CS 170 Homework 4

Due **2021-09-27, 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 Running Errands

You need to run a set of k errands in Berkeley. Berkeley is represented as a directed weighted graph G, where each vertex v is a location in Berkeley, and there is an edge (u, v) with weight w_{uv} if it takes w_{uv} minutes to go from u to v. The errands must be completed in order, we'll assume the ith errand can be completed immediately upon visiting any vertex in the set S_i (for example, if you need to buy snacks, you could do it at any grocery store). Your home in Berkeley is the vertex h.

Given G, h, and all S_i as input, given an efficient algorithm that computes the time needed to complete all the errands starting at h. That is, find the shortest path in G that starts at h and passes through a vertex in S_1, then a vertex in S_2, then in S_3, etc.

Give a 3-part solution.

Solution: Main idea Create $k+1$ copies of G, called $G_0, G_1, ...G_k$, to form G'. Let the copy of v in G_i be v_i. For every v in S_i, we add an edge from v_{i-1} to v_i with weight 0. We run Dijkstra's starting from h_0 in G', and output the shortest path length to any vertex in G_k.

Correctness Any path in G' from h_0 to a vertex G_k can be mapped to a path in G of the same length passing through vertices, by taking each edge (u_i, v_i) and replacing it with the edge (u, v) in G, ignoring edges of the form (v_{i-1}, v_i). These paths must also complete the errands in order, since they must contain edges of the forms $(v_0, v_1), (v_1, v_2), \ldots$ in that order.

Runtime analysis This takes time $O(k(|V| + |E|) \log k|V|)$ since the new graph is k times the size of the original graph.

3 MST Variant

Give an undirected graph $G = (V, E \cup S)$ with edge weight $c(e)$. Note that S is disjoint with E. Design an algorithm to find a minimum one among all spanning trees having at most one edge from S and others from E.

Input: A graph $G = (V, E \cup S)$, and a cost function $c(e)$ defined for every $e \in E \cup S$.

Output: A tree $T = (V, E')$ such that T is connected (there is a path in T between any two vertices in V), $E' \subseteq E \cup S$, $\sum_{e \in E'} c(e)$ is minimized, and $|E' \cap S| \leq 1$.

Give a 3-part solution.

Solution:

Main Idea: Use Prim's (or Kruskal's) algorithm to find the MST T' of G. Then for each potential $s \in S$, try adding s to T' and keep track of the difference between $c(s)$ and $c(e)$ where e is the edge of highest weight along that cycle. If $c(s) < c(e)$, then adding s to T' and removing e from T' creates a new spanning tree with lower total weight than T'. Doing this for each $s \in S$ and taking the smallest resulting spanning tree gives us T.

Pseudocode:

procedure MSTVARIANT(graph G, additional edges S, cost function c)
  Use Prim's to find $T' = (V, E')$, the MST of G
  bestDifference $\leftarrow 0$
  bestEdges $\leftarrow E'$
  for $s \in S$ **do**
    Add s to T' and use DFS to find the cycle containing s. Let e be the most costly edge in that cycle.
    if $c(e) - c(s) >$ bestDifference **then**
      bestDifference $\leftarrow c(e) - c(s)$
      bestEdges $\leftarrow E' \cup \{s\} - \{e\}$
  return (V, bestEdges)

Running time analysis: Prim's takes $O(|E| \log |E|)$ to find the MST. For each edge in S, we run a DFS on T', which has $|V|$ vertices and $O(|V|)$ edges. Therefore, this takes $O(|S||V|)$ time. Thus, the overall running time is $O(|E| \log |E| + |S||V|)$.

Proof of correctness: The optimal solution containing exactly one edge from S cannot be better than the optimal solution containing exactly one edge from S and exactly $|V| - 2$ edges from T', the MST of G. To see this, assume towards contradiction that the optimal solution T contains some $e \in E$ such that e is not part of T'. Consider the cut in T defined by e, i.e. if $e = (u, v)$, then divide the vertices into those reachable from u and those reachable from v without using e. Let e' be the edge of lowest weight in T' that spans that cut. Adding e' to T creates a cycle with e. We know that $c(e') \leq c(e)$ because otherwise e would be in T' instead of e'. Therefore, removing e from T and adding e' yields a spanning tree that costs no more than T. Since we can do this for any edge in T but not in T', we can effectively transform T into a solution that is at least as good and contains no edges from E that are not in T'.

4 Blackout in the Igloos

It's the year 3077, exactly 1000 years after Cyberpunk happened. PNPenguins live in igloos that are equipped with one-directional portals, allowing them to instantly travel from one igloo to another. It is polar night now, so lights are necessary for PNPenguins to navigate inside their igloos – without light, igloos are completely dark inside. There are n igloos in total, and in each igloo, there are k one-directional portals numbered $\{1 \dots k\}$. Each portal can teleport PNPenguins to a certain igloo (including the one that this portal is located).

Recently, PG&E (Penguin Gaming and Electric Company - yes, gaming is considered a utility in 3077) issued a notice indicating power could be shut off at any time. PNPenguins

want to buy an emergency power generator for one igloo, and in the case of a power shut off, all PNPenguins need to rendezvous there. PNPenguins are terrible at memorizing which igloo they are in, but they all have a very good sense of direction. When the power is shut off at an igloo, they still can choose any portal and travel through it to another igloo.

In the event of a power shut off, PNPenguins need a way to know how to get to the igloo with the power generator. Thus, there has to be a fixed sequence of numbers $p_1, ..., p_l$ such that starting from any igloo, if a PNPenguin goes into portals p_1, ..., p_l, it ends up at the igloo with the power generator.

PNPenguins are wondering, if there exists any fixed sequence of portal numbers $p_1, ..., p_l$, such that regardless of the igloo they are currently in, after going through these portals, they end up in the same igloo? They only need to know if such a sequence of portals exist without knowing the sequence itself.

More formally, there are k functions $f_1, ..., f_k$ and each function maps each igloo to an igloo (possible the same). Your goal is to verify if there exists a sequence of numbers $p_1, ..., p_l$ such that $f_{p_1} \circ f_{p_2} \circ \cdots \circ f_{p_{l-1}} \circ f_{p_l}$ outputs the same igloo regardless of the igloo a PNPenguin starts from, i.e.,

$$\forall x, y \in \{1 \dots n\} f_{p_1} \circ f_{p_2} \circ \cdots \circ f_{p_{l-1}} \circ f_{p_l}(x) = f_{p_1} \circ f_{p_2} \circ \cdots \circ f_{p_{l-1}} \circ f_{p_l}(y)$$

$\circ$ is function composition:

$$f_{p_1} \circ f_{p_2} \circ \cdots \circ f_{p_{l-1}} \circ f_{p_l}(x) = f_{p_1}(f_{p_2}(\dots(f_{p_{l-1}}(f_{p_l}(x)))))$$

(a) Provide a 3-part solution for the described problem.

(b) Go to the online contest system at https://hellfire.ocf.berkeley.edu/, access the HW4 contest and provide a program to solve this problem.

Hint First, think about just two igloos. Can you find an algorithm that will find if there is a sequence of doors that will map these two igloos to the same igloo? Then, try to generalize this to all igloos by "merging" together pairs of igloos. Based on this, try to find an "if and only if" condition for the existence of a sequence of doors that will map all igloos to the same igloo. Can you develop an algorithm to verify if this "if and only if" condition holds?

Note that there are several ways to solve this problem with different time complexities, ranging from a very slow to a very fast solution. This is a hard problem and we do not expect everyone to get to the best possible solution. Because of this, we will be awarding partial credit for slower solution, even for a solution that is exponential in the input size. Try to do your best. We will be providing Office Hours support for this question, but note that we will emphasize conceptual help and we do not guarantee that a staff member will be able to debug your code.

Solution: To be written.