# Midterm 2 for CS 170

PRINT your name: ,

(last) (first)

SIGN your name:

WRITE your section number (e.g., 101):

WRITE your SID:

One page of notes is permitted. No electronic devices, e.g. cell phones and calculators, are permitted. Do all your work on the pages of this examination. If you need more space, you may use the reverse side of the page, but try to use the reverse of the same page where the problem is stated.

You have 60 minutes. The questions are of varying difficulty, so avoid spending too long on any one question.

In all algorithm design problems, you may use high-level pseudocode.

DO NOT TURN THE PAGE UNTIL YOU ARE TOLD TO DO SO.

| Problem | Score/Points |
|---|---|
| 1 | /5 |
| 2 | /9 |
| 3 | /5 |
| 4 | /10 |
| 5 | /10 |
| 6 | /10 |
| Total | /49 |

## 1. True/False

For each statement below, say whether it is true or false by circling TRUE or FALSE. You do not need to provide any explanation for your answer.

(a) TRUE or FALSE: The family of hash functions $h_{a,b}(x) = ax + b \bmod m$, where $a \neq 0$ always forms a 2-universal hash family.

(b) TRUE or FALSE: If we use path compression and rank-by-union heuristics, then every FIND operation runs in $O(\log^* n)$ time, where $n$ is the number of elements.

(c) TRUE or FALSE: If we just use rank-by-union heuristics, then every FIND operation runs in $O(\log n)$ time, where $n$ is the number of elements.

(d) TRUE or FALSE: $\log^* \log n = O(\log \log^* n)$.

(e) TRUE or FALSE: A linear program can have infinitely many optimal solutions.

(f) TRUE or FALSE: You filled in your name, your TA's name, and the section (either section number or section time) on the first page of this test.

## 2. True/False with Explanations

For each statement below, say whether it is true or false, *and provide a one-sentence and/or one-picture explanation.*

(a) TRUE or FALSE: If the frequency of characters are unique, then the optimal prefix code is unique.

(b) TRUE or FALSE: A 2-universal hash family can contain a hash function $h(x)$ that maps every element $x$ to 0.

(c) TRUE or FALSE: Let $G$ be an undirected, connected, weighted graph. Split $G$ into two equal halves, compute the MST of both halves, and connect the two MSTs with the shortest edge possible. The resulting tree is a MST of $G$.

## 3. A Huffman Code

Give the Huffman code for the character set $\{(a, 1/8), (b, 1/8), (c, 1/4), (d, 1/2)\}$, where the number following the character is its frequency.

What is the expected number of bits/character for a file with the frequencies as specified above?

## 4. Solving Simple Equations

Suppose we are given a set of $n$ variables $x_1, x_2, \ldots, x_n$, each of which can take one of the values in the set $\{0, 1\}$. We are also given a set of $k$ equations; the $r$th equation has the form

$$(x_i + x_j) \bmod 2 = b_r,$$

for some choice of two distinct variables $x_i$, $x_j$, and for some value $b_r$ that is either 0 or 1. Thus each equation specifies whether the sum of two variables is even or odd.

Consider the problem of finding an assignment of values to variables that maximizes the number of equations that are satisfied. Let $c^*$ denote the maximum possible number of equations that can be satisfied by an assignment of values to variables. Give a polynomial-time randomized algorithm such that the expected number of equations that are satisfied is at least $\frac{1}{2}c^*$. (Prove that your algorithm satisfies this bound).

## 5. Sums of Small Numbers

Given a nonnegative integer $n$, we want to determine how many strings of 1's, 2's, and 3's add up to $n$. For example, with $n = 3$ we have the strings 111, 12, 21, and 3—four of them. With $n = 5$ we have thirteen: 11111, 1112, 1121, 113, 1211, 122, 131, 2111, 212, 221, 23, 311, and 32.

(a) Let $T(n)$ be the number of such strings adding to $n$. Give a recurrence for $T(n)$ when $n$ is positive, based on the fact that the first digit in the string may be 1, 2, or 3. What is $T(0)$?

(b) Describe an efficient algorithm to compute $T(n)$. What is its big-O running time as a function of $n$?

## 6. Counting Connected Components

Suppose we start with a graph with $n$ vertices but no edges. The number of connected components is then $n$. Now suppose we add $m$ edges, $e_1, e_2, \ldots e_m$, to the graph. Let $G_k$ denote the graph obtained by adding $k$ edges $e_1, \ldots, e_k$. Give a $o((m+n)\log n)$ algorithm to compute the number of connected components at each step $k$.