Final Exam

Name:

Penn State ID:

Student ID number:

Instructions: Answer all questions (no % 20 option on the exam). Read them carefully first. Be precise and concise. Handwriting needs to be neat (if we can't understand what you write, we can't grade it). Box numerical final answers. Write only on the provided space; if you need extra space for a question, put a note in the question and use the extra pages provided at the end.

Please clearly write your name and your PSU Access User ID (i.e., xyz1234) in the box on top of every page.

Good luck!

Name:	PSU Access ID (xyz1234):

1 True/False *(60 points)*

True or false? Mark the right answer with an "x" (not a check mark or a circle). No justification is needed. No points will be subtracted for wrong answers. In some cases, it wil be helpful to come up with counterexamples or short proofs to convince yourself.

T F

☐ ☐ Every DAG has exactly one topological ordering.

☐ ☐ In a DFS of a directed graph G, the vertex with largest post number is in a sink strongly-connected component of G.

☐ ☐ If all edge capacities of a flow network are integers, the value of the maximum flow is an integer.

☐ ☐ If in a flow network, if we increase the capacity of an edge in a minimum cut, the maximum flow increases.

☐ ☐ If in a flow network, removing the edge with the smallest capacity would always reduce the value of the maximum flow.

☐ ☐ In a flow network, removing the edge with the largest capacity would always reduce the value of maximum flow.

☐ ☐ Deciding if a bipartite graph has a perfect matching can be reduced to computing the maximum flow in a flow network.

☐ ☐ Let S be the minimum cut value of a flow network where each edge has unit capacity. Then increasing each edge capacity by 1 would always increase the value of maximum flow by exactly S.

☐ ☐ The simplex algorithm runs in polynomial time.

☐ ☐ It is possible for an optimal solution to a linear program to be on an edge of the feasible region, as opposed to just a vertex.

☐ ☐ Each iteration of the simplex algorithm can be implemented in polynomial in n and m time (n is the number variables and m the number constraints).

☐ ☐ There is a polynomial-time algorithm for linear programming.

☐ ☐ For LPs, if the primal problem is a minimization problem, weak duality states that any feasible solution to the dual problem upper bounds the optimal value of the primal problem.

☐ ☐ Consider a two-player zero-sum game. If the row player has announced and fixed a pure strategy, there exists an optimal strategy for the column player which is pure (always choose the same move).

☐ ☐ In a dynamic programming algorithm, the time required by the algorithm is determined by the number of nodes in the DAG of the subproblems.

☐ ☐ Every NP-complete problem is in NP.

Name:	**PSU Access ID:**

☐ ☐ If a problem A reduces to a problem B (in polynomial time), and B is NP-complete, then A is NP-complete (Think of A and B as problems in NP.)

☐ ☐ k-SAT is in P.

☐ ☐ The class NP contains the set of all problems whose solution can be verified in polynomial time.

☐ ☐ There are NP-complete that can be solved with polynomial space.

☐ ☐ For a given problem X, if there is a polynomial time reduction from X to every problem in NP, then X is NP-hard.

☐ ☐ There is a polynomial time reduction from Linear Programming to Circuit SAT.

☐ ☐ The QSAT problem (i.e., quantifiable SAT) is PSPACE-complete.

☐ ☐ We know there is 3/2-approximation algorithm for the Traveling Salesman Problem that runs in polynomial time.

☐ ☐ In the primal-dual scheme, we iteratively improve the infeasibility of a primal solution and the optimality of a dual solution.

☐ ☐ For any NP-Complete problem, we have an α-approximation algorithm with polynomial running time for some constant $\alpha > 0$.

☐ ☐ In the coupon collecting problem (i.e., every day we receive one of n coupons uniformly at random), the expected number of days required to collect every coupon is $\Theta(n^2)$.

☐ ☐ If X is a non-negative random variable such that $E[X] = 12$, then the probability that $X \geq 30$ is at most 2/5.

☐ ☐ In the setting of the Multiplicative Weight Update algorithm, if we knew all losses of all experts on all days in advance (before the first day), we may be able to incur less total loss (over T days) than the total loss incurred by the best expert.

☐ ☐ In the setting of the Multiplicative Weight Update algorithm, there always exists a strategy that chooses the same expert on every day (and no other experts) that achieves regret equal to 0.

Name:	**PSU Access ID:**

2 Short questions *(20 points)*

1. (**8pts**) A set of n cell towers are located at fixed, numerical locations $(x_1, y_1), \ldots, (x_n, y_n)$. Each cell tower has an associated power p_i whose value you can control. The range of the cell tower i with power p_i is defined to be any point at a distance from (x_i, y_i) of less than or equal to p_i. You want to minimize the total power of all the towers while still guaranteeing that any person traveling in a straight line between (x_i, y_i) and (x_j, y_j) is within range of either tower i or tower j (or both). Write (but do not solve) a linear program that solves this problem. (Recall that the distance between points (x_i, y_i) and (x_j, y_j) is $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.)

$$\begin{aligned} &\text{minimize} \quad p_1 + \cdots + p_n \\ &\text{subject to } p_i + p_j \geq \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad \forall i = 1, \ldots, n \text{ and } j = 1, \ldots, n \\ &p_i \geq 0 \quad \forall i = 1, \ldots, n \end{aligned}$$

2. (**6pts**) Consider the following zero sum game. (Each entry represents the row player's payoff. The row player is trying to maximize the payoff.)

$$P = \begin{pmatrix} 3 & -1 \\ 5 & 5 \end{pmatrix}$$

If the row player plays $(3/4, 1/4)$ and the column player plays optimally, What is the expected payoff of this game?

It suffices to compare the two pure strategies:

$$\frac{3}{4} \times 3 + \frac{1}{4} \times 5 = \frac{7}{2}$$
$$\frac{3}{4} \times (-1) + \frac{1}{4} \times 5 = \frac{1}{2}$$

The column player tries to minimize, so the optimal expectefd payoff is 1/2.

Name:	**PSU Access ID:**

3. (**6pts**) Given a stream of length m containing only 1's, 2's, and 3's, let P denote the product of all elements of the stream. Describe an algorithm to compute $\log_2 P$ that uses $O(\log m)$ space. (Note that simply multiplying all the numbers does not work since this requires $\Theta(m)$ bits.)

   Keep counters k_2 and k_3 for the number of 2's and 3's in the stream, respectively, and outout

$$\log_2(2^{k_2}3^{k_3}) = k_2 + k_3 \cdot \log_2 3.$$

Name:	**PSU Access ID:**

3 Dynamic Programming: diverse knapsack *(20 points)*

Suppose we have n items available where item i has weight w_i and value v_i. We have an unlimited supply of each item. The value of any multiset of items (recall that a multiset is a set where repeated elements are allowed) is the sum of the values of all the items in the multiset. For example, the value of the multiset that contains item 1 twice and item 2 once is $2v_1 + v_2$. Likewise, the weight of the multiset is the sum of the weights of all the items in it.

Given as input a weight W and an integer K, you would like to find a multiset of items of maximum value with weight W that has at least K different items. Give an algorithm for finding the maximum value of any multiset of weight W with at least K different items.

Hint: Let $K(w, k, i)$ be the maximum value of a multiset of weight w that uses only items $1, \ldots, i$ and uses k different items; they to give a recurrence for computing $K(w, k, i)$ as well as the base cases.

Recurrence: Consider $K(w, k, i)$, the optimal value for using at least k different items among $1, \ldots, i$ with total weight no more than w, there are three cases regarding item i

1. The optimal solution doesn't use i, then the optimal value is the same as $K(w, k, i-1)$
2. The optimal solution uses i once, then the optimal value is the same as $K(w - w_i, k-1, i-1) + v_i$
3. The optimal solution uses i more than once. In this case we just take one copy away, and the value is $K(w - w_i, k, i) + v_i$, notice we don't decrease i, since we will (potentially) use i again, and we don't decrease k here, since we want to decrease it only once for all copies of item i, and we wait until the last time we use i to decrease k.

Thus we have the recurrence $K(w, k, i) = \max\{K(w, k, i-1), K(w-w_i, k-1, i-1) + v_i, K(w-w_i, k, i) + v_i\}$
Base case: For the above recurrence, we need to deal with three base cases

1. $w < 0$: this case violates the weight constraint, we need $K(w, k, i) = -\infty$
2. $w \geq 0, i = 0, k = 0$: We have no more items to choose, so $K(w, k, i) = 0$
3. $w \geq 0, i = 0, k > 0$: this case violates the number of different items constraint, we need $K(w, k, i) = -\infty$.

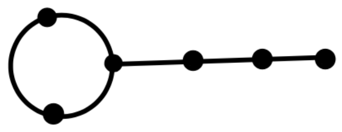
Notice in the base cases, if you have 0 instead of $-\infty$ for those cases, you may end up with a solution not satisfying the constraints.

Alternatively, you can have the recurrence $K(w, k, i) = \max\{\max_{1 \leq a \leq \lfloor w/w_i \rfloor}\{K(w - aw_i, k-1, i-1) + av_i\}, K(w, k, i-1)\}$ if you want to make the decision of how many copies of item i to use all at once. You won't need the $w < 0$ base case then, since your recurrence will never incur $K(w, k, i)$ for $w < 0$.

Name:	**PSU Access ID:**

4 A reduction: Lollipops (*20 points*)

Let G be a graph with $2n$ nodes. We say that G has a *lollipop* if G has a simple cycle with n nodes, and there is also a simple path consisting of the remaining n nodes of G, and finally there is an edge connecting one of the endpoints of the path with one of the nodes in the cycle, as in the following example where $n = 3$:



The Lollipop problem is the following problem: Given a graph G with an even number of nodes, does it have a lollipop? We want to show that this problem is NP-complete.

(a) Show that Lollipop is in NP.

The easiest solution was to assume that the certificate contained a list of edges sorted, so that the cycle came first and the path after that (or viceversa). Then, checking if the list is a lollipop is trivial; remember that all we need is a certificate that can be checked quickly.

If you only assume that the verifier recieved a list of edges, then it was also possible but more subttle. One correct solution was to identify the unique vertex of degree 1, follow the path for n steps until the vertex of degree 3, and then follow the cycle.

(b) Show that Lollipop is NP-complete.

The cleanest reduction was from Hamiltonia Cycle. Take a graph G and suppose we want to detect if there is a HC in G. Attach a path of n vertices to any vertex of G and feed this gadget to the Lollipop solver. If you get YES, then G must have a HC, otherwise there is no HC in G.

There were other correct, or nearly correct reductions that also received full credit.

Name:	**PSU Access ID:**

5 Approximation algorithms: (*20 points*)

1. Consider the following LP relaxation for the minimum vertex cover problem, in which, given an unweighted undirected graph $G = (V, E)$, the goal is to find the smallest set of vertices that includes at least one endpoint of every edge of G.

$$\begin{aligned} &\min \sum_{u \in V} x_u \\ &\text{s.t. } x_u + x_v \geq 1 \text{ for all } \{u, v\} \in E \\ &\qquad x_u \geq 0 \ \text{ for all } u \in V \end{aligned}$$

Consider the approximation algorithm for the vertex cover problem that first solves the LP and obtains a set of fractional values x_u (for each $u \in V$) and then takes vertex u into the solution if and only if $x_u \geq 1/2$. Show that this algorithm is a 2-approximation algorithm for the minimum vertex cover problem.

Need to check both that the solution is valid and the approximation ratio. In any solution to the LP, including the optimal, $x_u + x_v \geq 1$ for every edge, so either x_u or x_v needs to be $\geq 1/2$ and so every edge is covered.

For the approximation ration, note that $\alpha = \max_I \frac{A(I)}{OPT} \leq \frac{2OPT}{OPT} = 2$.

2. In the MAX-EXACT-k-SAT problem, we are given a set of clauses m clauses on n variables, each a disjunction of at eaxct k literals, and we want to find an assignment that satisfies as many of them as possible.

Provide a *randomized* approximation algorithm that is a $\left(1+\frac{1}{2^k-1}\right)$-approximation *in expectation.* The running time of your algorithm should be $O(n)$.

Assign every variable to True with probability 0.5 and False otherwise indepndently. This is the same idea as the coloring problem in the Problem Set 6. The probability that a clause is not satisfied is $1 - 1/2^k$ (only 1 out of 2^k possible variable assignments make a clause false). So, the expected number of satisfied clauses iss $m * (1 - 1/2^k)$. Since this is a maximization problem, the expected aproximation ratio is

$$\alpha = \max_I \frac{OPT}{A(I)} \leq \frac{m}{(1 - 1/2^k)m} = 1 + \frac{1}{2^k - 1}.$$

Name:	**PSU Access ID:**

6 Randomized grading (*20 points*)

Professor AB only assigns A and B grades in his class. He does so at random as follows: if a student attended office hours at least once in the semester, then Professor AB assigns A with probability 0.9 and B with probability 0.1 (independently of any other student). If, on the other hand, a student never showed up to office hours, then Professor AB assigns A to the student with probability 0.5 and B otherwise.

Suppose there are n students in the class (with n even) and that exactly half of the students attended office hours at least once. Let X be the random variable corresponding to number of students that receive an A in the class.

1. Compute the expectation of X.

$$E[X] = 0.9 \cdot \frac{1}{2} \cdot n + 0.5 \cdot \frac{1}{2} \cdot n = 0.7n$$

2. Compute the variance of X.

   Let X_1 and X_2 be the random variables corresponding to number of students that receive an A in the class who attended and not attended office hours, respectively. Then $X = X_1 + X_2$ and X_1 and X_2 are indepdendent, so

$$Var(X) = Var(X_1) + Var(X_2) = \frac{n}{2}0.9(1-0.9) + \frac{n}{2}0.5(1-0.5) = \frac{n}{2}(0.09 + 0.25) = 0.17n$$

3. Show that the probability that 80% or more of the students get A (i.e., that $X \geq 0.8n$) is at most $20/n$.

   Using Chebyshev's inequality taking $k = \frac{0.1n}{\sqrt{0.17n}}$

$$P[|X - E[X]| \geq k\sqrt{0.17n}] \leq 1/(0.1\sqrt{n}/\sqrt{0.17})^2 = 17/n \leq 20n.$$

   Note that

$$P[X \geq 0.8n] = P[X - 0.7n \geq 0.1n] \leq P[|X - 0.7n| \geq 0.1n] = P[|X - E[X]| \geq k\sqrt{0.17n}].$$

Name:	**PSU Access ID:**

Extra space.

Name:	PSU Access ID:

Extra space.

Name:	PSU Access ID:

Extra space.