Exercise 6: Can't see the Tree for the Forests!

Task 1: Forest Logging

This exercise leads to a very quick MST algorithm for specialized graphs. Note that this is what always happens: Looking at a specific type of graphs makes things easier.

We are given a complete weighted graph $G = (V, \binom{V}{2}, W)$. The goal is to find an extremely fast variant of the GHS algorithm in this setting. W.l.o.g., we assume that the system is synchronous. Message size is $\mathcal{O}(\log n)$, where we assume that an edge weight fits into a message. W.l.o.g., all edge weights are distinct and node identifiers are $1, \dots, n$.

a) Suppose that at the beginning of phase i, the set T_i of previously selected MST edges is known to all nodes. Denote by $\mathfrak{C}$ the set of (node sets of) connectivity components of (V, T_i). For $C, C' \in \mathfrak{C}$, denote by $e(C, C')$ the lightest edge between C and C', and define $\delta := \min\{\min_{C\in\mathfrak{C}}\{|C|\}, |\mathfrak{C}| - 1\}$. Find a subroutine that permits to determine for each $C \in \mathfrak{C}$ the lightest δ edges $e(C, \cdot)$ in $\mathcal{O}(1)$ rounds! At the end, the nodes in C must know these edges.

   Hint: For component C, use not only the edges within the component, but also those between C and all other nodes. This way you still don't overuse any edges! Exploit that $|\mathfrak{C}| \leq n$ and that only the lightest edge $e(C, C')$ between any pair of components C, C' is of interest.

b) Use a) to devise an MST algorithm that requires $\mathcal{O}(1)$ rounds per phase, maintains that at the beginning of each phase, all nodes know all previously determined MST edges, and in each phase, each component C is merged with at least δ other components!

c) Prove that the algorithm from b) computes the MST in $\mathcal{O}(\log \log n)$ rounds!

   Hint: Bound the minimal size of a component after phase i from below!

Task 2: Steiner Trees

The *Steiner tree* problem is a generalization of the MST problem. The goal is to find a minimum cost tree connecting a set of *terminals* $T \subseteq V$. This problem is NP-hard to approximate within factor 96/95; we'll settle for a 2-approximation.

Definition 1 (Terminal graph). *Given a connected simple weighted graph $G = (V, E, W_G)$ and terminals $T \subseteq V$, the* terminal graph *is $(T, \binom{T}{2}, W_T)$, where $W_T(s, t)$ is the minimum weight of a path from s to t in G. (For a path $p = (v_0, \dots, v_k)$, its weight is $\sum_{i=1}^{k} W_G(v_{i-1}, v_i)$.)*

Consider an optimal Steiner tree O.

a) Show that when visiting all nodes of O and returning to the root in the order given by a depth-first-search using the tree edges, each edge of O is traversed exactly twice. This is also called an *Euler tour* of O.

b) List the nodes of T according to the order in which they are visited in the Euler tour of O. Denote this list by $(t_1, \dots, t_{|T|})$. Show that

$$\sum_{i=1}^{|T|-1} W_T(t_i, t_{i+1}) \leq 2W_G(O).$$

   (The weight of an edge set is the sum of the edge weights.)

c) Conclude that the MST *edges* of the terminal graph induce a spanning subgraph[1] whose weight is at most twice the weight of O!

d) Assuming an MST of T is already constructed and a corresponding edge set in G is already known (in the sense that nodes know their incident edges of the induced spanning subgraph), can you construct a 2-approximate solution that is a tree fast? How fast is your post-processing routine?

This is only one fragment of a distributed algorithm that constructs a 2-approximate Steiner tree. But fear not! We will devise an efficient such algorithm in this manner in a future exercise.

Task 3*: The Traveling Salesman

In the *Traveling Salesman Problem (TSP),* a set of cities V is connected by charter flights $E \subseteq \binom{V}{2}$ of cost $W(e)$. The goal is to visit each city exactly once, and ending at the starting city. In other words, the goal is to connect them by a Hamiltonian cycle, while minimizing the total cost of chartered flights.

In this exercise, we consider a complete graph $G = (V, \binom{V}{2}, W)$. Furthermore, we say that G is a *metric* if

$$\forall v, w \in V: \quad \operatorname{dist}(v, w) = W(v, w),$$

where $\operatorname{dist}(\cdot, \cdot)$ is the weighted distance.

a) Consider the following algorithm:
   (1) Determine an MST on G.
   (2) Imagine a depth-first search on the MST.
   (3) Traverse V in prefix-order of this walk.
   Show that this yields a 2-approximation of TSP if G is a metric.

b) Show that the algorithm from a) does not yield a 2-approximation if G is not a metric.

c) Research how the size of the largest TSP instances that could be solved/approximated at the time changed over the years. How much of this is to be attributed to hardware improvements and how much to algorithmic improvements?

d) Tell about your journey in the exercise session!

[1] In fact, it might have arbitrarily weird cycles in it!