CS161 Practice Final Exam

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

Instructions: Answer as many problems as you can. You may cite any result we have seen in class or the textbook without proof. You have **180 minutes** to complete this exam. You may use two two-sided sheets of notes that you have prepared yourself. You may not use any other notes, books, or online resources. There is one blank page at the end for extra work. **Please write your name at the top of all pages.**

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: _____

The following summation formulas may or may not be useful for this exam:

$$\sum_{i=0}^{n-1} i = \Theta(n^2) \qquad \sum_{i=1}^{n-1} \frac{1}{i} = \Theta(\log(n)) \qquad \sum_{i=0}^{n-1} 2^{-i} = \Theta(1)$$

Section	1 (mult. ch.)	2 (short answer)	3 (alg. design)	4 (proofs)	5 (challenge)	Total
Score						
Maximum	28	24	33	15	5	105

1 Multiple Choice (28 pts)

No explanation is required for multiple-choice questions and no partial credit will be given. Please clearly mark your answers; if you must change an answer, either erase thoroughly or else make it **very** clear which answer you intend. **Ambiguous answers will be marked incorrect.**

1.1. **(10 pt.)** For each recurrence relation, choose the expression (A)-(E) below which best describes it and **write your choice (A)-(E)** in the blank.

Note that **not all options need to be used,** and some options may be used more than once. Assume that $T(1) = O(1)$ and don't worry about floors and ceilings.

(A) $\Theta(n)$ (B) $\Theta(n^2)$ (C) $\Theta(n \log n)$ (D) $\Theta(n^2 \log(n))$ (E) $\Theta(\log(n))$

1.1.1. **(2 pt.)** _____: $T(n) = 2T(n/2) + n^2$

1.1.2. **(2 pt.)** _____: $T(n) = 4T(n/2) + n^2$

1.1.3. **(2 pt.)** _____: $T(n) = T(n-3) + n$

1.1.4. **(4 pt.)** _____: $T(n) = T(n/3) + T(n/3) + T(n/10) + n$

1.2. **(12 pt.)** For each quantity, choose the *smallest* expression (A)-(F) below which describes it and **write your choice (A)-(F)** in the blank.

Note that **not all options need to be used,** and some options may be used more than once.

(A) $O(\log n)$ (B) $O(n)$ (C) $O(n \log n)$ (D) $O(n^2)$ (E) $O\left(n^2 \log(n)\right)$ (F) $O(n^3)$

1.2.1. **(2 pt.)** _____: Time to find the $n/10$'th biggest element in an unsorted array of length n.

1.2.2. **(2 pt.)** _____: Time to find a shortest path from vertex s to vertex t in a connected, directed, unweighted graph with n vertices and $\Theta(n^2)$ edges.

1.2.3. **(2 pt.)** _____: Time to find a shortest path from vertex s to vertex t in a connected, directed, *weighted* graph with n vertices and $\Theta(n)$ edges, and **no** negative edge weights.

1.2.4. **(2 pt.)** _____: Time to find the SCCs of a directed graph G with n vertices and $\Theta(n^2)$ edges.

1.2.5. **(2 pt.)** _____: Time to find a longest common subsequence of two strings of length n.

1.2.6. **(2 pt.)** _____: Time to sort an array of length n, which contains integers between 1 and 100 (possibly with repeats).

1.3. **(6 pt.)** Suppose that $N(n)$ is the number of binary strings of length n that do not have two 0's in a row. For example, $N(3) = 5$, because there are 5 binary strings of length 3 with no repeated zeros: $011, 101, 110, 010, 111$.

Which of the following is a correct recurrence relation for N (for $n \geq 3$)? **Circle exactly one answer.**

(A) $N(n) = N(n-1) + N(n-2)$

(B) $N(n) = 2N(n-1)$

(C) $N(n) = N(n-1) + 2N(n-2)$

(D) $N(n) = N(n-2) + 1$

(E) $N(n) = N(n-1) + 2N(n-2) + 1$

(This space intentionally blank)

The following definition may be helpful for the short answer questions on the next page:

Definition: The **diameter** of a strongly connected directed graph $G = (V, E)$ is the largest distance between any two vertices in V. That is,

$$\text{diam}(G) = \max_{u,v \in V} \text{dist}(u, v),$$

where as usual the distance $\text{dist}(u, v)$ between vertices $u, v \in V$ is the cost of a shortest[1] directed path from u to v; in an unweighted graph, it is the number of edges on a shortest directed path from u to v.

[1]For this definition, it is okay if the "distance" is negative if there are negative edge weights in G; assume that there are no negative cycles in G.

2 Short answers (24 pts)

For the questions in this section **you do not need to write pseudocode (unless you want to) or give any formal proofs.** Your answers should fit easily in the space provided—don't waste time writing too much!

2.1. **(6 pt.)** Suppose that G is an *unweighted* strongly connected directed graph on n vertices with m edges. Explain how to find the diameter of G in time $O(nm)$. (Note that *diameter* is defined on the previous page).

[**We are expecting:** *A short English description of your algorithm* ***OR*** *pseudocode.*]

2.2. **(6 pt.)** Suppose that G is a *weighted* strongly connected directed graph on n vertices with m edges. G may have negative edge weights, but suppose that it does not have a negative cycle. Explain how to find the diameter of G in time $O(n^3)$. (Note that *diameter* is defined on the previous page).

[**We are expecting:** *A short English description of your algorithm* ***OR*** *pseudocode.*]

2.3. **(6 pt.)** Suppose that you have access to a genie which does the following in time $O(1)$:

- Input: an array A of length n, containing arbitrary comparable elements.
- Output: with probability $1/2$, it outputs the maximum element of A. With probability $1/2$, it outputs an aelement of A which may or may not be the maximum.

Design a randomized algorithm which uses the genie and finds the maximum of the array. Your algorithm should have worst-case running time $O(1)$, and be correct with probability $1 - 1/2^{10}$.

[**We are expecting:** *A short English description of your algorithm* ***OR*** *pseudocode.*]

2.4. **(6 pt.)** Suppose that G is a directed acyclic graph with n vertices and m edges. An *exact traversal* of G is a path which touches each vertex of G exactly once. Design an algorithm which decides whether or not G has an exact traversal, which runs in time $O(n + m)$.

[**We are expecting:** *A short English description of your algorithm* ***OR*** *pseudocode.*]

3 Algorithm Design (33 points)

3.1. **(11 pt.) (Making change.)** The currency in the country CS161land is cents. In CS161land, there is a p_0-cent coin, and p_1-cent coin, and so on. Let $P = \{p_0, \ldots, p_m\}$ be this set of coin values.

Design a $O(n|P|)$-time **dynamic programming** algorithm which takes as input the set P, along with a positive integer n, and outputs the minimal number of coins needed to represent n cents in CS161land.

For example, if $P = \{1, 6, 10\}$ (aka, there are one-cent, six-cent, and 10-cent coins), the best way to make $n = 12$ cents is with two six-cent coins, so your algorithm should return 2.

You may assume that $1 \in P$ (that is, there is a one-cent coin).

3.1.1. **(6 pt.)** What sub-problems are you using in your dynamic programming algorithm? What is the recursive relationship that they satisfy and with what base case(s)?

[**We are expecting:**

- *A clear English description of the sub-problems.*
- *A mathematical statement of the recursive relationship that they satisfy.*
- *Any necessary base case(s).*

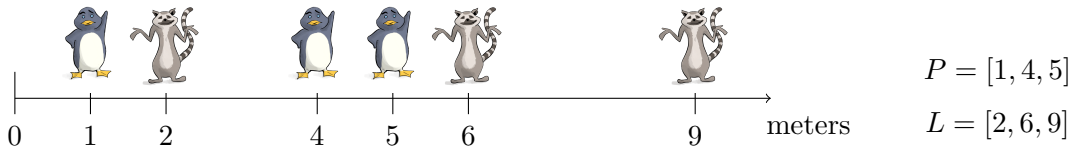
]

[Another part on next page]

3.1.2. **(5 pt.)** Write pseudocode for your algorithm and explain (in a sentence or two) what it does.

[**We are expecting:** *Pseudocode* ***AND*** *and an English description of what your algorithm is doing. You do not need to justify the correctness or the running time.*]

3.2. **(11 pt.) (Penguins and Lemurs.)** Suppose there are n penguins and n lemurs arranged along a line. There is a sorted array P which gives the locations of the penguins, and a sorted array L which gives the locations of the lemurs. The locations are measured in meters from a starting point on the left. For example, one set-up for $n = 3$ could be:



A penguin and a lemur can be matched together (say, to work on a course project) if they are within t meters of each other. For example, in the above, if $t = 2$, the lemur at 2 could be matched with the penguins at 1 or 4, but not with the penguin at 5.

Design an algorithm which takes as input P, L and t and outputs the maximum number of penguin-lemur matches which can be made. The algorithm should run in time $O(n)$.

For example, if $t = 2$ and P, L are as in the above example, the algorithm should return 2. If $t = 10$ and P, L are as in the above example, the algorithm should return 3.

You may assume that all of the values in P and L are distinct, and that both P and L are sorted in increasing order.

[**Hint:** *Try a greedy algorithm.*]

[**We are expecting:** *Pseudocode* ***AND*** *an English description of what your algorithm is doing. You do not need to prove that the algorithm is correct or justify its running time.*]

[More space on next page]

[More space for problem 3.2.]

3.3. **(11 pt.) (Missing duck.)** Let $n = 2^t$ for some integer t, so that there are exactly n binary strings of length t. You encounter a population of n ducks, each of which has a unique t-bit ID. For example, if $n = 8$, then $t = 3$ and there are 8 ducks with IDs $000, 001, 010, 011, 100, 101, 110, 111$.

You cannot ask a duck directly what its ID is, but you can call a function `getBit(duck,i)`, for $i = 0, \ldots, t-1$, which will return the i'th least significant bit of `duck`'s ID number. For example, if Dolly the Duck has ID 011, then `getBit(dolly,0)` would return 1 and `getBit(dolly,2)` would return 0.

The ducks are standing in a line (not necessarily in any sorted order), but there are only $n-1$ of them! One of them is missing, and you don't know which one. Give a deterministic algorithm which will query the remaining $n-1$ ducks to find the ID of the missing duck, using $O(n)$ calls to `getBit`.

[**We are expecting:**

- *Pseudocode* ***AND*** *a clear English description of what your algorithm is doing.*
- *A justification that your algorithm uses only $O(n)$ calls to* `getBit`*.*
- *You do not need to prove that your algorithm is correct.*

]

[More space on next page]

[More space for Problem 3.3.]

4 Proving stuff (15 pts)

4.1. **(15 pt.)** Consider the following greedy algorithm which finds a minimum spanning tree in a graph G.

Algorithm 1: MyMST

Input: A connected, weighted, undirected graph $G = (V, E)$.
$\mathcal{S} \leftarrow \{\{v\} \ : \ v \in V\}$
$T \leftarrow \emptyset$
while $|\mathcal{S}| > 1$ **do**
  Choose a random $C \in \mathcal{S}$.
  Let $e = \{u, v\}$ be the edge in E with the smallest weight so that $u \in C$ and $v \notin C$.
  Add e to T.
  Suppose that $v \in C'$ for $C' \in \mathcal{S}$ (so $C' \neq C$).
  Remove C and C' from $\mathcal{S}$.
  Add $C \cup C'$ to $\mathcal{S}$.
return T

In words, the algorithm keeps a collection $\mathcal{S}$ of components. At each step, it chooses a component $C \in \mathcal{S}$ at random, and finds the edge $\{u, v\}$ with the smallest weight leaving C and adds $\{u, v\}$ to the growing MST T. If the other endpoint of this edge is in some other component C', the algorithm then replaces C and C' with their union $C \cup C'$ in $\mathcal{S}$.

Prove that this algorithm *always* correctly finds a minimum spanning tree (no matter how the randomness plays out). You may use any statement we have proved in class without proving it again.

[**We are expecting:** *A formal proof. If you use induction, be sure to state your inductive hypothesis, base case, inductive step, and conclusion.*]

[More space on next page]

[More space for problem 4.1.]

5 One more problem

Note: This problem may be more difficult, and it is only worth 5 points. You might want to do the rest of the exam first.

5.1. **(5 pt.)** Prove or disprove:

> There exists an M so that the following holds:
> Let $\mathcal{U} = \{1, 2, \ldots, M\}$ be a universe of size M. Then there is a universal hash family $\mathcal{H}$ consisting of functions $h : \mathcal{U} \to \{0, 1\}$ (aka, $\mathcal{H}$ hashes $\mathcal{U}$ into two buckets), so that $|\mathcal{H}| \leq \log(M) - 1$.

[**We are expecting:** *Whether the statement is true or false, and formal proof to back it up.*]

This is the end!

This page intentionally blank for extra space for any question.
Please indicate in the relevant problem if you have work here that you want graded, and label your work clearly.