HW#10 due on Sunday May 7 at 11:59pm **ECE 374, Spring 2023**

Instructions

- please check HW1-3 for detailed instructions (they remain the same)

28 (100 PTS.) Skip distance.

Let $\mathsf{G} = (\llbracket n \rrbracket, \mathsf{E})$ be a connected undirected graph with positive distinct costs on its edges (the costs are real numbers), with n vertices and m edges, where $\llbracket n \rrbracket = \{1, 2, \ldots, n\}$. You can assume here that $n \leq m$. Here, the cost of an edge $e \in \mathsf{E}$ is denoted by $c(e)$.

28.A. (30 PTS.) Given a set $X \subseteq \mathsf{E}$, Let $C_1, \ldots, C_k$ be the connected components of the graph $\mathsf{J} = (\llbracket n \rrbracket, X)$. Consider the reduced graph $\mathsf{G}/X = (\llbracket k \rrbracket, \mathsf{E}')$, where there is an edge $i'j' \in \mathsf{E}'$, if and only if there exists an edge $ij \in \mathsf{E}$, such that $i \in \mathsf{V}(C_{i'})$ and $j \in \mathsf{V}(C_{j'})$. If $i'j' \in \mathsf{E}'$, then

$$c(i'j') = \min_{ij \in \mathsf{E}: i \in \mathsf{V}(C_{i'}), j \in \mathsf{V}(C_{j'})} c(ij).$$

Describe in detail an algorithm, as fast as possible, that computes the reduced graph G/X. You might need to use radix sort here[1]. A solution using hashing is worth no points at all.

28.B. (20 PTS.) The ***skip*** price of a path π in G is $s(\pi) = \max_{e \in \pi} c(e)$. The ***skip distance*** for two vertices $i, j \in \mathsf{V}(\mathsf{G})$, is

$$s(i, j) = \min_{\pi: \text{path connecting } i \text{ to } j \text{ in } \mathsf{G}} s(\pi).$$

Given two vertices $u, v \in \llbracket n \rrbracket$, and a real number α, describe an algorithm, as fast as possible, that decides if $s(u, v) > \alpha$.

28.C. (50 PTS.) For a number β, let

$$\mathsf{E}_{\leq \beta} = \{e \in \mathsf{E} \mid c(e) \leq \beta\},$$

be the set of edges in G with cost at most β.

A natural algorithm for computing the skip distance between u and v is to pick some value α, and decide if $s(u, v) > \alpha$, and if so, the algorithm recurses on computing the skip distance between u' and v' in $\mathsf{G}' = \mathsf{G}/\mathsf{E}_{\leq \alpha}$, where u' and v' are the two meta vertices of G' that their original connected components contains u and v, respectively. Otherwise, if $s(u, v) \leq \alpha$, the algorithm computes the skip distance between u and v in the graph $(\llbracket n \rrbracket, \mathsf{E}_{\leq \alpha})$. If the graph has constant size, the algorithm computes the skip distance directly using brute force.

First prove that this algorithm is correct. Next, describe how to implement this algorithm efficiently, so that the resulting algorithm is as fast as possible (in particular, what is the right value of α to pick, and how do you compute it?). What is the running time of your algorithm as a function of n and m?

29 (100 PTS.) More on MST.

[1] Recall that radix sort allows you to sort n integer numbers that are in the range $1, \ldots, n^{O(1)}$ in $O(n)$ time (if you do not know what radix sort is, read the wikipedia page).

29.A. (40 PTS.) THE SPANNING GRAPH THAT SURVIVES.

Let $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ be a connected graph on n vertices and m edges, with unique costs on the edges. A subgraph H of G is a ***MST survivor***, if for any edge $e \in \mathsf{E}$, we have that $\mathsf{G} - e$ and $\mathsf{H} - e$ have the same minimum spanning forest, where $\mathsf{G} - e$ denotes the graph G after we delete the edge e from it. Present an algorithm, as fast as possible, that computes a MST survivor of G that has a minimum number of edges among all such graphs. How fast is your algorithm? Prove that your algorithm is correct.

29.B. (40 PTS.) LINEAR TIME MST.

Some Nigerian prince, that needs your help, sent you as a gift a black box B and a graph G. The graph G has n vertices and m edges, and real positive distinct weights on its edges. The black box can computes the MSF (minimum spanning forest) of any graph with n' vertices, and at most $(3/2)n'$ edges, in $O(n')$ time (it will compute the minimum spanning forest of this graph if it is not connected). You can use the box B only on a graph with n' vertices, such that $n' \leq n$.

Present an algorithm that computes the MST of G, in $O(m)$ time, using this black box (you can safely assume that $m \geq n$). Prove the correctness of your algorithm.

29.C. (20 PTS.) Assume you are given a union-find data-structure that can perform an operation in $O(1)$ time. Given a graph G with n vertices and m edges, where the weights on the edges are positive integers that are all $O(n^5)$, describe an algorithm, as fast possible, for computing the MST of G. What is the running time of your algorithm?

30 (100 PTS.) Not on MST.

You are given a directed graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ with $2n$ vertices, and m edges. Here, the vertices appear in pairs (i.e., twins) x_i, y_i, for $i = 1, \ldots, n$. For a vertex u in this graph, let $\mathbf{twin}(u)$ be the other vertex in the pair of u (thus, if $u = x_i$, then $\mathbf{twin}(u) = y_i$).

There are no edges between twins in this graph. If an edge $u \to v$ appears in the graph, so does the edge $\mathbf{twin}(v) \to \mathbf{twin}(u)$. A subset of vertices $S \subseteq \mathsf{V}(\mathsf{G})$ is ***closed*** if $x \in S$, then all the vertices reachable from x in G are in S. A closed set is ***bad*** if it includes two vertices that are twins. A set of vertices $S \subseteq \mathsf{V}(\mathsf{G})$ is ***perfect*** if it is closed, $|S| = n$, and it includes exactly one vertex from each pair (i.e., it is not bad).

Describe a greedy algorithm, as fast as possible, that decides if there is a perfect set in the graph, and if so computes it. What is the running time of your algorithm? Prove the correctness of your algorithm.

(Hint: Compute the strong connected components of G, and analyze the meta graph of strong connected components of this graph. Add vertices to the output set using the meta graph, in a greedy fashion.)