Midterm 1

Name:

SID:

Exam Room:

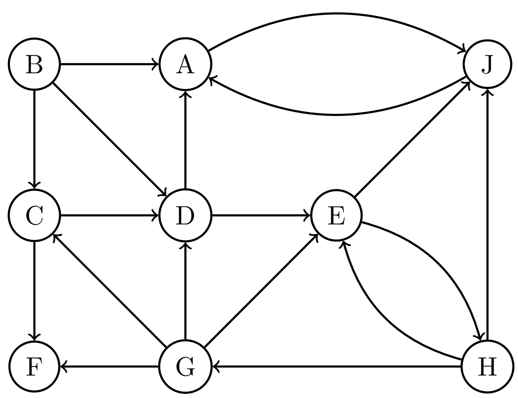
Name of student to your left:

Name of student to your right:

Do not turn this page until your instructor tells you to do so.

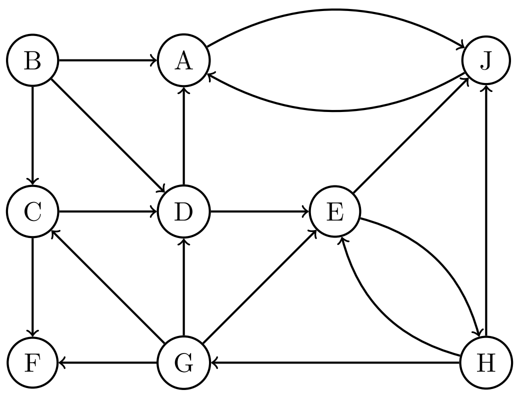
- After the exam starts, please *write your student ID (or name) on every odd-numbered page* (we will remove the staple when scanning your exam).
- For short question, your answers must be written clearly inside the box region. Any answer outside the box will not be graded. For longer question, if you run out of space, you must clearly mention in the space provided for the question if part of your answers is elsewhere.
- Try to answer all questions. Not all parts of a problem are weighted equally. Before you answer any question, read the problem carefully. Be precise and concise in your answers.
- You may use the blank page at the back for scratch work, but it will not be graded.
- You may consult only *one sheet of notes*. Apart from that, you may not look at books, notes, etc. Calculators, phones, computers, and other electronic devices are NOT permitted.
- There are **16** single sided pages on the exam. Notify a proctor immediately if a page is missing.
- **Any algorithm covered in the lecture can be used as a blackbox.**
- **You have 80 minutes: there are 11 questions on this exam worth a total of 90 points.**

1. **(4 points)** For the directed graph below, find all the strongly connected components and draw the DAG of strongly connected components. Label each strongly connected component with all the nodes it contains.



Draw the DAG in the box below:

2. **(8 points)** Execute DFS on the same graph (reproduced here for convenience) starting at node A and breaking ties alphabetically. Draw the DFS tree/forest. Mark the pre and post values of the nodes with numbering starting from 1.



Node	pre	post
A		
B		
C		
D		
E		
F		
G		
H		
J		

Draw the DFS Tree/Forest in the box below:

3. **(4 points)** In the DFS execution from above, mark the following edges as as **T** for Tree, **F** for Forward, **B** for Back and **C** for Cross. (No justification necessary)

Edge	Type
$G \to C$	
$A \to J$	
$B \to A$	
$B \to D$	

4. (a) **(4 points)** Draw a strongly connected graph with 6 vertices with the smallest possible number of edges in the box below.

   (b) **(2 points)** In general, the minimum number of edges in a strongly connected directed graph with n vertices is ______ . (no justification necessary)

5. **(6 points)** Suppose $G = (V, E)$ is an undirected graph with positive integer edge weights $\{w_e | e \in E\}$. We would like to find the shortest path between two vertices s and t with an additional requirement: if there are multiple shortest paths, we would like to find one that has the minimum number of edges.

   We would like to define new weights $\{w'_e | e \in E\}$ for the edges so that, a single execution of Dijkstra's algorithm on the graph G with new weights $\{w'_e\}$, starting from s finds the shortest path to t with this additional requirement.

   How should we set the new weights w'_e?

   $$w'_e = \boxed{}$$

   No justification necessary.

6. **(8 points)** Here is an implementation of Bellman-Ford algorithm:

`Input:` `Directed Graph` $G = (V, E)$, `with edge lengths` $\{\ell_e | e \in E\}$.
`Output:` `Compute distances` $dist(u)$ `to each vertex` u `from a start vertex` s.

It turns out that the runtime of the above algorithm can be very sensitive to the way in which vertices

for $i = 1$ to n **do**
  $dist(u) \leftarrow \infty$
  $prev(u) \leftarrow nil$
end for
$dist(s) \leftarrow 0$
$k \leftarrow 0$
repeat
  for $i = 1$ to n **do**
    for each directed edge (i, j) **do**
      update(i, j)
    end for
  end for
  $k \leftarrow k + 1$
until all $dist$ values stop changing OR $(k = n)$

Algorithm 1: Bellman-Ford Algorithm

in a graph are numbered. In other words, the runtime of the algorithm on the same graph can widely vary, if we change the numbering of the vertices.

Give one graph G on 11 vertices and two ways to label the vertices of G, such that in one labelling the algorithm makes 20 calls to update, while in the other labelling the algorithm terminates in 10^2 calls to update.

7. **(6 points)** We computed the minimum spanning tree T on a graph G with costs $\{c_e\}_{e \in E}$. Unfortunately, after computing the minimum spanning tree, we discover that the costs of all the edges in the graph have changed as follows: the new cost w_e are given by,

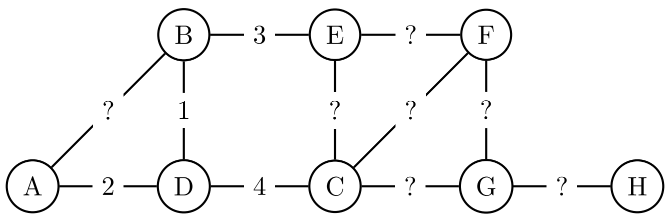
$$w_e = \begin{cases} 2 \cdot c_e & \text{if } c_e > 100 \\ 0 & \text{if } c_e \leq 100 \end{cases}$$

Is the tree T that we computed earlier, still a minimum spanning tree of the graph?

Write "yes" or "no":

If yes, prove; if no, disprove with a counterexample.

8. (**6 points**) In this graph, some of the edge weights are known, while the rest are unknown.



$$cost(A, D) = 2, cost(B, D) = 1, cost(C, D) = 4, cost(B, E) = 3$$

List all edges that **must** belong to a minimum spanning tree, regardless of what the unknown edge weights turn out to be. Justify each of your edges briefly (a sentence or less is enough).

Edges that must belong to every MST	Justification

SID:

9. Design an efficient algorithm for the following problem

`Input:` n `numbers` $\{a_1, \ldots, a_n\}$
`Goal: Compute the polynomial with` $a_1, \ldots, a_n$ `as its roots. In other words, compute coefficients` $b_0, \ldots, b_n$ `so that` $(x - a_1) \cdot (x - a_2) \cdots (x - a_n) = b_0 + b_1 x + \ldots b_n x^n$.

(Hint: Try divide and conquer & use $O(n \log n)$ time polynomial multiplication algorithm as a blackbox)

(a) **(10 points)** Pseudocode:

(b) **(3 points)** Write the recurrence for the running time of the algorithm in the box.

$T(n) =$

(c) **(6 points)** Solve the recurrence to compute the running time and put your answer in the box. Show your work below the box.

$T(n) = \boxed{}$

10. **(13 points)** You are given the road network $G = (V, E)$ of a country, and the lengths $\{\ell_e | e \in E\}$ of each road in the network.

Some of the cities have airports, while others don't. Let F be the subset of cities that have an airport in them.

Devise an algorithm to compute the distance from each city to the nearest airport. (Assume that the graph is directed and that all edge lengths are non-negative).

Remember every *correct* algorithm will receive a score depending on its runtime. (can you do it with the same run-time as Dijkstra's?).

(a) **Main Idea:** (try less than 6 sentences if you can, but don't fret if you go over)

(b) **Runtime of the algorithm** =

(c) **Proof of Correctness** (try less than 4 sentences if you can, but don't fret if you go over)

11. **(10 points)** Suppose you are given an array $A[1 \ldots n]$ of sorted integers that has been circularly shifted k positions to the right for some k. For example, $[35, 42, -5, 15, 27, 29]$ is a sorted array that has been circularly shifted $k = 2$ positions, while $[27, 29, 35, 42, -5, 15]$ has been shifted $k = 4$ positions. We can obviously find the largest element in A in $O(n)$ time.

    Assuming all the integers in the array are distinct, describe an $O(\log n)$ algorithm to find the largest element in A.

    Brief but precise description of the algorithm: (try less than 6 sentences if you can, but don't fret if you go over)

(Extra Page)

(Extra Page)

(Extra Page)