CS 170 Homework 13

Due **Saturday 11/29/2025, at 10:00 pm (grace period until 11:59pm)**

1 Study Group (1 point)

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

2 Firefighters (10 points)

PNPLand is made of n cities that are numbered from $0, 1, \ldots, n-1$, which are connected by two-way roads. You are given a matrix D such that, for each pair of cities (a, b), $D[a][b]$ is the distance of the shortest path between a and b. The distances $D[a][b]$ are metric, so you can assume that the triangle inequality always holds: $0 \leq D[a][b] \leq D[a][c] + D[c][b]$ for all a, b, c, and also that distances are symmetric: $D[a][b] = D[b][a]$ for all a, b.

We want to pick s distinct cities and build fire stations there. For each city without a fire station, the response time for that city is given by the distance to the nearest fire station. We define the response time for a city with a fire station to be 0. Let R be the maximum response time among all cities. We want to create an assignment of fire stations to cities such that R is as small as possible.

In this problem, we'll analyze an efficient greedy approach to solving this problem that will not always give the optimal solution, but that we can prove is a "good enough" approximation.

We can formalize the problem as follows: Suppose the optimal assignment of fire stations to cities produces response time R_{opt}.

(a) (2 points) Consider the following algorithm: given positive integers n, s and the 2D matrix D as input, start by building the first fire station at an arbitrary city, e.g., at city 1. Then, for the remaining $s-1$ fire stations, repeatedly find the city with the largest response time and build a fire station there.

Analyze the runtime of this greedy algorithm.

(b) (8 points) Prove that this greedy algorithm achieves an approximation factor of 2, i.e., that it always achieves a response time of $R_g \leq 2 \cdot R_{\text{opt}}$. You may devise your own proof or follow the outline below:

Let the fire stations found by the greedy solution be at $g_1, \ldots, g_s$ and let the fire stations of some optimal solution be at $\omega_1, \ldots, \omega_s$. Assume for the sake of contradiction that $R_g > 2 \cdot R_{\text{opt}}$, i.e., there exists some city v^* whose response time is greater than $2R_{\text{opt}}$.

(i) Show for any $i \neq j$ that $D[g_i][g_j] > 2 \cdot R_{\text{opt}}$.

(ii) Show that for any i, there is at most one j such that $D[\omega_i][g_j] \leq R_{\text{opt}}$.

(iii) Show that for any i, there is at least one j such that $D[\omega_i][g_j] \leq R_{\text{opt}}$. Conclude that for any i, there is **exactly** one j such that $D[\omega_i][g_j] \leq R_{\text{opt}}$.

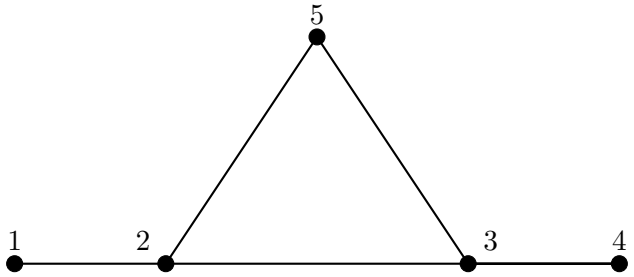
(iv) Now, consider the city v^* defined above. Say it is served by some fire station ω_i in the optimal solution (that is, $D[\omega_i][v^*] \leq R_{\text{opt}}$). Use the previous parts to show that there must exist a g_j that is at most distance $2 \cdot R_{\text{opt}}$ away from it.

(v) Conclude that the greedy algorithm indeed achieves a response time of $R_g \leq 2 \cdot R_{\text{opt}}$.

3 Second Smallest Global Min-cut (17 points)

Recall that a *cut* of an undirected, unweighted graph (V, E) is a partition of the vertices into two non-empty sets S and $V \backslash S$. The *weight* of such a cut is the number of edges crossing the cut, which we denote $|E(S, V \backslash S)|$. Karger's contraction algorithm from class finds a cut of minimum weight with probability $1 - p$, with runtime $O(poly(n) \cdot \log(1/p))$, where $n := |V|$ and $m := |E|$.

In this problem, we will be concerned with finding not the smallest, but rather the second smallest minimum cut. Ties are broken arbitrarily. For example, consider the graph below:



It has two cuts of weight one, given by $S = \{1\}$ or $S = \{4\}$. All other cuts weigh at least two. Since we break ties arbitrarily, we would say in this example that both the smallest *and* second smallest cuts have weight equal to 1.

(a) (2 points) Give a general formula for the number of cuts in a graph with n vertices and m edges. **Note:** We are asking for the number of cuts, not necessarily *minimum* cuts.

(b) (3 points) Focus on a particular second smallest mincut S^* of G. Consider running just the first step of Karger's contraction algorithm, where we contract a uniformly random edge. Compute a tight upper bound α for the probability that the contracted edge crosses $(S^*, V \backslash S^*)$. Your answer is allowed to depend on $k \coloneqq |S^*|$ or n, but not on m.

(c) (1 point) Starting with n vertices, imagine doing $n - 3$ random contraction steps in sequence. How many "supervertices" are left?

(d) (5 points) Suppose your answer in part (c) is c. Imagine then picking a uniformly random cut (not necessarily a mincut) on this graph of c supervertices to get a resulting cut of the original graph. Show that the probability that this resulting cut is S^* is $\Omega(1/n^2)$.

(e) (6 points) Give a $O(\text{poly}(n) \cdot \log(1/p))$ time algorithm for finding the second smallest mincut in a graph with success probability at least $1 - p$. You need not calculate the precise exponent of n in the poly(n) term; any polynomial dependence suffices.

Hint: $(1 + x)^T \leq e^{xT}$ *for all* $x \in \mathbb{R}$ *and* $T \geq 0$.

4 One-Sided Error and Las Vegas Algorithms (10 points)

An RP algorithm is a randomized algorithm that runs in polynomial time and always gives the correct answer when the correct answer is 'NO', but only gives the correct answer with probability greater than 1/2 when the correct answer is 'YES'.

(a) (5 points) Prove that every problem that admits an RP algorithm is in NP (i.e., show that $\mathsf{RP} \subseteq \mathsf{NP}$).

Hint: It may be helpful to view a randomized algorithm $R(x)$ as a deterministic algorithm $A(x, r)$ where x is the input and r is the result of the 'coin flips' which the algorithm uses for its randomness.

(b) (5 points) A *Las Vegas Algorithm* is a randomized algorithm that always gives the right solution, but whose runtime is random. A ZPP algorithm is a Las Vegas algorithm that runs in expected polynomial time (ZPP stands for Zero-Error Probabilistic Polytime). Prove that if a problem has a ZPP algorithm, then it has an RP algorithm.

Hint: Since we have not told you anything about the structure of the ZPP algorithm, your RP algorithm can only use it by running it somehow. What can you do to make sure the runtime is bounded? Use Markov's inequality.

5 Better-Than-Most TSP Tour (Optional)

An instance of TSP consists of n cities and distances $d[\cdot, \cdot]$ between every pair of cities. The distances may not satisfy the triangle inequality. A TSP tour is a walk that visits every city exactly once and returns to the first visited city.

There are $(n-1)!$ possible TSP tours in any instance with n cities. Finding the TSP tour that has the smallest total length among all these $(n-1)!$ tours is NP-Hard. For distances that don't satisfy the triangle inequality, there are no approximation algorithms for the problem either.

Let us try to approximate TSP from another perspective. We define a TSP tour to be *Better-Than-Most* if its cost is smaller than 99% of the $(n-1)!$ possible TSP tours.

Given a constant $\delta \in (0, 1)$, describe an algorithm that outputs a tour that is *Better-Than-Most* with probability $1 - \delta$. Your algorithm should run in polynomial time with respect to n and $1/\delta$. Please provide a 3-part solution.

Hint: as a first step, try to figure out how to uniformly sample a tour at random. Then, consider uniformly sampling some number of tours as part of your algorithm.

6 $\sqrt{n}$ coloring (Optional)

(a) Let G be a graph of maximum degree δ. Show that G is $(\delta + 1)$-colorable.

(b) Suppose $G = (V, E)$ is a 3-colorable graph. Let v be any vertex in G. Show that the graph induced on the neighborhood of v is 2-colorable.

Note: the graph induced on the neighborhood of v refers to the following subgraph:

$$G' = (V' = \text{neighbors of } v, E' = \text{all edges in } E \text{ with both endpoints in } V').$$

(c) Give a polynomial time algorithm that takes in a 3-colorable n-vertex graph G as input and outputs a valid coloring of its vertices using $O(\sqrt{n})$ colors. Prove that your algorithm is correct.

Hint: think of an algorithm that first assigns colors to "high-degree" vertices and their neighborhoods, and then assigns colors to the rest of the graph. The previous two parts might be useful.

