CS 170 HW 5

Due **2020-10-05, at 10:00 pm**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write none.

In addition, we would like to share correct student solutions that are well-written with the class after each homework. Are you okay with your correct solutions being used for this purpose? Answer "Yes", "Yes but anonymously", or "No"

2 Huffman Coding

This question is a solo question.

In this question we will consider how much Huffman coding can compress a file F of m characters taken from an alphabet of $n = 2^k$ characters x_0, x_1, ... , x_{n-1} (each character appears at least once).

(a) Let $S(F)$ represent the number of bits it takes to store F without using Huffman coding (i.e., using the same number of bits for each character). Represent $S(F)$ in terms of m and n.

(b) Let $H(F)$ represent the number of bits used in the optimal Huffman coding of F. We define the *efficiency* $E(F)$ of a Huffman coding on F as $E(F) := S(F)/H(F)$. Among all files with m characters and alphabet size n, describe a file F for which $E(F)$ is as small as possible.

(c) Among all files with m characters and alphabet size n, describe a file F for which $E(F)$ is as large as possible. How does the largest possible efficiency increase as a function of n? Give you answer in big-O notation.

3 Preventing Conflict

A group of n guests shows up to a house for a party, but m pairs of these guests are enemies (a guest can be enemies with multiple other guests). There are two rooms in the house, and the host wants to distribute guests among the rooms, breaking up as many pairs of enemies as possible. The guests are all waiting outside the house and are impatient to get in, so the host needs to assign them to the two rooms quickly, even if this means that it's not the best possible solution. Come up with a $O(n + m)$-time algorithm that breaks up at least half of the pairs of enemies. **Give a three-part solution.**

The remaining questions on this homework are optional and intended as additional resources for preparing for the midterm. They are not necessarily of an equivalent difficulty to the midterm problems, and will not be graded. Feel free to discuss these problems and their solutions publicly, including on Piazza.

4 (Optional) The Resistance

We are playing a variant of The Resistance, a board game where there are n players, k of which are spies. In this variant, in every round, we choose a subset of players to go on a mission. A mission succeeds if no spies are chosen to go on the mission, but fails if at least one spy goes on the mission, and when a mission fails we are not told who the spies are that went on the mission.

Come up with a strategy that identifies all the spies in $O(k \log(n/k))$ missions. Only a main idea and runtime analysis are needed.

5 (Optional) Cyclic Shifts

Given an n-element vector u, the jth cyclic shift of u is the vectors $u^{(j)}$ defined as $u_i^{(j)} = u_{(i+j) \mod n}$ for some j (using 0-indexing). For example, the cyclic shifts of $u = (0, 1, 2)$ are $u^{(0)} = (0, 1, 2), u^{(1)} = (1, 2, 0)$, and $u^{(2)} = (2, 0, 1)$.

Given two n-element vectors u, v, give an efficient algorithm that computes the n dot products $u^{(0)} \cdot v, u^{(1)} \cdot v, \ldots, u^{(n-1)} \cdot v$. Just the main idea and runtime analysis are needed.

Hint: Recall the Protein Matching problem from HW3. Is there a short vector u' such that every cyclic shift of u is contained in u'?

6 (Optional) Finding Ancestors

You are given a tree $T = (V, E)$ with a designated root node r. For each vertex v, let $a(v)$ be the kth ancestor of v in the tree (so for $k = 1$ this is the parent of v, for $k = 2$ this is v's parent's parent, etc.). We follow the convention that the root node, r, is its own parent. Modify DFS so that it takes r, k in as input, and computes $a(v)$ for all vertices in $O(|V| + |E|)$ time. Just a main idea is needed.

7 (Optional) Shortest Path Between Sets

Given a undirected weighted graph G with non-negative edge weights, let $d(s, t)$ be the shortest path length from s to t. Give an efficient algorithm that takes as input two subsets of vertices S and T and outputs $\min_{s \in S, t \in T} d(s, t)$, i.e. the shortest path from any vertex in S to any vertex in T. Give a three-part solution.