *t*, but there is no specified route - you may reach *t* from s by traversing any route in G. The length of any road(edge) *e* in G is given by l_e meters. You will start the race with your own bike which has a speed of X m/s. Here is an interesting twist - there are upgrade stations at a subset B of vertices. Each station *u* in B has a particular gear available that can be fit to your bike so that speed increases from X to $X + X_u$ (Note that each *u* may offer different type of gear and thus, different speedup X_u from it). You are allowed to add gear to your bike during the race from any ONE station if you wish, but fitting the gear adds a delay of T seconds (known constant).

You want to plan your race route to finish the race quickest.

a) Design an efficient algorithm to determine whether you should stop to get an upgrade and if so at which node. For full credit, your algorithm should not be asymptotically slower than Dijkstra's algorithm. (16 pts)

b) Provide the time complexity analysis for your algorithm. (3 pts)

Solution: Run dijkstra from s and from t. A route that gets a gear upgrade at any u in B would result in total time d(s,u)/X + d(u,t)/(X + Xu) + T. Compare for all u in B, as well as d(s,t) (i.e. no upgrade) and choose the least.

Student ID:

4) 12 pts (3 pts each)

Solve the following recurrences using the Master Method—if it applies—by giving tight theta-notation bounds in terms of n for sufficiently large n. Here, T(.) represents the running time of an algorithm. For each part below, briefly describe the steps and provide the final answer. If the Master Method does not apply, explain why.

a) $T(n) = 2\,T(n/2) + 1/n$

b) $T(n) = 1/4\ T(n/4) + 1/n$

c) $T(n) = 10^{100}\ T(n/2) + 2^n$

d) $T(n) = 64\ T(n/8) - n^2$

(a) $T(n) = \theta(n)$
(b) *Master theorem does not apply*
(c) $T(n) = \theta(f(n)) = \theta(2^n)$
(d) *Master theorem does not apply*

Student ID:

5) 16 pts
Imagine you're designing an electrical grid to connect several substations in a city. You are given a graph G where nodes represent substations, and edges represent power lines that can be built to connect pairs of substations, with each power line having an associated construction cost. Your primary goal is to connect all substations with the minimum total construction cost (forming a Minimum Spanning Tree, MST).

Now, consider there's one specific, crucial power line **e**. The city planners want to know: can this crucial line **e** be part of some electrical grid configuration that achieves the overall minimum construction cost?

Knowing that there are at most 10 cycles in G, describe a linear time algorithm (WRT the number of edges and nodes in G) to determine if a minimum cost grid connecting all substations can be built such that it contains line **e**.

Idea: We need to find out if e is a maximum cost edge in a cycle or not.

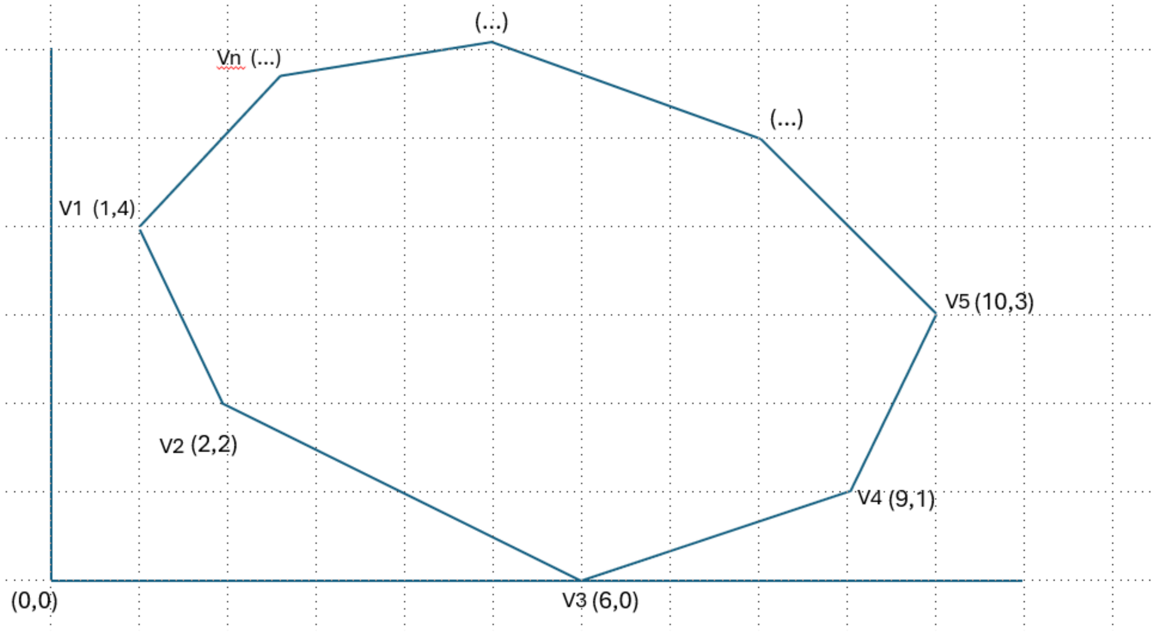
Step 1. Remove edge e going from v to u, and find a path from v to u on G-e. If there is no path from v to u then the answer is yes (STOP)
Step 2. If there is a path from v to u, then check to see if the cost of e is higher than the cost of all edges on the path found from v to u. If so, the answer is no (STOP), otherwise, remove the highest cost edge on the path found from v to u
Step 3. If answer is not determined repeat steps 1 and 2 at most 10 times

Run time: Each step 1 and 2 takes O(m+n) time. This is repeated at most 10 times. Overall time will be O(m+n).

Student ID:

6) 10 pts

A polygon is called *convex* if all its internal angles are less than 180° and none of the edges cross each other. We represent a convex polygon as an array V with n elements, where each element represents a vertex of the polygon in the form of a coordinate pair (x, y). We are told that $V[1]$ is the vertex with the least x coordinate and that the vertices $V[1], V[2], ..., V[n]$ are ordered counterclockwise.



Assuming that the x coordinates (and the y coordinates) of the vertices are all distinct, do the following:

a) Give a *divide and conquer* algorithm to find the vertex with the largest x coordinate in $O(\log n)$ time. (8 pts)
b) Justify the time complexity of your algorithm (2 points)

- Brief solution to the problem
- (a) Since $V[1]$ is known to be the vertex with the minimum x coordinate (leftmost point), moving counter-clockwise to complete a cycle must first increase the x coordinates and then, after reaching a maximum (rightmost point), should decrease the x coordinates back to that of $V[1]$. Thus, the array $V_x[1:n]$ (the x -coordinates of the vertices) forms a *unimodal* array, and the problem reduces to finding the **maximum** element in a **unimodal** array, which can be done in $O(\log n)$ time.

Algorithm

```
        1. If $n = 1$ , return $V_x[1]$ .
```

2. If $n = 2$, return $\{V_x[1], V_x[2]\}$.
3. Let $k \leftarrow \lfloor n/2 \rfloor$.
4. If $V_x[k] > V_x[k - 1]$ and $V_x[k] > V_x[k + 1]$, then return $V_x[k]$.
5. If $V_x[k] < V_x[k - 1]$, recursively call the algorithm on $V_x[1: k - 1]$;

otherwise, call it on $V_x[k + 1: n]$.

This binary search-based method ensures $O(\log n)$ time complexity.

The time complexity satisfies: $T(n) \leq T(n/2) + \Theta(1)$. By applying the Master Theorem, we obtain: $T(n) = O(n)$. Thus, we can find the vertex with the largest y -coordinate in $O(\log n)$ time using a divide and conquer approach.

Student ID:

7) 14 pts
A painter has n projects. Project i yields revenue r_i and requires h_i hours for full completion. Partial completion is allowed: working t hours on project i earns $\frac{r_i}{h_i} \times t$ revenue, up to a maximum of r_i. The painter may work fractional hours.

a) Given the target revenue R, design an $O(n \log n)$-time algorithm to choose which projects to work on and how many hours to allocate to each to accumulate at least R revenue in the minimum total time. (6 pts)
b) Prove the correctness of your algorithm. (6 points)
c) Justify the time complexity of your algorithm (2 points)

Solution:
A greedy approach is used to solve this problem. Specifically, the commissions are sorted in descending order of their "rate" $\rho_i = \frac{r_i}{h_i}$ (the revenue per hour ratio). Then, starting from the commission with the highest rate, you spend time on each commission until the required revenue R is reached. The last commission done might be only partially completed. The algorithm's time complexity is dominated by the sorting step, which is $O(n \log n)$.
Proof by Contradiction.
Assume that the greedy solution is not optimal, and there exists an optimal solution where some commission j with $\frac{r_j}{h_j} > \frac{r_k}{h_k}$ is only partially done or completely undone, while a lower-rate commission k is partially done or completed.
Since $\frac{r_j}{h_j} > \frac{r_k}{h_k}$, this implies that the time spent on commission k could have been better spent on commission j to earn more revenue in the same amount of time. By spending more time on commission j, you could earn revenue faster and reach the required revenue R in less time.
Thus, the assumption that the optimal solution does not include completing more of commission j is incorrect. This contradicts the assumption that the solution was optimal.
Rubrics:
Algorithm:
- x points: Sort the commissions according to their rates. (Or use heap to achieve similar results)
- x points: Start from the commission with the highest rate following the descending order. (Or use correct heap operations)

x points: Correct time complexity
Proof:
- x points: Correct non-optimality assumption
- x points: Correct contradiction analysis.

Student ID:

Additional Space

Student ID:

Additional Space