CS161 Practice Final Exam
Date: May 31, 2016

Instructions: Please answer the following questions to the best of your ability. If you are asked to show your work, please include relevant calculations for deriving your answer, explaining steps as you go. If you are asked to prove a result, please write a complete proof at the level expected on your homework. When writing proofs, please strive for clarity and brevity (in that order).

You have 180 minutes to complete the exam. The exam is closed-book, closed-lecture-notes, closed-internet, etc. You may use 1 sheet, front and back, of notes as reference. By signing your name below, you acknowledge that you have abided by the Stanford Honor Code while taking this exam.

Name: _____

SUNetID: _____

Signature: _____

Throughout this exam, unless explicitly stated otherwise, you may assume the following:

- Graphs are represented by adjacency lists.
- Hash functions provide simple uniform hashing. You can assume that hash tables use chaining to resolve collisions, so the expected time for look-up in a hash table is $O(\alpha)$, where α is the ratio between the number of elements stored and the number of slots in the hash table.

On all questions, *you may use any of the algorithmic machinery that we've discussed in this course.* Any time you are asked to give an algorithm to solve a problem, you are expected to prove the algorithm's correctness and runtime. Unless the problem explicitly asks for a proof, you may cite the runtime and proof of correctness of any algorithm presented in class.

Note: The above instructions are the instructions from the actual final exam. This exam is meant as practice and will not necessarily take the full 180 minutes.

1 True or False

For each question, indicate whether the statement is True or False (by circling `T` or `F`, respectively). Correct answers will receive 4 points. Blank responses will receive 2 points. Incorrect answers will receive 0 points.

`T` `F` Iterating over all pairs of vertices in a graph requires $\Omega(n^{1.5})$ time.

`T` `F` Poe gives Barr a comparison-based sorting algorithm that runs in $T(n) = 4T(n/5) + 5n \log \log n$. There is not enough information to know whether Poe's algorithm is correct.

`T` `F` Barr gives Poe a comparison-based sorting algorithm that runs in $T(n) = 7T(n/6) + n/42$. There is not enough information to know whether Barr's algorithm is correct.

`T` `F` Consider using a variant of Karger's algorithm to compute an $s-t$ cut. To make sure that s and t end up in different partitions, we will then delete any edges between the supernodes that contain s and t, and repeat until there are two supernodes. This variant of Karger's algorithm computes the min $s-t$ cut with probability $1/n^c$ for some constant c.

`T` `F` Let G be an undirected, weighted graph. Multiplying every edge weight by 2 does not change the shortest paths in G.

`T` `F` Suppose in some graph G, the closest negative cycle to a node v has all its nodes at least 5 hops away from v. Consider any node x, whose shortest path from v has fewer than 5 hops. Bellman-Ford computes $d(v, x)$ correctly.

`T` `F` Suppose G has $\Theta(n)$ edges with integer weights on the range $O(n^3)$. The runtime of Kruskal's and Prim's algorithm will be asymptotically equivalent.

2 What's Missing?

Suppose you're given a sorted array of length $n-1$ containing all but one element of $\{1, \ldots, n\}$. Give an algorithm for determining which element is missing. Prove that your algorithm is correct and runs in $O(\log n)$ time.

3 Scheduling Jobs

Suppose you're given a list of n jobs $[j_1, j_2, \ldots, j_n]$. You have two machines, M_1 and M_2 that can run these jobs. Unfortunately, you may not schedule the jobs arbitrarily. Due to resource constraints, some pairs of jobs must be scheduled on the same machine, while other pairs of jobs must be scheduled on different machines. You are given a list of n jobs and for each job j, you are given a list $\mathsf{same}(j)$ containing the jobs which must be scheduled on the same machine as j, and $\mathsf{diff}(j)$ containing the jobs which must be scheduled on the opposite machine than j. In total, there are m such scheduling constraints. Design an algorithm that produces an allocation of the jobs onto M_1 and M_2 where all the constraints are met, or reports that no such allocation is possible. Prove that your algorithm is correct and runs in $O(m + n)$ time.

4 Unique Requests

Suppose you're given a list of n web requests $r_1, \ldots, r_n$. Each request r_i contains the ID of the server to which the request was sent. You are tasked with finding the first request in the list which is sent to a unique server, or report that no such request exists. Design an algorithm that runs in expected $O(n)$ time. Prove it's correctness. You should assume that there are too many server IDs to store information about each server.

5 KT 6.27

The owners of an independently operated gas station are faced with the following situation. They have a large underground tank in which they store gas; the tank can hold up to L gallons at one time. Ordering gas is quite expensive, so they want to order relatively rarely. For each order, they need to pay a fixed price P, for delivery in addition to the cost of the gas ordered. However, it costs c to store a gallon of gas for an extra day, so ordering too much ahead increases the storage cost. They are planning to close for a week in the winter, and they want their tank to be empty by the time they close. Luckily, based on years of experience, they have accurate projections for how much gas they will need each day until this point in time. Assume that there are n days left until they close, and they need g_i gallons of gas for each of the days $i = 1, \ldots, n$. Assume the tank is empty at the end of day 0. Give an algorithm to decide on which days they should place orders, and how much to order so as to minimize their total cost.

6 k-Clustering

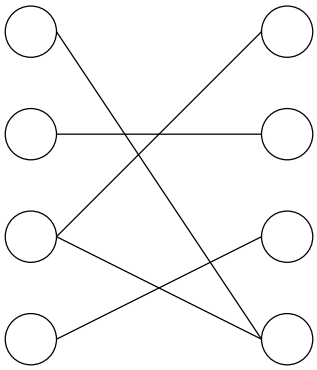
Suppose you're given n points in $\mathbb{R}^3$. We want to cluster these points into k groups such that we maximize the minimum distance between the separated points. That is, we want to find a partition of the points into $P_1, \ldots, P_k$ such that

$$\min_{\substack{p_i \in P_i, p_j \in P_j \\ i \neq j}} \|p_i - p_j\|$$

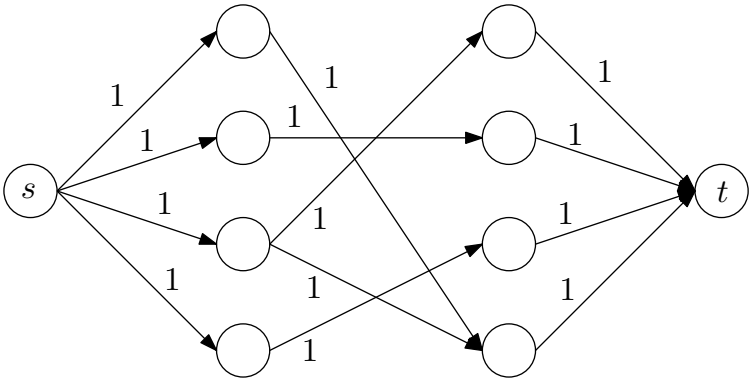
is maximized. Design an algorithm to find an optimal k-Clustering that runs in $O(n^2 \log n)$ time; prove it's correctness and runtime.

7 Flows and Matchings

Consider the bipartite graph G below.



We want to find the maximum cardinality matching in G. Recall, that we can frame this as a max flow problem by reducing G to following directed graph G'.



(a) What is the maximum flow from s to t through G'?

(b) What is the weight of the minimum st-cut in G'?

(c) How many augmenting paths will Ford-Fulkerson find while computing the maximum flow?

(d) Over all possible executions of Ford-Fulkerson, what is the longest augmenting path from s to t through the residual network G'_f?