# CS 170 Homework 6

Due **Friday 10/11/2024, at 10:00 pm (grace period until 11:59pm)**

## 1 Study Group

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

## 2 Firefighters

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 $a, b, c$.

We want to pick $K$ distinct cities and build fire stations there. For each city without a fire station, the response time for that city is given by 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 devise a greedy algorithm to solve this problem. However, a greedy approach will not always give the optimal solution. Instead, we'll just aim to get a solution that is "good enough".

We can formalize the problem as follows: Suppose the optimal assignment of fire stations to cities produces response time $R_{\text{opt}}$. Given positive integers $N, K$ and the 2D matrix $D$ as input, describe an $O(N^2 \cdot K)$ (or faster) greedy algorithm to output an assignment that achieves a response time of $R_g \leq 2 \cdot R_{\text{opt}}$.

**Provide a 3-part solution.** For your proof of correctness, show that your algorithm achieves the desired approximation factor of 2.

# 3 Updating a MST

You are given a graph $G = (V, E)$ with positive edge weights, and a minimum spanning tree $T = (V, E')$ with respect to these weights; you may assume $G$ and $T$ are given as adjacency lists. Now suppose the weight of a particular edge $e \in E$ is modified from $w(e)$ to a new value $\hat{w}(e)$. You wish to quickly update the minimum spanning tree $T$ to reflect this change, without recomputing the entire tree from scratch.

There are four cases. In each, give a description of an algorithm for updating $T$, a proof of correctness, and a runtime analysis for the algorithm. Note that for some of the cases these may be quite brief. For simplicity, you may assume that no two edges have the same weight (this applies to both $w$ and $\hat{w}$).

(a) $e \in E'$ and $\hat{w}(e) < w(e)$

(b) $e \notin E'$ and $\hat{w}(e) < w(e)$

(c) $e \in E'$ and $\hat{w}(e) > w(e)$

(d) $e \notin E'$ and $\hat{w}(e) > w(e)$

 

# 4 Minimum Spanning $k$-Forest

Given a graph $G(V, E)$ with nonnegative weights, a spanning $k$-forest is a cycle-free collection of edges $F \subseteq E$ such that the graph with the same vertices as $G$ but only the edges in $F$ has $k$ connected components. For example, consider the graph $G(V, E)$ with vertices $V = \{A, B, C, D, E\}$ and all possible edges. One spanning 2-forest of this graph is $F = \{(A, C), (B, D), (D, E)\}$, because the graph with vertices $V$ and edges $F$ has components $\{A, C\}, \{B, D, E\}$.

The minimum spanning $k$-forest is defined as the spanning $k$-forest with the minimum total edge weight. (Note that when $k = 1$, this is equivalent to the minimum spanning tree). In this problem, you will design an algorithm to find the minimum spanning $k$-forest. For simplicity, you may assume that all edges in $G$ have distinct weights.

(a) Define a $j$-partition of a graph $G$ to be a partition of the vertices $V$ into $j$ (non-empty) sets. That is, a $j$-partition is a list of $j$ sets of vertices $\Pi = \{S_1, S_2 \dots S_j\}$ such that every $S_i$ includes at least one vertex, and every vertex in $G$ appears in exactly one $S_i$. For example, if the vertices of the graph are $\{A, B, C, D, E\}$, one 3-partition is to split the vertices into the sets $\Pi = \{\{A, B\}, \{C\}, \{D, E\}\}$.

Define an edge $(u, v)$ to be crossing a $j$-partition $\Pi = \{S_1, S_2 \dots S_j\}$ if the set in $\Pi$ containing $u$ and the set in $\Pi$ containing $v$ are different sets. For example, for the 3-partition $\Pi = \{\{A, B\}, \{C\}, \{D, E\}\}$, an edge from $A$ to $C$ would cross $\Pi$.

Show that for any $j$-partition $\Pi$ of a graph $G$, if $j > k$ then the lightest edge crossing $\Pi$ must be in the minimum spanning $k$-forest of $G$.

(b) Give an efficient algorithm for finding the minimum spanning $k$-forest.

**Please give a 3-part solution.**

 

## 5 Copper Pipes

Bubbles has a copper pipe of length n inches and an array of nonnegative integers that contains prices of all pieces of size at most $n$. He wants to find the maximum value he can make by cutting up the pipe and selling the pieces. For example, if length of the pipe is 8 and the values of different pieces are given as following, then the maximum obtainable value is 22 (by cutting in two pieces of lengths 2 and 6).

| length | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| price | 1 | 5 | 8 | 9 | 10 | 17 | 17 | 20 |

Give a dynamic programming algorithm so Bubbles can find the maximum obtainable value given any pipe length and set of prices. Clearly describe your algorithm and analyze its runtime (proof of correctness not required).

# 6 [Coding] Simple Dynamic Programming

For this week's homework, you'll implement implement some simple dynamic programming algorithms. There are two ways that you can access the notebook and complete the problems:

1. **On Datahub**: click here and navigate to the `hw06` folder.
2. **On Local Machine**: `git clone` (or if you already cloned it, `git pull`) from the coding homework repo,

   `https://github.com/Berkeley-CS170/cs170-fa24-coding`

   and navigate to the `hw06` folder. Refer to the `README.md` for local setup instructions.

Notes:

- *Submission Instructions:* Please download your completed submission `.zip` file and submit it to the Gradescope assignment titled "Homework 6 Coding Portion".
- *Getting Help:* Conceptual questions are always welcome on Edstem and office hours; *note that support for debugging help during OH will be limited.* If you need debugging help first try asking on the public Edstem threads. To ensure others can help you, make sure to:

  1. Describe the steps you've taken to debug the issue prior to posting on Ed.
  2. Describe the specific error you're running into.
  3. Include a few small but nontrivial test cases, alongside both the output you expected to receive and your function's actual output.

  If staff tells you to make a private Ed post, make sure to include *all of the above items* plus your full function implementation. If you don't provide them, we will ask you to provide them.
- *Academic Honesty Guideline:* We realize that code for some of the algorithms we ask you to implement may be readily available online, but we strongly encourage you to not directly copy code from these sources. Instead, try to refer to the resources mentioned in the notebook and come up with code yourself. That being said, we **do acknowledge** that there may not be many different ways to code up particular algorithms and that your solution may be similar to other solutions available online.

 