CS 170 Homework 3

Due **9/21/2020, 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.

2 Exam Policy

Please read the exam policy document that was distributed to all students and then do the following:

(a) Submit a Zoom meeting link that you will use during the exam here:

https://tinyurl.com/y6zdtkd5

(b) Follow the steps in the setup checklist in the exam policy document. As your answer to this question, please describe your setup for taking the exam, where you plan to sit, and any other relevant details in a few sentences.

(c) Please record a 5min sample recording following the Set-Up Checklist and After the Exam sections. Submit the link to this recording here:

https://forms.gle/ovsWX1SSxR93rCyv9

3 Triple sum

We are given an array $A[0..n-1]$ with n elements, where each element of A is an integer in the range $0 \leq A[i] \leq n$ (the elements are not necessarily distinct). We would like to know if there exist indices i, j, k (not necessarily distinct) such that

$$A[i] + A[j] + A[k] = n$$

Design an $\mathcal{O}(n \log n)$ time algorithm for this problem. Note that you do not need to actually return the indices; just yes or no is enough. *Hint: Use FFT/fast polynomial multiplication.*

Please give a 3-part solution to this problem.

4 Protein Matching

Often times in biology, we would like to locate the existence of a gene in a species' DNA. Of course, due to genetic mutations, there can be many similar but not identical genes that serve the same function, and genes often appear multiple times in one DNA sequence. So a more practical problem is to find all genes in a DNA sequence that are similar to a known gene.

To model this problem, let g be a length-n string corresponding to the known gene, and let s be a length-m string corresponding to the full DNA sequence, where $m \geq n$. We would like to solve the following problem: find the (starting) location of all length n-substrings of s which match g in at least $n - k$ positions. For example, using 0-indexing, if $g = ACT$, $s = ACTCTA$, and $k = 1$ your algorithm should output $[0, 2]$.

(a) Give a $O(nm)$ time algorithm for this problem.

(b) Assume g and s are given as bitstrings, i.e. every character is either 0 or 1. Give a $O(m \log m)$ time algorithm that works for any k.

Hint: Represent the strings as vectors, and use FFT/fast polynomial multiplication.

You do not need a 3-part solution for each part. Instead, describe the algorithms clearly and give an analysis of the running time.

5 Practice with SCCs

This question is a "solo question". It is meant as a way for you to check how well you would do on a question of similar difficulty if it appeared on an exam. Do not collaborate with other students on this problem, and staff may give less help on this problem than they do on other homework problems.

(a) Design an efficient algorithm that given a directed graph G, outputs the set of all vertices v such that there is a cycle containing v. In other words, your algorithm should output all v such that there is a non-empty path from v to itself in G.

(b) Design an efficient algorithm that given a directed graph G determines whether there is a vertex v from which every other vertex can be reached. (Hint: first solve this for directed acyclic graphs. Note that running DFS from every single vertex is not efficient.)

Please give a 3-part solution to both parts of this problem. For both parts, try to keep each part of the 3-part solution no longer than a few sentences.

6 Splitting Edges

Let $T = (V, E)$ be an undirected connected tree. That is, T has no cycles and there is a path from every vertex to every other vertex in T. For each edge $e \in E$, $T \setminus e$ has exactly two connected components. Let us call the sizes of these two components the *split numbers* of e.

(a) Describe a linear-time algorithm to compute the split numbers of all edges $e \in E$, and prove its runtime (proof of correctness not required).

(b) For any $u, v \in V$, define $d(u, v)$ to be the number of edges on the unique path from u to v in T. Suppose we've already computed the split numbers for each edge $e \in E$. Now, given only the split numbers and not the original graph, describe how to compute the total pairwise distance between all vertices

$$R := \sum_{u,v \in V} d(u,v)$$

More specifically, if the edge e has split numbers x_e, y_e, then you're only given all (x_e, y_e) values as input. Justify your answer.

Hint: for a given edge $e \in E$, how many times does it contribute to the above sum?

(c) A perfect matching is a set of edges $M \subseteq E$ such that every vertex in V is incident to *exactly* one edge in M. Again define the split numbers of e to be x_e, y_e.

Let T be a tree that has a perfect matching M. Prove that

$$M = \{e \in E : x_e \text{ is odd and } y_e \text{ is odd}\}$$

Note that not all trees have perfect matchings - you only want to prove this property holds for T that does have a perfect matching.

7 Vertex Separators

Let $G = (V, E)$ be an undirected, unweighted graph with $n = |V|$ vertices. We call a set of vertices S a $u - v$ *separator* if (1) S does not contain u or v and (2) deleting all vertices in S and all edges adjacent to these vertices from G disconnects u and v. In other words, S is a $u - v$ separator if every path from u to v goes through some vertex in S.

Give an efficient algorithm that takes G, u, v as input and finds a $u - v$ separator of size at most $\frac{n-2}{d-1}$, where d is the length of the shortest path from u to v. Assume that u and v are connected in G and $d > 2$. **Give a three-part solution.**

Hint: Is there a natural way to partition (some of) the vertices that aren't u, v into $d - 1$ sets?

8 The Greatest Roads in America

Arguably, one of the best things to do in America is to take a great American road trip. And in America there are some amazing roads to drive on (think Pacific Crest Highway, Route 66 etc). An intrepid traveler has chosen to set course across America in search of some amazing driving. What is the length of the shortest path that hits at least k of these amazing roads?

Assume that the roads in America can be expressed as a directed weighted graph $G = (V, E, d)$, and that our traveler wishes to drive across at least k roads from the subset $R \subseteq E$ of "amazing" roads. Furthermore, assume that the traveler starts and ends at her home $h \in V$. You may also assume that the traveler is fine with repeating roads from R, i.e. the k roads chosen from R need not be unique.

Design an efficient algorithm to solve this problem. Provide a 3-part solution with runtime in terms of $n = |V|$, $m = |E|$, k.

Hint: Create a new graph G' *based on* G *such that for some* s', t' *in* G'*, each path from* s' *to* t' *in* G' *corresponds to a path of the same length from* h *to itself in* G *containing at least* k *roads in* R*. It may be easier to start by trying to solve the problem for* $k = 1$*.*