MPCS 55001 Algorithms Autumn 2023
Homework 7

Test Student
Test ID - 2022
Collaborators: Student A, Student B

Instructions: Write up your answers in LaTeX and submit them to Gradescope. We will not grade handwritten homework.

Collaboration policy: You may work together with other students on homework. If you do, state it at the beginning of your solution: **give the name(s) of collaborator(s)** and the nature and extent of the collaboration. Giving/receiving a hint counts as collaboration. Note: you must write up solutions **independently without assistance**.

Internet sources: You must include the url of any internet source in your homework submission.

1 Dynamic MST (13 points)

Suppose that we have a weighted graph $G = (V, E, w)$ with **distinct** edge weights and an MST T already computed. We want to update a minimum spanning tree for G as vertices and edges are added or deleted.

For each part below, write **pseudocode** for an algorithm which takes $G = (V, E, w)$ and its MST T, the modified graph $G' = (V', E', w')$, and the vertices/edges being modified as inputs, and outputs an MST T' for G'.

(a) (3 points) G' has the same vertices as G and one new edge e is added. Your algorithm must run in $O(V)$ time. **Prove your solution is correct**, and analyze its running time.

(b) (4 points) G' has the same vertices as G and one edge e is removed. You may assume that G' is still connected. Your algorithm must run in $O(E)$ time. **Prove your solution is correct**, and analyze its running time.

(c) (4 points) G' has one new vertex v and k new edges $e_1, e_2, \ldots, e_k$ are added connecting v to vertices in G, where $1 \leq k \leq |V|$. The running time of your algorithm should not depend on $|E|$, the number of edges in G. **No credit for $\Omega(E \log V)$ time algorithms.**

(d) (2 points) Show that T may provide no information about T' when we remove a vertex and the incident edges. Specifically, give an example of $G = (V, E, w)$ and $G' = G - v$ such that the MST of G has no edges in common with the MST of G'.

2 Borůvka's Algorithm (12 points)

Borůvka's MST algorithm (1926) can be described as a distributed version of Kruskal's algorithm. The input is an undirected connected weighted graph $G = (V, E)$ with edge weight function w. We assume that distinct edges have distinct weights. (Alternatively, the algorithm can be modified to work with arbitrary edge weight functions by adding a tie-breaking rule.)

The algorithm proceeds by phases. In the first phase, we begin by having each vertex mark the edge incident to it whose weight is minimum. (For example, if the graph were a 4-cycle with edges of weights 1, 3, 2, and 4 around the cycle, then two vertices would mark the "1" edge and the other two vertices would mark the "2" edge.) This creates a forest F of marked edges. In the next phase, each tree F_i in F marks the minimum weight edge incident to it, i.e., the minimum weight edge having one endpoint in the tree F_i and one endpoint not in F_i, creating a new forest F'. The algorithm repeats these phases until there is only one tree.

(a) (2 points) Given the forest $F \subseteq E$, write **pseudocode** to compute the connected components F_i in $O(V+E)$ time.

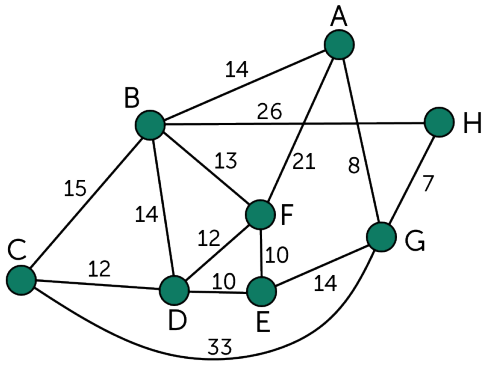
(b) (3 points) Write **pseudocode** for Borůvka's algorithm.

(c) (3 points) **Prove** that each phase can be implemented in $O(V+E)$ time. **Prove** that Borůvka's algorithm has at most $O(\lg V)$ phases. This gives a running time of $O((V+E)\lg V)$ for the entire algorithm.

(d) (4 points) **Prove** the correctness of the algorithm by arguing that $E(F)$ is always contained in the MST. Conclude that the constructed graph F never has any cycles. [Hint: use induction]

3 Lightest Branch Spanning Tree (17 points)

Let $G = (V, E, w)$ be an undirected weighted graph. The **heaviest branch** of a spanning tree T is an edge which has the highest weight in T. Moreover, we refer to this weight as the **heaviest branch value**. A **lightest branch spanning tree** (LBST) is a spanning tree that has the least possible **heaviest branch value**.



For example, in the graph above, the tree given by $T_1 = \{(A,B),(B,F),(C,D),(D,E),(E,F),(E,G),(G,H)\}$ is an LBST with the heaviest branch being (A,B) and value 14.

The tree given by $T_2 = \{(A,B),(B,C),(C,D),(D,E),(E,F),(E,G),(G,H)\}$ has (B,C) as the heaviest branch with weight 15; this is **not** an LBST.

(a) (3 points) Prove that any minimum spanning tree (MST) is also a lightest branch spanning tree (LBST).

(b) (2 points) Prove or disprove: Every LBST is also an MST.

(c) (2 points) Write **pseudocode** for function `LBST_below` which takes as input $G = (V, E, w)$ and a number b and outputs true if G has an LBST with heaviest branch value at most b. Else, it outputs false. Justify why your function works and compute its running time.

(d) (2 points) Use the function you wrote in part (c) to write **pseudocode** for an $O\big((V+E)\log V\big)$ time algorithm to compute a LBST. *You may not call Prim or Kruskal.*

(e) (1 point) Prove that if graph $G_b = (V, E_b)$ where $E_b = \{e : (e \in E) \cap (w(e) \leq b)\}$ is connected then the heaviest branch value of an LBST of G is at most b.

(f) (1 point) Prove that if graph $G_b = (V, E_b)$ is disconnected, you can greedily produce a partial LBST of G missing only the edges between the connected components of G_b.

(g) (1 point) Earlier you studied the Quicksort algorithm, a randomized algorithm for sorting an array of numbers. Describe (2-3 sentences is fine) how you would modify Quicksort to instead select the median value in an array of numbers in linear expected time.

(h) (5 points) Combine the results of (e) and (f) with a choice of b to reduce the edges you need to consider by half in each iteration and reduce the complexity of your algorithm to $O(V+E)$. Write pseudocode for your algorithm. You may assume that you already have a function to find the median of an unsorted array of numbers in deterministic linear time.

Hint: Consider the relationship between the number of nonisolated vertices (i.e., vertices with adjacent vertices) and the number of edges in a graph. How might you be able to leverage this relationship in analyzing the running time of your algorithm?

4 Programming: Image Segmentation (20 points)

Follow this GitHub Classroom link to accept the assignment. Your code should be pushed to GitHub; you do not need to include it here.