CS 161 Final Exam

Do not turn this page until you are instructed to do so!

Instructions: Solve all questions to the best of your abilities. You may cite any result we have seen in class or CLRS without proof. You have **180 minutes** to complete this exam. You may use two handwritten double-sided sheet of notes that you have prepared yourself. You may not use any other notes, books, or online resources. Please write your name at the top of all pages. Anything written on the reverse side of a page will **not** be graded, but you may use the reverse sides as scratch paper.

Advice: If you get stuck on a problem, move on to the next one. Pay attention to how many points each problem is worth. Read the problems carefully.

The following is a statement of the Stanford University Honor Code:

1. *The Honor Code is an undertaking of the students, individually and collectively:*
   (1) *that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;*
   (2) *that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.*
2. *The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.*
3. *While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.*

By signing your name below, you acknowledge that you have abided by the Stanford Honor Code while taking this exam.

Signature: ______________________________

Name: ______________________________

SUNetID: ______________________________

1 Multiple Choice (9 points)

Box all answer choices that apply. No explanations required. Ambiguous answers will be marked incorrect. **We are expecting:** Clearly boxed answers around the letter choices of each problem. If you think none of the answers are correct, you must box option (e) to receive credit.

1.1. **(3 pt.)** Which of the following algorithms runs in $O(n^2)$ time? Select all that apply. (Note: A fully-connected graph is a graph in which there is an edge between every pair of nodes.)

(a) Bellman-Ford on a fully connected graph with n nodes.

(b) Kosaraju's algorithm on a fully connected graph with n nodes.

(c) Floyd-Warshall on a connected graph with n nodes and exactly 3 edges per node.

(d) Dijkstra's algorithm on an acyclic graph with n nodes and up to 7 positive-weight edges per node.

(e) None of the above.

1.2. **(3 pt.)** Let U be the set of positive integers. Which of the following are universal hash families mapping U to the buckets $\{0, 1\}$? Select all that apply.

(a) $H = \{h \mid h : U \to \{0, 1\}\}$. That is, H is the family of all hash functions mapping U to $\{0, 1\}$

(b) $H = \{f(x), g(x)\}$, where $f(x)$ maps even numbers into the 0 bucket and odd numbers into the 1 bucket, while $g(x)$ maps even numbers into the 1 bucket and odd numbers into the 0 bucket.

(c) $H = \{h(x)\}$, where $h(x)$ converts the input x into binary and places x into the bucket corresponding to the least significant digit of $x's$ binary representation.

(d) Any hash family H with the following property: For any given pair of distinct $u, v \in U$, and a randomly chosen function $h \in H$, the probability $P[h(u) = h(v)] = \frac{11}{23}$.

(e) None of the above.

1.3. **(3 pt.)** Which of the following are not characteristics of wicked problems? Select all that apply.

(a) Their solutions are either correct or incorrect.

(b) They have no definitive and exhaustive formulation,

(c) They are symptomatic of, and interconnected with, other problems.

(d) Solutions may be tested as many times as is necessary.

(e) None of the above.

2 Short Answers (30 points)

Describe an algorithm for each task. No pseudocode, justification, or runtime analysis required. Your description must fit in the space provided. If you are writing a lot, look for a simpler algorithm.

Your algorithm must satisfy the runtime either in the worst-case (if deterministic) or in expectation (if randomized). **You may cite or modify any algorithm from lecture, section, or homework, and only describe the parts you modify.** For all input graphs $G = (V, E)$, let $n = |V|$ and $m = |E|$.

2.0. **(0 pt.)** *(Example.)* Let G be an directed unweighted connected graph. Given G, find the pair of nodes s and t with the largest shortest path distance in $O(mn)$ time.

From every node s, run a BFS and record the node t that is last discovered. Out of the n pairs of nodes (s, t) recorded, return the pair with the largest distance.

For 2.1 and 2.2: Let $G = (V, E)$ be an undirected weighted graph with positive weights w_i. Each edge i takes one hour and costs w_i dollars to traverse. Let $s, t \in V$ be distinct nodes.

2.1. **(3 pt.)** Given G, s, and t, find the cheapest path (i.e. path costing fewest number of dollars) from s to t. Your algorithm should run in $O(m + n \log n)$ time.

We are expecting: A 1 sentence English description. No pseudocode or justification.

2.2. **(6 pt.)** Given G, s, and t, find the cheapest path (i.e. path costing fewest number of dollars) from s to t that takes at most 100 hours to traverse. Your algorithm should run in $O(m)$ time.

We are expecting: A 1-2 sentence English description. No pseudocode or justification.

2.3. **(6 pt.)** Let $G = (V, E)$ be a directed weighted graph with possibly negative weights, and $V = [v_1, v_2, \cdots, v_n]$. The *all-pairs shortest path matrix* (APSPM) of G is an $n \times n$ matrix A where $A[i, j]$ is the shortest path distance between v_i and v_j.

Given G, its APSPM A, and a new edge $e = (u \to v) \notin E$ with weight w, find the APSPM of G after adding e to E. Assume that there no negative cycles before or after e is added.

Your algorithm should run in $O(n^2)$ time.

We are expecting: A 1-3 sentence English description. No pseudocode or justification.

2.4. **(6 pt.)** Let $G = (V, E)$ be an undirected unweighted graph. Given G and distinct non-adjacent $s, t \in V$, find the most number of round trips you can make from s to t and back to s without visiting any node (other than s and t) more than once. *(i.e. once you visit a node x that is not s or t, you may never visit x again in the same or in future round trips.)*

Your algorithm should run in $O(mn)$ time.

[**Hint**: *start by turning G into a flow graph/network.*]

We are expecting: A 1-3 sentence English description. No pseudocode or justification.

2.5. **(6 pt.)** A directed unweighted graph $G = (V, E)$ with n nodes represents a social media platform Baaahtter with n sheep. The edge $a \to b$ means b follows a, so when a makes a post, b receives that post and b also re-posts the post which all of b's followers receives, etc.

Plucky has created a Baaahtter account and has not followed anyone yet. Given G, find the smallest set of accounts Plucky can follow so that Plucky eventually receives every unique post ever posted on Baaahtter.

Your algorithm should run in $O(m + n)$ time.

We are expecting: A 1-3 sentence English description. No pseudocode or justification.

2.6. **(3 pt.)** This problem can be completed without having completed 2.5. Answer the following:

- What is **one** value you are optimizing for in an algorithm for 2.5?
- Compare Plucky's problem in 2.5 to the real-world problem of deciding what users to follow on social media platforms. What is **one** other value that may be considered when solving this real-life problem that was not considered when designing an algorithm for 2.5?
- Describe **one** aspect of the real-world problem that is characteristic of wicked problems.

We are expecting: 1 sentence answers for each bullet point.

3 Algorithm Design (36 points)

From all the chocolates that Plucky bought from Lucky on the midterm, Lucky is now a very rich lemur. Lucky wants to spend some of that money on fruit from the island shop so that he can enjoy his beach-side retirement.

Lucky has a budget of B dollars, and there are n types of fruit. The i^{th} type of fruit costs c_i dollars, and makes Lucky happier by v_i. Lucky wishes to buy fruits such that he maximizes his happiness. The island shop has infinite copies of each fruit, and it is given to you that for each i, c_i and v_i are both integers such that $1 \leq c_i \leq C$ and $1 \leq v_i \leq V$.

3.1. **(3 pt.)** If Lucky can buy at most 1 of each type of fruit, mention an algorithm seen in class that solves this problem in $O(n \cdot B)$.

We are expecting: Reference to an algorithm we saw in class or a description of how to solve the problem.

3.2. **(3 pt.)** If Lucky can buy any number of fruits of each kind, mention an algorithm seen in class that solves this problem in $O(n \cdot B)$.

We are expecting: Reference to an algorithm we saw in class or a description of how to solve the problem.

3.3. **(15 pt.)** Prove the following:

Consider an optimal solution $S = [x_1, \ldots, x_n]$ where x_a is the number of times that fruit a appears in the solution. If $x_i \geq C$, $x_j \geq C$, and $\frac{v_i}{c_i} \geq \frac{v_j}{c_j}$, removing c_i copies of fruit j and adding c_j copies of fruit i gives a **valid** and **optimal** solution S'.

We are expecting: A rigorous proof that the S' defined above is both valid and optimal.

[Extra room for solution to 3.3]

3.4. **(15 pt.)** Via the result in 3.3, one can show the following:

Lemma 1. *There exists an optimal solution such that for at most one fruit i, $x_i \geq C$.*

Use one of the algorithms from 3.1 and 3.2 in combination with Lemma 1 to design a $O(C^2n^2)$ solution.

Note that compared to 3.1 and 3.2 which had runtimes of $O(Bn)$, the runtime here is *not* a function of B — your solution must run in $O(C^2n^2)$ even if B is much larger than C^2n.

We are expecting: Pseudocode or a clear English description of your algorithm. An argument as to why the algorithm is correct and runs in $O(C^2n^2)$.

[Extra room for solution to 3.4]

4 Algorithm Analysis (25 points)

4.1. **(10 pt.)** Let C be any cycle in a connected, weighted, undirected graph G, and let $\{u, v\}$ be the edge in that cycle with the largest weight. Prove that there exists an MST of G that does not include edge $\{u, v\}$.

We are expecting: A rigorous proof.

4.2. **(15 pt.)** Next, consider the following algorithm for finding a minimum spanning tree (MST) in a connected, weighted, undirected graph $G = (V, E)$.

Algorithm 1: findMST

```
input: $G$: connected, undirected, weighted graph
while there is a cycle in $G$ do
    let $C$ be any cycle in $G$
    remove the largest-weight edge from $C$
return $G$
```

That is, while the algorithm can find a cycle in G, it deletes the edge with the largest weight in that cycle. When it can no longer find a cycle, then it returns whatever is left.

Prove that the *findMST* algorithm above is correct using the fact we proved in 4.1.

We are expecting: A rigorous proof by induction.

[Extra room for solution to 4.2]

[End of final exam.]