CS 170 Midterm 1

Write in the following boxes clearly and then double check.

Name :

SID :

Exam Room : ◯ Pimentel 1 ◯ Dwinelle 155 ◯ VLSB 2050 ◯ Other (Specify):

Name of student to your left :

Name of student to your right :

- **The exam will last 110 minutes.**
- The exam has 12 questions with a total of 110 points. You may be eligible to receive partial credit for your proof even if your algorithm is only partially correct or inefficient.
- Only your writing inside the answer boxes will be graded. **Anything outside the boxes will not be graded.** The last page is provided to you as a blank scratch page.
- Answer all questions. Read them carefully first. Not all parts of a problem are weighted equally.
- Be precise and concise.
- The problems may **not** necessarily follow the order of increasing difficulty.
- The points assigned to each problem are by no means an indication of the problem's difficulty.
- The boxes assigned to each problem are by no means an indication of the problem's difficulty.
- Unless the problem states otherwise, you should assume constant time arithmetic on real numbers. Unless the problem states otherwise, you should assume that graphs are simple.
- If you use any algorithm from lecture and textbook as a black box, you can rely on the correctness and time/space complexity of the quoted algorithm. If you modify an algorithm from textbook or lecture, you must explain the modifications precisely and clearly, and if asked for a proof of correctness, give one from scratch or give a modified version of the textbook proof of correctness.
- Assume the subparts of each question are **independent** unless otherwise stated.
- **Please write your SID on the top of each page.**
- Good luck!

1 Big-O (6 points)

Label each of the following statements as "True" or "False".

(a) $(\log\log n)^{100} = O(\sqrt[7]{\log(n)})$. ◯ True ◯ False

(b) $2^{\sqrt{\log n}} = O(n^{0.003})$. ◯ True ◯ False

(c) $2^{(\log n)^2} = O(n^{0.003})$. ◯ True ◯ False

(d) $2^{\sqrt{n}} = O(n^{\log n})$. ◯ True ◯ False

(e) If $T(n) = 2T(\frac{n}{2}) + O(\sqrt{n})$, then $T(n) = O(n)$. ◯ True ◯ False

(f) If $T(n) = 2T(\frac{n}{2}) + O(n)$, then $T(n) = \Theta(n \log n)$. ◯ True ◯ False

Exam continues on next page

2 Recurrences (5 points)

Consider the recurrence relationship $T(n) = 2T(\frac{n}{2}) + O(\frac{n}{\log n})$ with the base case $T(1) = 1$. Prove that $T(n) = O(n \log \log n)$ using the **tree method**.

Hint: You may use the fact that $\sum_{j=1}^{k} \frac{1}{j} = O(\log k)$.

3 Short Answers (10 points)

1. **Fill in True or False:** The following algorithm correctly computes single-source shortest paths in any graph with no negative-length cycles.

   ◯ True ◯ False

Algorithm 1 Input: Weighted Graph $G = (V, E)$; Source Vertex $s \in V$

$dist \leftarrow [+\infty, \ldots, +\infty]$
$dist[s] \leftarrow 0$ ▷ $dist$ is a length $|V|$ array with $+\infty$ everywhere except s
for $(u, v) \in E$ **do**
  for $i = 1, \ldots, |V| - 1$ **do**
    $dist[v] \leftarrow \min(dist[v], dist[u] + \ell(u, v))$

2. **Fill in True or False:** In an undirected graph, it is possible for two vertices with an edge between them to have the same parent in the DFS tree.

   ◯ True ◯ False

3. Recall the greedy algorithm we saw in class for solving the Horn-SAT problem. Suppose that we ran this algorithm on an instance of the Horn-SAT problem with variables a, b, c, d, e and it returned the following satisfying True/False assignment to these variables: $(a, b, c, d, e) = (T, F, F, T, T)$.

   For each of the following True/False assignments, state whether it is possible or impossible for that assignment to also satisfy the same Horn-SAT formula.

   (a) $(a, b, c, d, e) = (T, F, F, T, F)$ ◯ Possible ◯ Impossible

   (b) $(a, b, c, d, e) = (T, T, F, T, T)$ ◯ Possible ◯ Impossible

4. Suppose you have a set of 8 distinct numbers $a_1, ..., a_8$ such that the set $a_1^2, ..., a_8^2$ has size 4. Are the a_i's necessarily the 8-th roots of unity? If not, provide a counter-example.

   ◯ Yes ◯ No

5. Let $\mathsf{Roots}_4 := \{\omega_0, \omega_1, \omega_2, \omega_3\}$ be the 4-th roots of unity, where $\omega_0 = 1$ and the other three are numbered counter-clockwise on the unit circle. We saw in class that ω_1 is a *generator* of this set, meaning that for every a, there is an integer b such that $\omega_a = \omega_1^b$. Are any of the other 4th-roots of unity generators of Roots_4? If so, list them.

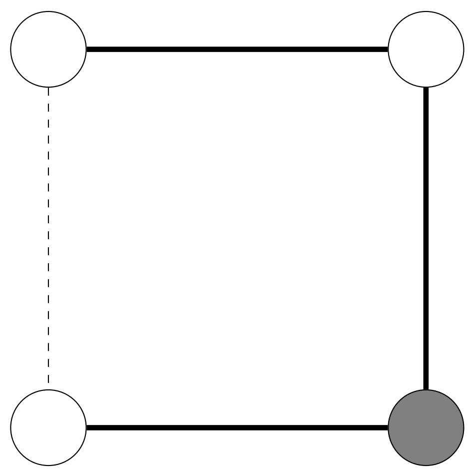
   ◯ Yes ◯ No

Exam continues on next page ⟹

4 Depth-First Search (10 points)

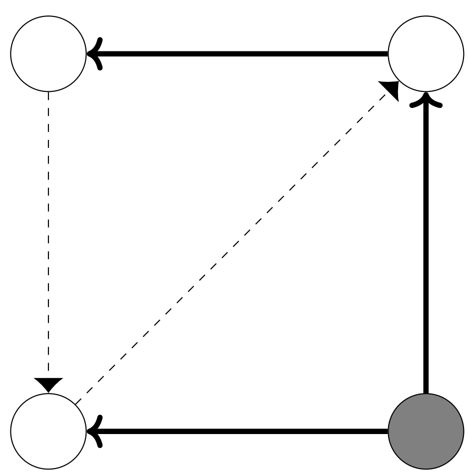
In the following graphs, the bold solid edges correspond to tree edges in the DFS traversal of the graph and the dashed edges are all of the other edges. The grey, filled-in vertex is the first node explored by the DFS. For each graph, state whether it is possible or impossible for the DFS algorithm to match such a traversal, given some tie-breaking rule between the vertices. Note that the first graph is undirected while the rest are directed.

1.



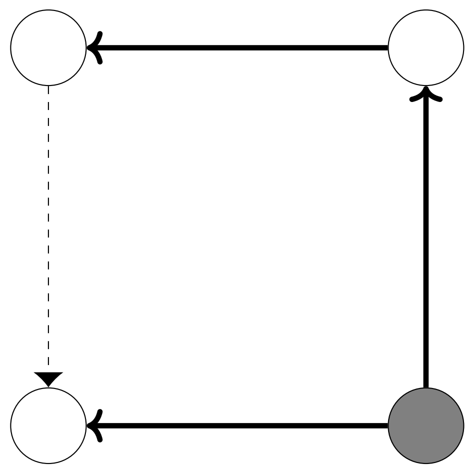
○ Possible ○ Impossible

2.

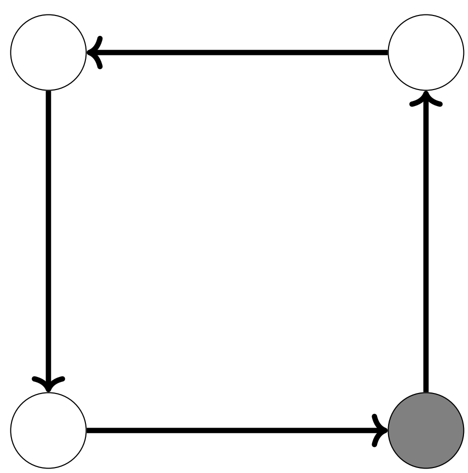


○ Possible ○ Impossible

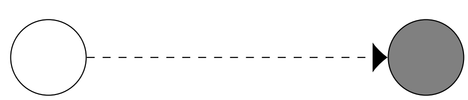
Exam continues on next page

3. 

○ Possible ○ Impossible

4. 

○ Possible ○ Impossible

5. 

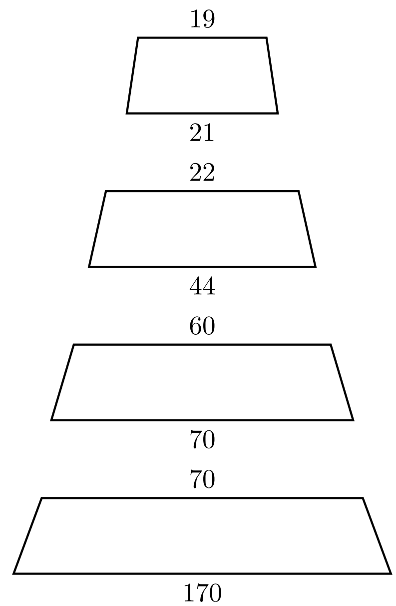
○ Possible ○ Impossible

Exam continues on next page

5 Spaceship (10 points)

To escape being turned into a paperclip by a rogue artificial intelligence, you are building a spaceship to Mars. You have n trapezoidal components available to build your spaceship. The i-th component is a trapezoid with height 1 and bases of length a_i and b_i such that $a_i < b_i$. A spaceship consists of a sequence of components of decreasing width stacked on top of each other. That is, the i-th component can go on top of the j-th component if and only if $b_i \leq a_j$. Design an efficient algorithm to output the height of the tallest spaceship you can make from the available components.

In the figure below, the spaceship has height 4. (Figure not to scale.)



Give a succinct and precise description of an algorithm to solve this problem. (Proof of correctness not required.) Your algorithm should run in time asymptotically faster than $O(n^3)$ to receive full credit.

Exam continues on next page

6 Karatsuba's (5 points)

Suppose we run Karatsuba's algorithm (using integers in base 10) on the two integers 5432 and 6789 in order to compute the product 5432×6789. Write down the 3 subproblems that Karatsuba's algorithm will call in its top level of recursion.

Subproblem 1: Run Karatsuba's algorithm on $\boxed{}$ and $\boxed{}$.

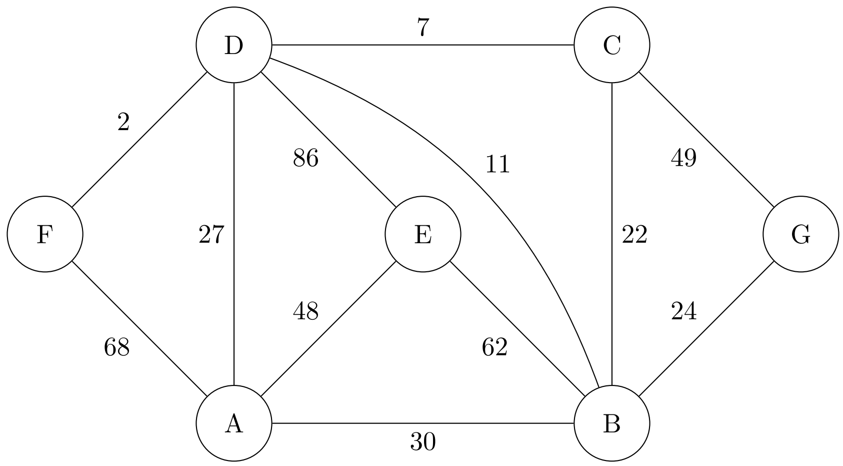
Subproblem 2: Run Karatsuba's algorithm on $\boxed{}$ and $\boxed{}$.

Subproblem 3: Run Karatsuba's algorithm on $\boxed{}$ and $\boxed{}$.

For each subproblem, write the smaller of the two numbers on the left. Order the three subproblems according to their leftmost number, from smallest to largest.

7 Kruskal's (6 points)

Run Kruskal's algorithm on the following graph. For each edge, indicate either the order it was added to the MST or if it's not in the MST. For example, if edge UV was the 3rd edge added to the MST, then bubble in "in MST" and write 3 in the corresponding order box. If an edge is not in the MST, bubble in "not in MST" and leave the order box blank.



Edge	In MST?	Order	Edge	In MST?	Order
AB	◯ in MST ◯ not in MST		AD	◯ in MST ◯ not in MST	
AE	◯ in MST ◯ not in MST		AF	◯ in MST ◯ not in MST	
BC	◯ in MST ◯ not in MST		BD	◯ in MST ◯ not in MST	
BE	◯ in MST ◯ not in MST		BG	◯ in MST ◯ not in MST	
CD	◯ in MST ◯ not in MST		CG	◯ in MST ◯ not in MST	
DE	◯ in MST ◯ not in MST		DF	◯ in MST ◯ not in MST	

8 Batched Selection (12 points)

The selection problem asks us to find the k-th largest element of an unsorted list of length n. Recall that QuickSelect solves this problem in expected $O(n)$ time. There is also a deterministic algorithm called DeterministicSelect that solves this problem in $O(n)$ time always. To answer m distinct selection queries for the k_1-th, k_2-th, …, k_m-th largest elements of a given list, one could run DeterministicSelect with $k = k_i$ for each i and take $O(nm)$ time. But we aspire for better!

Design an algorithm to answer m selection queries for distinct $k_1, \ldots, k_m$ on a list $[a_1, a_2, \ldots, a_n]$ of n distinct integers in $O(n \log m)$ time. Your algorithm may call DeterministicSelect as a subroutine. Give a succinct and precise description of your algorithm. (Proof of correctness not required.) Note: algorithms that run in $\Omega(n \log n)$ time will get no credit.

9 Birthday Party Attendance (12 points)

You invited n friends to your upcoming birthday party. Your i-th friend will show up with probability p_i (and does not show up with probability $1 - p_i$) independently of your other friends. Let random variable M be the number of friends who show up. To plan the party, you want to know the probability distribution of M. That is, you would like to compute $\mathbb{P}(M = m)$ for each $m \in \{0, 1, 2, \ldots, n\}$.

a) Describe how to deduce the probability distribution of M from the coefficients of the polynomial

$$P(x) = ((1 - p_1) + p_1 x)((1 - p_2) + p_2 x) \cdots ((1 - p_n) + p_n x).$$

Exam continues on next page

b) Describe an algorithm to compute the coefficients of $P(x)$ in $O(n(\log n)^2)$ time. As a reminder,

$$P(x) = ((1-p_1)+p_1x)((1-p_2)+p_2x)\cdots((1-p_n)+p_nx).$$

10 Unique Topological Sort (10 points)

Given a directed acyclic graph G with n vertices and m edges, design an algorithm that determines whether G has a unique topological sort.

a) Give a succinct and precise description of an algorithm to solve this problem. (Proof of correctness not required.) Full credit is given to algorithms that run in $O(n + m)$ time.

b) What is the runtime of your algorithm?

11 Min-max tree (12 points)

Let $G = (V, E)$ be an undirected graph with a weight $w_e \geq 0$ for each edge $e \in E$. Recall that a minimum spanning tree T of G is a spanning tree of minimum total weight, where the total weight of T is defined as

$$w(T) = \sum_{e \in T} w_e.$$

In this problem, we will also consider a different type of spanning tree, called a *min-max tree*. A min-max tree T' of G is a spanning tree of G which minimizes the quantity

$$\max(T') = \max_{e \in T'} \{w_e\},$$

among all spanning trees of G. In other words, among all spanning trees of G, the min-max tree T' has as small of a maximum edge as possible.

a) In this part, we will show that if T is a minimum spanning tree, then it is also a min-max tree. We will use a proof by contradiction. To do so, assume for the sake of contradiction that T is *not* a min-max tree of G. Then there is another spanning tree T' of G whose largest weight edge is smaller than T's largest weight edge.

Given T', it is possible to modify T to create a new spanning tree with smaller total weight than T. Give a succinct and precise explanation of how to do so. (Proof of correctness not required. Note that this contradicts the fact that T is a minimum spanning tree.)

b) In this part, you will show that the converse to the last part is not true: that is, a min-max tree is not necessarily a minimum spanning tree. To prove this, construct a counter-example on a graph G with at most 4 vertices, in which there exists a min-max tree that is not a minimum spanning tree. So you should construct such a graph G, give a min-max tree of G, and show that it is not a minimum spanning tree of G.

Give a concrete construction of G in the following box. Draw the vertices of the graph, the edges between them, and label the weights on the edges.

Give an example of a min-max tree T' of G in the following box. Draw the vertices and only the edges included in the min-max tree. Compute the total weight of the tree $w(T') = \sum_{e \in T'} w_e$.

Give a minimum spanning tree T of G with a total weight $w(T) = \sum_{e \in T} w_e$ which is less than that of the min-max tree you gave in the previous box.

12 Flights (12 points)

You are planning to travel by plane from San Francisco to New York. The airline you selected operates in n cities, indexed 1 through n, and has m distinct flight routes. The i-th route flies from city u_i to city v_i and has cost w_i.

Luckily for you, the airline is running a promotion for your trip where you may take up to k flights for free. Your goal is to compute the minimum total cost needed to travel from city 1 (San Francisco) to city n (New York), if you may take up to k flights for free. Give a succinct and precise description of an algorithm to solve this problem. (Proof of correctness not required.) Your algorithm should run in time $O(k(m+n)\log n)$ to receive full credit.

This page left intentionally blank
for scratch purposes.
This page **will not be graded**.

DO NOT DETACH THIS PAGE.