Final

- **The exam has 6 questions, is worth 100 points, and will last 180 minutes.**
- Read the instructions and the questions carefully first.
- Begin each problem on a new page.
- Be precise and concise.
- The questions start with true/false and short answer, and end with long answer. The problems may **not** necessarily follow the order of increasing difficulty.
- Good luck!

1 True/False + Justification (3pts each)

For the following parts, state clearly whether the given statement is true or false and give a brief justification for your answer.

(a) In a zero-sum game, it's possible for Player 2 to give Player 1 an advantage by announcing her strategy prior to the game.

(b) In a zero-sum game, it's possible for Player 2 to give Player 1 an advantage by announcing **an optimal Player 2 strategy** prior to the game.

(c) Suppose the SCC algorithm on some graph returns the SCCs in the order: {G, H, I, J, K, L}, {D}, {C, F}, {B, E}, {A}. Then, it is guaranteed that in the first DFS (the one done on G^R), Postorder(H) > Postorder(D).

(d) There exists an algorithm using $O(\log n)$ space to compute the mean of a stream of n integers, each in $[0, n]$.

(e) If you find a polynomial time algorithm for an NP problem, then you've proved P = NP.

(f) If you find a polynomial time algorithm for an NP-complete problem, then you've proved P = NP.

(g) Consider $h_1(x) = x$ and $h_2(x) = 1 + (x \pmod 4)$, both from $\{1, 2, \ldots, 24\}$ to itself. Then $\mathcal{H} = \{h_1, h_2\}$ is universal.

(h) There is guaranteed to exist an $O(n^3)$ solution to the Vertex Cover problem if P = NP.

(i) Suppose that a Monte-Carlo algorithm A exists for a problem which always runs in time $T(n)$ returns a correct answer with probability $\frac{1}{2}$. Then there exists a Las-Vegas algorithm that solves the problem in expected time $2T(n)$. (Assume a correct answer can be verified in constant time.)

2 Short Answer (5pts each)

(a) The SCC algorithm returns the SCC's one by one in a reverse topological order (sink to source). How would you make a small modification to the algorithm to return the SCC's in topological order (source to sink) instead?

(b) Consider two algorithms for the same problem:

- Algorithm A, which runs in $O(n)$ and produces a correct answer with probability 0.7, and a wrong answer with probability 0.3.
- Algorithm B, which runs in $O(n \log n)$ and produces a correct answer with probability 0.99, and a wrong answer with probability 0.01.

Which of the two algorithms should you use to build a more asymptotically (in n) efficient algorithm with probability 0.99 of producing a correct answer? Justify. Assume that you can always check if an answer is correct in constant time.

(c) Argue (very briefly) that any algorithm to compute the minimum spanning tree of a graph $G = (V, E)$ must use at least $\Omega(|E|)$ time.

(d) Show that it is possible to achieve negative regret in the expert's problem if the losses are chosen ahead of time. Give the losses and your choices of experts.

(e) Show that if there is no unique minimum weight edge in a graph G, but exactly two minimum-weight edges e_1 and e_2, then both e_1 and e_2 must be included in any MST of G.

3 Party Planning Committee (12pts)

Dunder Mifflin is having a corporate party, and you (head of the Party Planning Committee) want to make the party as cool as possible! Here's what you know:

- Every employee in Dunder Mifflin (except the CEO) has one and only one direct boss – that is, the corporate hierarchy can be represented as a rooted tree.
- Due to the awkward nature of the corporate culture, an employee will not show up to the party if their direct boss (i.e. parent node) will also be present at the party.
- Each employee e has a coolness $C(e)$. The coolness of a party is the total sum of the coolnesses of all its participants.

Your task is to design an efficient dynamic programming algorithm which, given a corporate hierarchy tree, outputs the maximum possible coolness of the party.

(a) Describe your subproblems, the corresponding recurrence relation / base cases, and in what order to compute them. Also state how to compute the maximum possible coolness of the party from your subproblems.

(b) Give a runtime analysis for your algorithm in terms of N (the number of Dunder Mifflin employees).

4 Rudrata / Hamiltonian Redux (12pts)

Recall from lecture that the Rudrata / Hamiltonian Cycle (RHC) problem is: given an undirected graph G, does G have a cycle (i.e. a closed loop) which visits every vertex exactly once? Consider now the Rudrata / Hamiltonian Path (RHP) problem: given an undirected graph G, does G have a path (not closed) which visits every vertex exactly once?

(a) Give a reduction from RHP to RHC which has runtime polynomial in $|V|$ and $|E|$. Justify correctness.

Hint: Given an RHP instance G, consider adding a dummy vertex d.

(b) Give a reduction from RHC to RHP which has runtime polynomial in $|V|$ and $|E|$. Justify correctness.

Hint: Given an RHC instance G, consider duplicating a vertex in G and also adding dummy vertices s, t.

(c) Show that RHP is NP-Complete. Clearly identify whether you used the reduction in (a) or the reduction in (b) for your proof – only one of these reductions is relevant for this part.

Note: You may use the fact that RHC is NP-Hard without proof.

5 GME to the... Forest Hideouts? (12pts)

Robin Hood and his merry band are on the run from the Sheriff! Let $G = (V, E)$ be an undirected, unweighted graph where vertices represent towns and edges represent roads between towns. Robin's band has n members (including Robin), and each member starts in a different town s_i. The objective for Robin's band is to find paths for each member which end at one of the hideouts, where the set of hideouts is given by $H \subseteq V$.

However, there's a catch: No two members of Robin's band can visit the same town, since two band members passing through the same town (even at different times), would draw too much attention from the Sheriff (e.g. if Robin's path includes town t, then Little John's path cannot include town t).

Note: The above restriction applies to hideouts as well, i.e. two band members cannot end at the same hideout.

(a) Given G, H and $s_1, \ldots, s_n$, reduce the problem of whether non-intersecting paths to hideouts exist for Robin's band to the max-flow problem from class.

(b) Prove the correctness of your reduction.

Note: You may use without proof the fact that an integer max-flow exists if all capacities are integers.

(c) Give a runtime analysis of the pre-processing step in your reduction, in terms of $|V|$ and $|E|$.

6 K-Paths (12pts)

Suppose T is a rooted tree (not necessarily binary) and k is an integer. Your first task is to devise an algorithm to compute **the largest possible set of non-intersecting directed k-paths**.

- A directed k-path is a path in the tree of exactly k edges that goes strictly upwards (i.e. always from child to parent) in the tree, i.e. no path contains two children of the same node. Directed k-paths do not have to start at leaves.
- By *non-intersecting*, we mean that distinct k-paths do not share any vertices.

(a) Describe a greedy algorithm which, given T and k as input, computes the largest possible set of non-intersecting directed k-paths. Show that your algorithm is correct and analyze its runtime.

(b) Now, suppose that each vertex in T has a reward associated to it and you want to find the set of non-intersecting directed k-paths which maximizes the sum of the vertex rewards it covers. Draw or describe a counterexample to show that your greedy algorithm does not work for this problem.

(c) Now suppose each reward is independently drawn uniformly at random from $\{0, 1, 2, \ldots, 10\}$. Show that for any given rooted tree, your greedy algorithm achieves an average approximation factor of at least $\frac{1}{2}$ for the problem in (b). That is, show that

$$\mathbb{E}_r\left[\frac{G(r)}{M(r)}\right] \geq \frac{1}{2},$$

where $G(r)$ and $M(r)$ are respectively the greedy algorithm's total reward and maximum total reward for r, a random assignment of rewards for the given tree.

Hint: Let N be the maximum number of non-intersecting k-paths for the given tree. Start by showing that

$$\mathbb{E}_r\left[\frac{G(r)}{M(r)}\right] \geq \frac{1}{10N(k+1)} \cdot \mathbb{E}_r[G(r)].$$