CS170 Final

Name:

SID:

GSI and section time:

Write down the names of the students on your left and right as they appear on their SID.

Name of student on your left:

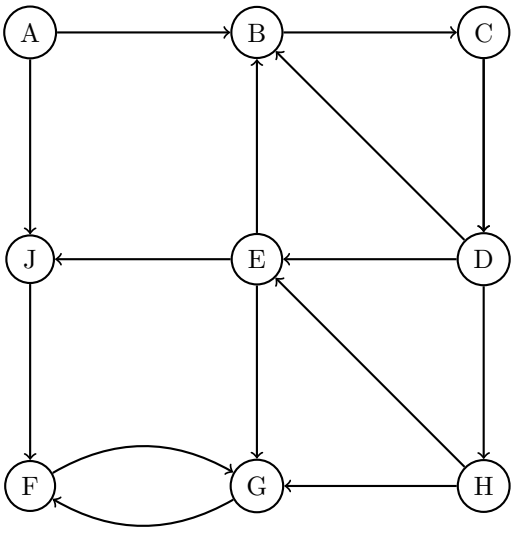
Name of student on your right:

Answer all questions. Read them carefully first. Be precise and concise. The number of points indicate the amount of time (in minutes) each problem is worth spending. Not all parts of a problem are weighted equally. Write in the space provided, and use the back of the page for scratch. By "reduction" in this exam it is always meant "polynomial-time reduction." Also, when you are asked to prove that a problem is **NP**-complete, no need to show that it is in **NP**, unless asked to do so.

Good luck!

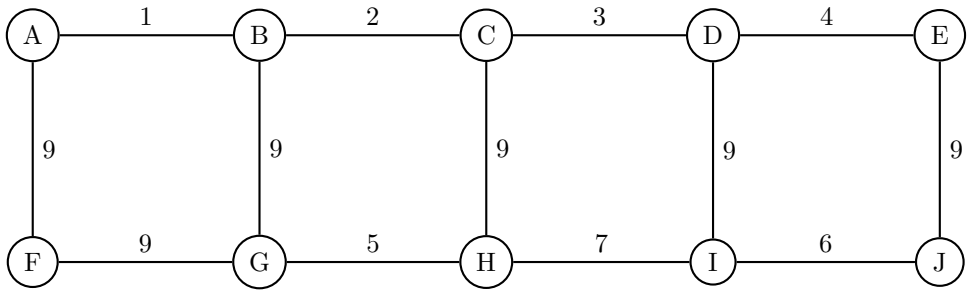
1. **Strongly Connected Components** ***(6 points)***

For the directed graph below, list the strongly connected components **in the order in which they are output by the strongly connected components algorithm.**



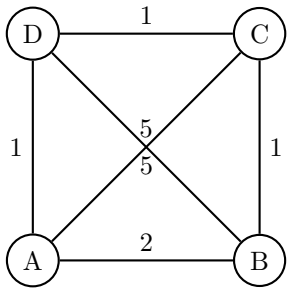
2. Kruskal's Algorithm *(6 points)*

Draw the state of the union-find data structure (union operations using rank, but without path compression) at the end of the $7th$ iteration of Kruskal's algorithm on the graph shown below.



3. Traveling Salesman Problem *(10 points)*

In the lecture, we've learnt an approximation algorithm for traveling salesman problem based on computing MST and a depth first traversal. Suppose we run this approximation algorithm for Traveling Salesman Problem on the following graph:



The algorithm will return different tours based on the choices it makes during its depth-first traversal stage.

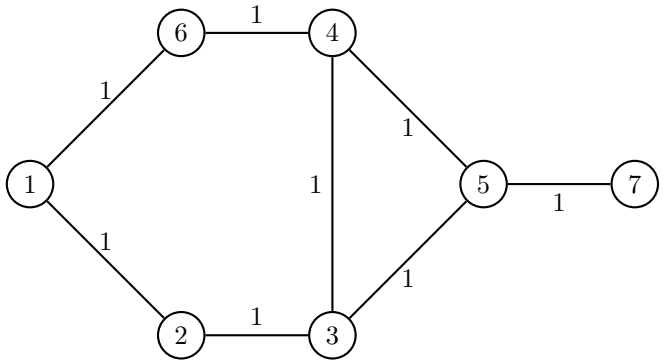
(a) *(3 points)* Which DFS traversal leads to the best possible output tour?

(b) *(3 points)* Which DFS traversal leads to the worst possible output tour?

(c) *(4 points)* What is the approximation ratio given by the algorithm in the worst case for the above instance? Why is it worse than 2?

4. Shortest Paths *(12 points)*

Consider the following graph on 7 vertices.



(a) *(4 points)* Suppose we execute Djikstra's algorithm to compute distances from the vertex 1. Djikstra's algorithm maintains an array $Dist[1, \ldots, n]$ where the i^{th} entry $Dist(i)$ will eventually contain the distance from vertex 1 to vertex i.

What is the order in which the following quantities "reach their final correct values"?

$$\{Dist(3), Dist(5), Dist(6), Dist(7)\}$$

(b) *(8 points)* The Floyd-Warshall algorithm maintains an array of pairwise distances $d(i, j)$ during its execution. During the execution, the $d(i, j)$ values keep decreasing until they reach their "correct value".

What is the order in which the following quantities "reach their correct values"?

$$\{d(1,2), d(1,3), d(1,4), d(1,5), d(2,6), d(1,7)\}$$

5. Multiplying Numbers *(6 points)*

The following algebraic identities can be used to design a divide-and-conquer algorithm for multiplying n-bit numbers.

$$\left(2^{2n/3}a_2 + 2^{n/3}a_1 + a_0\right)\left(2^{2n/3}b_2 + 2^{n/3}b_1 + b_0\right)$$
$$= 2^{4n/3} \cdot a_2b_2 + 2^n \cdot (a_1b_2 + a_2b_1) + 2^{2n/3} \cdot (a_2b_0 + a_0b_2 + a_1b_1) + 2^{n/3} \cdot (a_1b_0 + a_0b_1) + a_0b_0$$

$$(a_0b_2 + a_2b_0) = (a_0 + a_2) \cdot (b_0 + b_2) - a_0b_0 - a_2b_2$$
$$(a_0b_1 + a_1b_0) = (a_0 + a_1) \cdot (b_0 + b_1) - a_0b_0 - a_1b_1$$
$$(a_1b_2 + a_2b_1) = (a_1 + a_2) \cdot (b_1 + b_2) - a_1b_1 - a_2b_2$$

(a) *(4 points)* Write the recurrence relation for running time $T(n)$ of the algorithm.

(b) *(2 points)* What is the running time of the algorithm?

6. Reduction Essentials *(8 points)*

Assume A and B are search problems, and A reduces to B in polynomial time. In each part you will be given a fact about one of the problems. Determine what, if anything, this allows you to determine about the other problem. *Answer each part in one sentence.*

(a) A is in **P**.

(b) B is in **P**.

(c) A is **NP**-hard.

(d) B is **NP**-hard.

7. Breaking Encryption *(5 points)*

After years of research, Horizon Wireless released an encryption algorithm E that encrypts an n-bit message in time $O(n^2)$.

Show that if $P = NP$ then this encryption algorithm can be broken in polynomial time. More precisely, argue that if $P = NP$, then the following decryption problem can be solved in polynomial time.

DECRYPT
Input: An encrypted message M (encrypted using the algorithm E)
Goal: Decryption of M.

8. Zero-Sum Games *(22 points)*

Alice and Bob are playing a zero-sum game whose payoff matrix is shown below. The ij^{th} entry of the matrix shows the payoff that Alice receieves if she plays strategy i and Bob plays strategy j. Alice is the row player and is trying to maximize her payoff.

Alice \ Bob	A	B
1	4	1
2	2	5

Now we will write a linear program to find a strategy that maximizes Alice's payoff.

(a) *(2 points)* The variables of the linear program are x_1, x_2 where x_i denotes

_____

and a variable p denoting Alice's payoff.

(b) *(4 points)* Write the linear program for maximizng Alice's payoff.

(c) *(4 points)* Eliminate x_2 from the linear program and write it in terms of p and x_1 alone.

(d) *(8 points)* Draw the feasible region of the above linear program in p and x_1.

(e) *(4 points)* What is the optimal solution and what is the value of the game?

9. NP-search problems *(10 points)*

Which of the following are NP-search problems? Justify your answers.

(a) Longest Increasing Sequence
Input: A sequence of numbers $a_1, \ldots, a_n$.
Goal: Find the longest increasing subsequence of $a_1, \ldots, a_n$.

(b) Longest Path
Input: A directed graph $G = (V, E)$.
Goal: Find the longest directed path in G.

10. Computing Diameter *(19 points)*

The diameter d of an undirected graph $G = (V, E)$ with unit lengths, is defined to be the maximum distance between any two vertices, i.e. $d = \max_{u,v \in V}(u, v)$, where $d(u, v)$ is the length of the shortest path between u and v.

(a) *(5 points)* Give a simple algorithm to compute d in time $O(|V| \cdot (|V| + |E|))$.

(b) *(10 points)* Give a 2-approximate algorithm that runs in $O(|V| + |E|)$ time for approximately computing the diameter. Justify why the output of your approximation algorithm d_{approx} satisfies: $d \leq d_{approx} \leq 2d$. (*Hint: diameter is twice the radius*)

(c) *(4 points)* Give an example where: $d_{approx} = 2d$.

11. Assigning Backups *(20 points)*

Horizon Wireless is building a 5G network. The company has a set V of wireless towers; the distance $d(i, j)$ between any two of them is known, and each tower is capable of transmitting to other towers within a distance r.

- To make this network fault-tolerant, Horizon wants to assign each tower $v \in V$ to **two** other backup towers, so that if v is about to fail, it can transmit its data to them.
- Due to storage constraints, each tower can only serve as a backup for up to **three** other towers.

(a) *(10 points)* Suppose Horizon partitions its towers V into active towers A and backup towers B. Devise an algorithm which, given $V = A \cup B$, a distance function $d(i, j)$ on V, and tower radius r, assigns each active tower $a \in A$ to two backup towers in B (subject to the storage constraint), or reports that no assignment is possible. (*Hint: build a graph and use a known algorithm.*)

(b) *(10 points)* To use its network more efficiently, Horizon wants to use all towers in V as active towers, but still wants to assign each tower $v \in V$ to two other towers in V as backups. Again, no tower can be a backup for more than three other towers. Give an algorithm to find the assignment of backups for every tower.

12. NP-complete problems *(24 points)*

Prove that the following problems are NP-hard.

In each case, specify which problem you are reducing from, which problem you are reducing to. Briefly, but precisely describe how you transform an instance of one problem to another. *Proof of correctness of the reduction is not necessary.*

(a) *(8 points)* Directed Rudrata Cycle
Input: A directed graph $G = (V, E)$.
Goal: Find a directed cycle that visits every vertex in V exactly once.

(b) *(8 points)* Californian Cycle
Input: A directed graph $G = (V, E)$ with each vertex colored *blue* or *gold*, i.e., $V = V_{blue} \cup V_{gold}$.
Goal Find a *Californian cycle* which is a directed cycle through all vertices in G that alternates between blue and gold vertices. (*Hint: Directed Rudrata Cycle*)

(c) *(8 points)* 4-SAT
Input: n boolean variables $\{x_1, \ldots, x_n\}$ and clauses $\{C_1, \ldots, C_m\}$ with each having exactly *four distinct* literals. For example, the following is an instance of $4 - SAT$.

$$(x_1 \vee x_2 \vee \overline{x_4} \vee \overline{x_5}) \wedge (x_3 \vee \overline{x_4} \vee \overline{x_1} \vee x_2) \wedge (x_1 \vee x_3 \vee x_4 \vee x_5)$$

Note that all the 4 literals within a clause have to be distinct.
Goal: Find an assignment to the variables $x_1, \ldots, x_n$ that satisfies all the clauses.

13. 3-Set Cover *(32 points)*

The 3-Set Cover problem is a special case of the Minimum Set Cover problem where every element appears in at most 3 sets. The formal definition of the problem is as follows.

3-Set Cover
Input: A universe of elements $\mathcal{U}$, a family of subsets $\{S_1, S_2, \ldots, S_m\} \subseteq \mathcal{U}$ such that each element $i \in \mathcal{U}$ belongs to at most 3 subsets in $\{S_1, \ldots, S_m\}$.

Solution: Find the smallest collection of subsets that covers every element in the universe $\mathcal{U}$.

(a) *(10 points)* We will now show that 3-Set Cover is NP-hard. using the fact that Vertex Cover is NP-complete.

We will reduce the _____ problem to the _____ problem.

Given an instance of the _____ problem, we construct an instance

_____ as follows.

The universe $\mathcal{U}$ consists of _____.

There is one set corresponding to every _____

Every element appears in at most 3-sets because ... _____.

The rest of the proof is straight-forward, and we skip it.

(b) *(6 points)* Now we will try to develop an approximation algorithm for 3-Set Cover using linear programming. First, let us try to model the problem using a linear program (our model will eventually only give us an approximation to the problem).

- There is one variable x_i for every set S_i, which denotes _____

- What are the constraints?

- What is the objective function?

(c) *(3 points)* Suppose $LPOpt$ denotes the optimal value for the linear program. Suppose Opt is the size of the smallest 3-Set Cover. Can $LPOpt > Opt$? Justify your answer.

(d) *(3 points)* Can $LPOpt < OPT$? Justify your answer.

(e) We will now prove that the following algorithm yields a 3-approximation for the problem.

```
Solve the linear program described earlier.
For each set S_i do,
  • Pick S_i if x_i ≥ 1/3
```

The proof of correctness of the algorithm has two parts.

i. *(5 points)* Argue that every element in the universe $\mathcal{U}$ is covered.

ii. *(5 points)* Argue that the number of sets picked by the algorithm is at most $3 \cdot LPOpt$.

(Scratch Work)

(Scratch Work)