Final Solutions

- **The exam has 8 questions, is worth 110 points, and will last 170 minutes.**
- The point distribution is 3/15/9/8/15/15/15/30. You may be eligible to receive partial credit for your proof even if your algorithm is only partially correct or inefficient.
- Answer all questions. Read them carefully first. Not all parts of a problem are weighted equally.
- Be precise and concise.
- The problems may **not** necessarily follow the order of increasing difficulty.
- The points assigned to each problem are by no means an indication of the problem's difficulty.
- Unless the problem states otherwise, you should assume constant time arithmetic on real numbers.
- If you use any algorithm from lecture and textbook as a blackbox, you can rely on the correctness and time/space complexity of the quoted algorithm. If you modify an algorithm from textbook or lecture, you must explain the modifications precisely and clearly, and if asked for a proof of correctness, give one from scratch or give a modified version of the textbook proof of correctness.
- Good luck! (Destress question) Recommend us an album/movie you like:

1 DP Warmup

(3pts) Write a dynamic programming recurrence to count $f(n, m)$, the number of distinct shortest paths in a 2D integer grid from $(0, 0)$ to (n, m), where n and m are positive integers. You can only travel along the x and y directions (no diagonals). A counting argument will receive no credit. Only the recurrence relation is needed.

2 Misc True/False + Justification

(3pts each) State clearly whether the given statement is true or false and give a brief justification for your answer.

(a) There exists a graph with 3 vertices such that the lowest postorder number of *any* DFS on the graph is not in a sink SCC.

(b) Let $f\colon \mathbb{N} \to \{0, 1, 2, 3, 4\} = 3x + 4 \pmod 5$. Then $\mathcal{H} = \{f\}$, where $\mathcal{H}$ constitutes of *that single function* f, is a universal hash family of functions: $\mathbb{N} \to \{0, 1, 2, 3, 4\}$.

(c) Let the output of the Morris algorithm for an approximate counting of a stream of n numbers be $\tilde{n}$. Then the difference between n and $\tilde{n}$ (*i.e.* $|n - \tilde{n}|$) is bounded by a constant.

(d) We don't need to calculate the 7th roots of unity to calculate the `FFT` of a length 7 vector.

(e) $\sqrt{n} = O(2^{\sqrt{\log n}})$

3 reducecuder

(3pts each) State clearly whether the given statements are True, True iff $P = NP$, True iff $P \neq NP$, or False and give a brief justification for your answer.

(a) There exists a polynomial time reduction from 3SAT to Palindrome Checking. Assume that Palindrome Checking is a program that verifies if a given word is a palindrome or not.

Note: A palindrome is a word that reads the same forwards as well as backwards. For example, "CS17071SC" is a palindrome, but "CS170" is not.

(b) There exists a polynomial time reduction from any NP hard problem to any other NP hard problem.

(c) There exists a polynomial time reduction from Palindrome Checking to 3SAT.

4 Param's Picnic Planning Fiasco

(8pts) The CS 170 course staff is planning a road trip to Guinland. Originally, Param was in charge of seating assignments in cars for all members of the course staff. He arranged p people in n cars where car i has capacity $c_i \forall i \in 1, \ldots, n$.

However, given Param's poor planning skills, he assigned people in cars terribly inefficiently leaving a lot of empty space in many cars. Given the number of people p, and the capacities of each of the n cars $c_1, \ldots c_n$, come up with an efficient greedy algorithm to fit people in cars such that the CS 170 course staff can take the least number of cars possible on the road trip.

Prove the optimality of your algorithm using an exchange argument. No runtime analysis required.

Exam continues on next page ⟹

5 Mo' Money Mo' Problems

(15 pts) Thanks to your efforts on the project, CS 170's igloo polishing venture is flourishing! In an attempt to cut costs and further maximize profits, CS 170 now needs your help figuring out the cheapest way to get from one igloo polishing task to the next. Our clients are spread out in the vast kingdom of Guinland which consists of $|V|$ cities. Assume that no two consecutive polishing jobs are in the same city.

The $|V|$ cities of Guinland are connected by $|E|$ directed edges which can be used to travel from one city to another, i.e. we can use an edge (u, v) to travel from city u to another city v. To travel from one city u to another city v we can use either cable car, snowmobiles, or trains which have strictly positive costs $C_{c(u,v)}$, $C_{s(u,v)}$, and $C_{t(u,v)}$ respectively. If you can travel from city u to city v, you can use any of the three modes of transport, i.e. if the directed edge (u, v) exists, all 3 modes of transport will be available between u and v each with their own respective costs.

Unfortunately, for some bizarre reason, we cannot take any same mode of transportation twice in a row, i.e., we cannot enter city u on a snowmobile and leave city u on a snowmobile, and the same applies for cable cars and trains. Given a directed graph representation of Guinland $G = (V, E)$ and cities s and t, come up with an algorithm that helps us find the cheapest path between city s and city t in G and analyze its runtime. No need for a proof of correctness.

Exam continues on next page ⟹

6 Happy Birthday, Adnaan

(15pts) It's Adnaan's birthday today and the CS 170 staff are shopping presents for him in a shop. There are n items on the shopping list. The shop is doing a peculiar holiday sale: there are m "works well together" unordered pairs $e_k = (i, j)$, $k \in \{1, \ldots, m\}; i, j \in \{1, \ldots, n\}$. For the k-th pair ($k \in \{1, \ldots, m\}$), if you have already bought one item in the pair from the store, you are allowed to buy the other item in the pair for the price of d_k dollars instead of its original price. Given $(e_k)_{k=1}^m$, $(d_k)_{k=1}^m$, and the original price of the n items $(p_i)_{i=1}^n$, design an algorithm that outputs

- the least amount of money the staff could spend to buy all n items and
- a possible order of buying the n items that achieves this least amount of money.

Describe your algorithm and analyze its runtime. Proof of correctness is not needed.

Example:

$$n = 3, m = 3$$
$$(e_k)_{k=1}^m = (1, 2), (2, 3), (3, 1)$$
$$(d_k)_{k=1}^m = 4, 2, 6$$
$$(p_i)_{i=1}^n = 10, 5, 3$$

The minimum amount of money is \$9, and it could be achieved as follows.

1. First buy item 3 with its original price $p_3 = 3$.
2. Since we have bought item 3, we can buy item 2 with \$2 or its original price of \$5. Of course we pay \$2 for item 2 to minimize expenditure.
3. Since we have item 3, we can buy item 1 with 6 dollars. We also have item 2, which allows us to buy item 1 with 4 dollars. Its original price is \$10. Therefore we pay \$4 for item 1.

Therefore if we buy the items in the order of $3, 2, 1$, we spent a total of \$9 to obtain all items.

Hint: Minimum Spanning Tree.

Exam continues on next page ⟹

7 FALSY-SAT

(15 pts) Define the FALSY-SAT problem as follows:

Given a 3-CNF boolean formula ϕ, **decide** if there's an assignment to the variables of ϕ that satisfies ϕ without containing three true literals in any clause.

In other words, a FALSY-SAT instance is satisfied if and only if exactly 1 or 2 literal(s) in each clause is/are assigned to true.

For example, $(\overline{x} \vee \overline{x} \vee \overline{x})$ is not satisfiable under the conditions of FALSY-SAT because all three literals are the same, so to make the clause true, false must be assigned to x, and three literals in the clause, namely three $\overline{x}$, must all be true.

(a) Give a polynomial time reduction from 3SAT (*i.e.* **decide** if the given 3-CNF is satisfiable) to FALSY-SAT by replacing each clause c_i

$$(a_1 \vee a_2 \vee a_3)$$

with the two clauses

$$(a_1 \vee a_2 \vee b_i) \text{ and } (\overline{b_i} \vee a_3 \vee t)$$

where b_i is a new variable for each clause c_i and t is a single new additional new variable.

Hint: For one direction for the proof, show that you could always make $t = 0$.

(b) Suppose you have a correct answer to the previous part, conclude that FALSY-SAT is NP-Complete.

Exam continues on next page ⟹

8 Set Cover Symphony

(30pts) In class we saw that there is a $\ln n$ greedy approximation algorithm for set cover. In this problem, we will first see another approximation algorithm for set cover (part (a)), then we will discover how randomness could give us a much better algorithm (part (b) through (d)).

Note: all subparts are doable without completing the other subparts of this question. So if you're stuck, feel free to try out other subparts.

Recall in set cover we have a universe $\mathcal{U} = \{1, \ldots, n\}$ of items that we would like to cover by taking some subcollection of given sets $S_1, \ldots, S_m \subseteq \mathcal{U}$. We would like to take as few sets as possible. One can express set cover as an Integer Linear Program in the following way:

$$\min \sum_{i=1}^{m} x_i$$

s.t.

$\forall u \in \mathcal{U}$, $\sum_{i:\, S_i \ni u} x_i \geq 1$
$\forall 1 \leq i \leq m$, $0 \leq x_i \leq 1$
$\forall 1 \leq i \leq m$, x_i is an integer

Figure 1: Integer linear program representing set cover.

In the Integer Linear Program (ILP), we have one variable x_i per set S_i. We interpret an ILP solution with $x_i = 1$ as indicating that S_i should be taken in the subcollection, and $x_i = 0$ indicates that it should not be taken. Also, the notation "$\sum_{i:\, S_i \ni v}$" denotes a summation over all i such that v is in S_i.

(a) Show that if no element is contained in more than k sets, then there is a polynomial-time k-approximation algorithm for set cover *by using the linear programming relaxation* obtained by removing the integrality constraints. Specifically, you should describe how to obtain a solution from the relaxed linear program without integrality constraints, show your algorithm works (*i.e.* all items will be covered), and prove the approximation factor is k.

(b) Consider taking the linear programming relaxation of the above ILP and solving it to obtain a vector $x^* \in [0,1]^m$ (*i.e.* m real numbers between 0 and 1, inclusive). Now define $p \in [0,1]^m$ by

$$p_i := \min\{1, \beta \cdot x_i\}$$

We interpret the p_i as probabilities and randomly take set S_i with probability p_i and do not take it otherwise. Show that the *expected* number of sets we take is at most $\beta \cdot \mathsf{OPT}$, where OPT is the number of sets taken in an optimal set cover.

(c) Consider the same process as in part (b). How do we know that the sets we take actually form a set cover?! Show that if for any particular item, the probability that it is not covered by at least one of the sets we randomly take is at most $e^{-\beta}$.

Hint: for any $z \in \mathbb{R}$, recall from class that $1 + z \leq e^z$ (you may use this fact without proof).

$$\min \sum_{i=1}^{m} x_i$$

s.t.

$\forall u \in \mathcal{U}$, $\sum_{i:\, S_i \ni u} x_i \geq 1$
$\forall 1 \leq i \leq m$, $0 \leq x_i \leq 1$
$\forall 1 \leq i \leq m$, x_i is an integer

Figure 2: The integer linear program representing set cover is copied here for your convenience.

(d) Now upper bound the probability that the subcollection we take *fails* to cover all items in the universe. Conclude that if we set $\beta = \ln(10n)$, then the probability that the subcollection of sets we take is a set cover is at least 90%. You may use the result from part (c) without proof.

Blank scratch page.