CS570 Spring 2025: Analysis of Algorithms Exam I

	Points		Points
Problem 1	20	Problem 5	15
Problem 2	9	Problem 6	15
Problem 3	15	Problem 7	16
Problem 4	10		
			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]

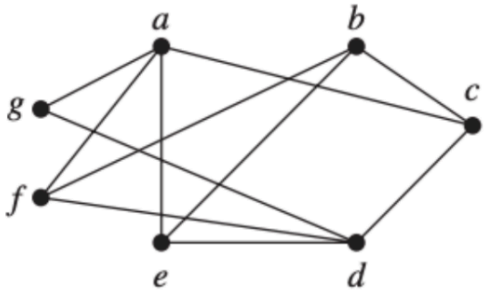
A Fibonacci Heap extract-min operation has a worst-case run time of O(log n).

[TRUE/FALSE]

Greedy algorithms make decisions to guarantee global optimization at each step.

[TRUE/FALSE]

The graph below is bipartite.



[TRUE/FALSE]

Dijkstra's algorithm will correctly find the shortest path in a graph containing negative edge weights, provided there are no negative cycles.

[TRUE/FALSE]

When performing DFS and BFS starting from a node u in a graph G, the number of levels in the DFS tree can never exceed the number of levels in the BFS tree.

[TRUE/FALSE]

If an edge is not part of any MST of G, then it must be the maximum weight edge on some cycle in G.

[TRUE/FALSE]

Suppose you have a directed graph G with a source vertex s, a target vertex t, and nonnegative edge weights. Furthermore, each edge has a distinct length that is a power of 2 (e.g., 1, 2, 4, 8, …, and no two edges share the same power of 2). Then the shortest s-t path (in terms of total length) is guaranteed to be unique

Continued on the next page

Student ID:

[TRUE/FALSE]
Any function that is Ω(log n) is also Ω(log log n).

[TRUE/FALSE]
We know that algorithm A has a worst-case running time of $O(nlog(n))$ and algorithm B has a worst-case running time of $O(n^2)$. It is possible for algorithm B to run faster than algorithm A on an input size n which is arbitrarily large.

[TRUE/FALSE]
When using the accounting method to find amortized costs for operations A, B, and C on data structure S, it is possible for all operations to end up with the same amortized cost even if their worst-case costs are different.

Student ID:

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

I. Which of the following algorithms use the divide and conquer approach?
A) Heap Sort
B) Merge Sort
C) Prim's Algorithm
D) Breadth-First Search (BFS)

II. Which of the following statements about binary min-heaps is incorrect?
A) They are always complete binary trees
B) Every parent node's key value is always less than its child nodes' key values
C) They can be used to implement priority queues
D) Merging two heaps of size n takes O(log n) worst-case time
E) DECREASE-KEY takes O(1) amortized time

III. Which of the following is true for asymptotic notations
A) The worst-case time complexity of Merge sort is $\Omega(\log n)$
B) If $f(n) = n \log(n^8)$ and $g(n) = \log n^n$ then $f(n) = O(g(n))$
C) If $f(n) = \mathbf{o}(g(n))$, then $g(n) = \Omega(f(n))$
D) If $f(n) = O(n^2)$ and $g(n) = O(n^2)$, Then $f(n)/g(n) = O(1)$

Student ID:

3) 15 pts

Solve the following recurrence equations 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, either use the Master Method if it directly applies and clearly show which case applies and why, or explain why the Master Method does not apply and solve using the brute force method.

(a) $T(n) = 4T(n/2) + n^2 \log n$

(b) $T(n) = T(n/2) + T(n/4) + n^2$

(c) $T(n) = 16T(n/2) + 2^n$

(a) Case # 2 f(n) grows faster by a logarithmic factor $T(n) = \theta(n^2 log^2(n))$

(2 points) - For using the correct case
(3 points) - For all the intermediate steps and the final answer

(b) $Master\ theoreom\ does\ not\ apply\ \theta(n^2)$

Cost of the operations at the first level of the recursion tree: cn^2

Cost of the operations at the second level: 5/16 cn^2

Cost of the operations at the 3rd level: 25/256 cn^2

At each level the cost goes down by a factor of 5/16

Total cost due to f(n) = cn^2 + 5/16 cn^2 + 25/256 cn^2 + … = $\theta(n^2)$

Total number of subproblems = O(n) (assuming T(n) = 2T(n/2))

Total run time = $\theta(n^2)$

(2 points) - For plotting the correct recurrence tree/showing cost of operations at each level.
(3 points) - For correct simplification and final answer.

(c) $T(n) = \theta(f(n)) = \theta(2^n)$ **Need to check the regularity condition**

f(n) is exponential and grows faster than n^4

regularity check:

f(n) = 2^n a.f(n/b) = $16.2^{n/2}$

For any non-trivial problem size n>8 a.f(n/b) is a fraction of f(n) (C=$8/2^{n/2}$)

So the regularity condition is met.

T(n) = $\theta(2^n)$

(2 points) - For using the correct case
(3 points) - For all the intermediate steps and the final answer

Student ID:

4) 10 pts

Suppose you have a stable matching problem with n men and n women. You know all the preferences using which you run the G-S algorithm (with men proposing) to compute a stable matching. You also note down the number of times each man is rejected. Later, when you want to use the stable matching you computed, you realize that it is lost, and you have also lost some of the women's preference lists. Thus, all you are left with is
1) all of the men's preference lists
2) some but not all of the women's preference lists
3) the no. of rejections (r_i for each man i)
Devise an algorithm (with explanation and time complexity) to find a stable matching from the information available, or prove that it is not possible to compute one using the information available.

Solution: In GS, men propose in the decreasing order of their respective preferences. Since any i was rejected r_i times, he was matched with the (r_i +1)'th woman on his preference list. All of this information is available and gives us a linear time algorithm. (We do not need any of the women's preference lists).

(2 points) - Stating or explaining that it is possible to find a stable matching and the required information is enough
(6 points) - Explain the algorithm or write the pseudo code
(2 points) - Prove or explain the time complexity to be linear

Student ID:

5) 15 pts

We have a list of N pets consisting of some dogs and others cats. We know the list has all dogs listed first, followed by all cats. You can make queries of the form "Is the n^{th} pet on the list a dog or a cat?" for any $n \leq N$. You want to determine how many dogs and how many cats are on the list. Devise an algorithm to do this by asking $\mathbf{o}(n)$ (little o) number of queries (along with time complexity analysis to show it has the desired efficiency). (Note that the simple algorithm making a pass over all pets would make $\boldsymbol{\theta}(n)$ queries, thus, not meeting the requirements).

Solution: Divide and Conquer. To get count(L) for no. of cats on any input list L, consider *equal* halves T and B. Make a query for the pet ranked |T|. If dog, output count(B). If cat, output count(T) + |B|. Base case for |L| = 1: just query that pet. (To be very precise, lists should be represented as their start/end index pair).
The recurrence we get is T(n) = T(n/2) + 1, and thus, T(n) = O(logn) by Master theorem. This is asymptotically strictly slower than linear as required.

Alternate perspective: Binary search. Instead of searching an 'element', we are searching the neighboring pair that is dog - cat. In a list of size n (indices 1 through n), if the pair is at (i,i+1), we have i dogs and n-i cats.
Recurrence and Complexity Analysis follows similarly.

(2 points) - Correctly dividing the problem into two *equal* sub-problems
(4 points) - Establishing the correct way to conquer each problem (recurse on only B or T depending on if middle element cat or dog)
(3 points) - Showing the correct combine step (return count(B) in case of middle element dog and count(T) + |B| if cat)
(2 points) - Establishing the base case.
(2 points) - Correct recurrence
(2 points) - Finding complexity from recurrence and mentioning that it satisfies the o(n) requirements.

Student ID:

6) 15 pts
Consider an undirected, connected, weighted graph G (with both positive and negative edge costs).

a) True or False: Kruskal's algorithm may not correctly find an MST in the presence of negative cost edges. You need to explain your answer.

False: Proof of correctness for Kruskal's stays valid even if the edge costs are negative.

(1 point) - Correctly stating the answer is False.
(3 points) - Correctly explaining that the answer is False.

Now, for the graph G as above, we define a spanning network as a connected graph that contains a subset of the edges of G and spans across all nodes in G (but is not necessarily a tree, distinguishing it from a spanning tree). A Minimum Spanning Network (MSN) is a spanning network of minimum cost.

b) True or False: Every MST in G is an MSN. You need to explain your answer.

False. G can have a cycle where all edges have negative costs. In that case all those edges will belong to the MSN but not to the MST.

(1 point) - Correctly stating the answer is False.
(4 points) - Correctly explaining that the answer is False.

c) How would you modify Kruskal's algorithm to find an MSN? No proof is necessary.

Algorithm 1: First add all negative cost edges to the MSN then proceed with positive cost edges as we do via Kruskal's to find an MST.

Alternative explanation: When going through edges in the non-decreasing order of their cost, discard an edge if 'it creates a cycle AND it has a positive cost', otherwise add it into the MSN.

Algorithm 2: Run Kruskal as it is and then add the remaining negative edges

Algorithm 3: Instead of checking for cycle:

- check for cycle if the edge is positive
- or skip for negative edges

- or something with the same implication!

(3 Points) - Using all the negative edges first
(3 Points) - Using positive edges to find MST

7) 16 pts
You will be on vacation and need to make sure your dog will be taken care of at all times while you are away. You have a number of dog sitters that have provided you with their availability schedule, i.e. the intervals of time that they will be available to take care of your dog. Each of the n intervals i has a start time S_i and an end time E_i. To minimize your dog's stress level you want to plan a schedule for your dog sitters in such a way that you minimize the number of times your dog sees one dog sitter going away and another dog sitter replacing them. Design an algorithm to schedule your dog sitters in a way that there is always a dog sitter staying with your dog and minimizes the number of dog sitter exchanges. Your algorithm should run in O($n$log$n$) in the worst case, and also be able to detect if such a schedule is not even feasible.
A) Describe your algorithm (6 pts)

Sort availability periods based on non-decreasing start times

Go through intervals in this sorted order finding interval i that starts before *the start of your vacation* and ends last (and ends after the start of your vacation).
If there is no such interval, there is no solution stop.
Otherwise Choose interval i
while E_i < end of your vacation AND i < n
    go through intervals in this sorted order finding interval j that starts before *Endtime* and ends last.
    If $E_j < E_i$ or no such interval exists, there is no solution. Stop
    choose interval j
    i = j
Endwhile
If E_i < end of your vacation, there is no solution. Stop
Otherwise print chosen intervals

(1 Point) - Correctly sorting availability periods

(3 Points) - Correctly showing the greedy step ie. finding interval i that starts before the start of your vacation and ends last

(2 Points) - Correctly stating the **if condition** in case solutions don't exist.

Student ID:

Parts B and C on the next page

B) Prove the correctness of your algorithm (7 pts)

Will prove by math induction that our intervals finish no sooner than the corresponding intervals in an optimal solution
Basis step: Our first interval is chosen such that it ends last among all intervals that start at or before our vacation start time.

Inductive step:
Hypothesis - Our kth interval finishes no sooner than the corresponding interval in the optimal solutions.

We will prove that our k+1st interval finishes no sooner than the corresponding interval in the optimal solution:
Assume that the k+1st interval in the optimal solution finishes later than ours. Since our solution finds the interval that has the latest finish time amongst intervals that overlap with the kth interval, and since our kth interval finishes no sooner than the kth interval in the optimal solution then it must be that the k+1st interval in the optimal solution has no overlap with the kth interval in the optimal solution. A contradiction.

Math. Inductions completed.
Now assume that the optimal solution has fewer intervals. Say our solution is of size k. We can easily see a contradiction since our k-1st interval did not cover the entire vacation period, and it finishes no sooner than the corresponding interval in the optimal solution, then the optimal solution will also need a kth interval to cover the entire vacation period.

(2 Points) - For stating the base case correctly
(2 Points) - For showing Inductive step correctly
(4 Points) - For Inductive step and proof

C) Analyze the complexity of your algorithm (3 pts)

Sorting takes constant time.
Inside the while loop, each interval is visited only once. So it takes linear time.

Student ID:

Overall complexity = O(nlogn)

(1 Point) - Correct time complexity
(2 Points) - Explanation

Additional Space

Student ID:

Student ID:

Additional Space