Final

Name:

SID:

Exam Room:

SID of student to your left:

SID 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 SID on the front of every page.* We may deduct points if your name is missing from any page. You will not be allowed to fill in your name after time is called.
- 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 are 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 we will not look at it for grading.
- You may consult only *two 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 **15** pages (8 double sided sheets) 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 170 minutes: there are 8 questions on this exam worth a total of 170 points.**
- Good luck!

1 True or False? (11 points)

Bubble in the right answer. No explanation needed. No points will be subtracted for wrong answers, so guess all you want! This part will be graded automatically. Please mark your answer clearly.

1. For every real $0 < x \leq 1$, $\sum_{i=1}^{n} x^i$ is $O(1)$ in terms of n asymptotically.

   ◯ True

   ◯ False

2. Dijkstra's algorithm (without any modification) can be used to compute the shortest path between 2 vertices in a DAG with exactly one edge having a negative weight (all other edges have a positive weight).

   ◯ True

   ◯ False

3. A bipartite graph can contain a simple cycle that consists of an odd number of unique vertices.

   ◯ True

   ◯ False

4. If a bipartite graph has a perfect matching of size n, then the minimum vertex cover of this graph must be also be of size equal to n. (A perfect matching is a bijective pairing of vertices).

   ◯ True

   ◯ False

5. It is possible that one-way permutations exist and P = NP.

   ◯ True

   ◯ False

6. For every odd prime p and every integer $a \in \{1, ..., p-1\}$, $a^{(p-1)/2}$ is either 1 or -1 modulo p.

   ◯ True

   ◯ False

7. It is possible that problem A reduces to problem B, problem B reduces to problem C, problem C reduces to problem A, problem A is NP-complete and problem B is in P. (Assuming P $\neq$ NP)

   ◯ True

   ◯ False

8. Doubling the capacities of all edges of a graph G doubles the maximum flow.

   ◯ True

   ◯ False

9. Factoring is known to be NP-hard.

   ◯ True

   ◯ False

10. Consider a graph with exactly one edge having rational capacity and all other edges having integral capacities. The max flow in this graph must be integral.

   ◯ True

   ◯ False

11. The maximum weight edge in a cut of the graph can never be a part of a MST.

   ◯ True

   ◯ False

2 Fill in the Blanks (20 points)

When asked for a bound, always give the tightest bound possible. Some questions have choices in parentheses after the answer box

1. (4 points) The node that receives the highest $\boxed{}$ (pre/post) number in a depth-first search must lie in a $\boxed{}$ (source/sink) strongly connected component.
2. (2 points) If an undirected graph with no duplicate edges has a cycle then it must have at least $\boxed{}$ edges.
3. (2 points) The space complexity of the Floyd-Warshall algorithm (all-pairs shortest path dynamic programming algorithm) is $\boxed{}$, where n and m are the number of vertices and edges in the graph, respectively.
4. (2 points) Let $x = BERKELEY$ and $y = BARKELY$ and $E(i, j)$ be the edit distance between $x[1 \cdots i]$ and $y[1 \cdots j]$. Then, $E(2, 3) = \boxed{}$.
5. (2 points) Continuing from above, $E(8, 7) = \boxed{}$.
6. (2 points) A degree d polynomial is uniquely characterized by its values at any $\boxed{}$ points.
7. (2 points) A directed graph has a cycle if and only if its depth-first search reveals a $\boxed{}$ (tree/forward/back/cross) edge.
8. (2 points) The size of the maximum flow in a network is always $\boxed{}$ ($<$, $\leq$, $=$, $\geq$, $>$) the capacity of any (s, t)-cut.
9. (2 points) In union/find data structure of n items, if we use union by size without path compression then a any combination of m unions and/or find operations takes at most $\boxed{}$ time.

Name:

3 Zero-Sum Games (9 points)

Consider a zero-sum game given by the following matrix (indicating the payoffs to the row player)

	C_1	C_2	C_3
R_1	-2	3	1
R_2	3	-1	2
R_3	-1	4	-1

Suppose the column player has fixed the following probabilistic strategy:

C_1	C_2	C_3
$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

what is the best strategy for the row player?

R_1	R_2	R_3

If row player chooses to play as above, then instead of the above (uniformly random) column player strategy what would have been the best strategy that the column player could have taken?

C_1	C_2	C_3

4 Mind Reading Strategy (20 points)

Alice has chosen a number x from $\{1, \ldots, n\}$ and Bob is trying to guess the number. Bob can ask Alice questions of the form: Is $x \geq i$? for some $i \in \{1, \ldots, n\}$. For each such asked question, Alice answers YES or NO. Let p_i denote the probability that Alice's chosen number is i. (Note that $\sum_{i=1}^{n} p_i = 1$.)

Bob knows these probabilities, and would like to devise a strategy to determine Alice's number correctly while minimizing the expected number of questions asked.

Given the probabilities p_i as input, design a dynamic programming algorithm to compute the expected number of questions needed for the best possible strategy for Bob.

1. Clearly define a subproblem that could be used to efficiently solve this problem.

2. What are the base cases?

3. What is the recurrence relation?

5 Melting Snow (15 points)

You are given a terrain map in the form of a $N \times N$ array T; namely, the value $T(i,j)$ represent the height of the location (i,j). As global temperature increases, snow melts and water flows through the terrain. At a particular coordinate, water will flow to the neighbor coordinate with the **smallest** height. Break ties arbitrarily. A neighboring coordinate could be directly or diagonally adjacent. Water stops flowing at a coordinate if all neighboring heights are **greater or equal**. These coordinates are called end coordinates.

For each coordinate (i,j), your task is to determine the coordinate (i',j') that the water will ultimately flow to starting with the coordinate (i,j). Specifically, your algorithm should output an $N \times N$ array M with the following property. For each i,j, $M[i,j]$ is equal to the coordinate that water will ultimately flow to starting with from the coordinate (i,j).

a Provide the main idea or pseudocode of your algorithm. Ideally your answer should be in **bullet point** format.

b What is the runtime in Θ notation?

6 Short Selling (35 points)

1. **2 trades (15 points)**
   In stock trading, short selling is the act of selling a stock that you don't own (by borrowing) with the promise to buy it back in the future. To profit from this, you would want to sell high first and then buy low at a later date. You are given an array of stock prices in chronological order $S[1..N]$. Profit off 1 trade in the period is defined as such that i and j are indices to S.

   $$S[i] - S[j], 1 \leq i < j \leq n$$

   Suppose you can make up to **two** transactions over the period such that you have to complete the first transaction before initiating the second one. You can perform Sell Buy Sell Buy, but not Sell Sell Buy Buy. You are allow to make 0 or 1 instead of 2 if it's not profitable to make 2 transactions. If you make a sell action, it must be paired with a buy action with a later index.

   Provide an **efficient algorithm** to determine the largest total profit if you are allowed to make up to 2 trades.

   For example, if $S = [5, 2, 6, 4, 10, 3, 7]$, can profit 3 if you sell at 6 and buy at 3. Can profit 5 if you sell at 6, buy at 4, sell at 10, and buy at 7.

   a Provide the main idea or pseudocode of your algorithm. Ideally your answer should be in **bullet point** format.

   b What is the runtime in Θ notation?

2. **k trades (20 points)**
Now find the largest profit if we can short sell **at most** K times. Again you cannot interleave sell and buy actions. It must be Sell Buy Sell Buy ...

   (a) What are the subproblems?

   (b) What are the base cases?

   (c) What is the recurrence relation?

   (d) What is the runtime in Θ notation?

7 NP-Completeness Reductions (38 points)

Show that the following problems are NP-complete by providing a polynomial-time reduction. You may skip showing that the given problem is in NP. You may assume that the following problems are NP-complete: Rudrata Path or Hamiltonian Path, Hamiltonian Cycle, Vertex Cover, Independent Set, 3-SAT, CircuitSAT, Integer Linear Programming, Clique, 3D Matching, Partition, and Subset Sum.

1. **Set Packing. (8 points)**
   Input: Subsets $S_1, \ldots, S_m$ of a set U and a positive integer $k \leq m$.
   Question: Are there k subsets among $S_1, \ldots, S_m$ that are mutually disjoint (i.e. the intersection of any two subsets are empty)?

   Proof: *We will reduce the problem **A** ...* to the problem **B** ...*

   Given an instance Φ of the problem A we construct an instance Ψ of the problem B

   as follows ...

   The proof that this is a valid reduction is as follows:

   ■

Name:

2. **Bin Packing. (10 points)**
Input: n items with sizes $a_1 \cdots a_n$ respectively, a positive integer B (bin capacity) and a positive integer k (number of bins).
Question: Is there a partition of the set $\{1 \cdots n\}$ into sets $S_1, \ldots, S_k$ such that for each $i \in \{1 \cdots k\}$ we have that $\sum_{j \in S_i} a_j \leq B$?

Proof: *We will reduce the problem **A** ...* *to the problem **B** ...*

Given an instance Φ of the problem A we construct an instance Ψ of the problem B

as follows ...

The proof that this is a valid reduction is as follows:

■

3. **Quadratic Assignment Problem. (10 points)**
   Input: Integers n and b, two $(n \times n)$ integer-value matrices $C = \{c_{ij}\}_{1 \le i,j \le n}$ and $D = \{d_{kl}\}_{1 \le k,l \le n}$.
   Question: Is there a permutation $f : \{1, ..., n\} \to \{1, ..., n\}$ such that $\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} d_{f(i)f(j)} \le b$?
   Hint: Rudrata Path/Cycle.

Proof: *We will reduce the problem* ***A*** *…* *to the problem* ***B*** *…*

Given an instance Φ *of the problem* A *we construct an instance* Ψ *of the problem* B

as follows …

The proof that this is a valid reduction is as follows:

■

Name:

4. **Induced Path. (10 points)**
Input: An undirected graph $G = (V, E)$ and an integer k.
Question: Is there an induced path of length k (i.e. a sequence of k distinct vertices $v_1, \ldots, v_k$ such that there exists an edge in G between every pair of consecutive vertices and and for every pair of non-consecutive vertices there does not exists an edge in G)?

Hint: Try to modify the reduction from 3SAT to Independent Set, in which you create a vertex for every literal in every clause, so that which vertex is chosen in the independent set means which literal is set to true for the clause. Try to follow a similar strategy and think about how to connect edges, and make use of the fact that these are induced paths rather than just paths. There are several constructions that work. You may also try to create two vertices for every literal in every clause instead of just one.

Proof: *We will reduce the problem* ***A*** *...* *to the problem* ***B*** *...*

Given an instance Φ *of the problem* A *we construct an instance* Ψ *of the problem* B

as follows ...

The proof that this is a valid reduction is as follows:

■

Name:

8 Approximation: Maximum Coverage (22 points)

The Maximum Coverage Problem is an optimization problem defined as follows.
Input: Subsets $S_1, \ldots, S_m$ of a set $\mathcal{U}$, and a positive integer $k \leq m$.
Output: Find $T \subseteq \{1, \ldots, m\}$ of size k such that the size of $\bigcup_{i \in T} S_i$ is maximized. (I.e., find k subsets whose union has maximum size.) We say that the elements in this union are *covered* by the subsets.

1. (6 points) Briefly argue that, if we can solve the Maximum Coverage problem optimally, then we can also solve the Set Cover problem. (Note: this implies that Maximum Coverage is NP-hard.)

2. Let us examine the greedy algorithm which proceeds in k steps as follows: In each step, pick the subset that covers the maximum number of (currently) uncovered elements.

   (a) (4 points) Give a counterexample for which the greedy algorithm does not give an optimal solution.

(b) (6 points) Show that, when $k = 2$, this algorithm is a $3/4$-approximation algorithm for the problem (i.e. the two sets picked by the algorithm covers $3/4$ as many elements as the optimal solution). *Hint:* It may be helpful to look at the hint from part (c).

(c) (6 points) Generalize the analysis above to show that, for every k, the greedy algorithm is an $\big(1 - (1 - 1/k)^k\big)$-approximation algorithm for Maximum Coverage.
Hint: Let x_i denote the number of elements covered by the greedy solution after picking i subsets. Try to show that $x_{i+1} - x_i \geq (OPT - x_i)/k$ where OPT is the total number of elements covered by the optimal solution. You might find it useful to consider the quantity $y_i = OPT - x_i$ as well.