Advanced topics in TCS

Exercise sheet 4.
Minimum spanning tree, Testing k-connectivity

Christian Konrad

Question 1. Minimum Spanning Tree (MST)

We consider a weighted graph $G = (V, E, w)$, where $w : E \to \mathbb{N}$ is an edge weight function. A *minimum spanning tree* $F \subseteq E$ in G is a spanning tree in G of minimum weight, i.e., the sum of its edge weights is as small as possible.

We consider the streaming edge-arrival model where the edges arrive together with their weights. More specifically, the input stream consists of a sequence of tuples $(e_i, w(e_i))_i$, where $w(e_i)$ is the weight of edge e_i.

1. Give a 1-pass semi-streaming algorithm for computing an MST.

   Hint: Adapt the spanning tree algorithm from the lecture.

2. Let E_i be the first i edges in the stream, $G_i = (V, E_i, w|_{E_i})$ (where $w|_{E_i}$ denotes the weight function w restricted to the domain E_i), and let F_i denote the collection of edges stored by the algorithm given in the previous exercise after iteration i. Prove by induction that F_i is a MST in G_i.

   The following property may be useful:

   Lemma 1. *Let $T \subseteq E$ be a spanning tree in a weighted graph $G = (V, E, w)$. Then, if T is not a minimum spanning tree, then there exists an edge $e \in E \setminus T$ such that $w(e) < w(f)$, for at least one edge f different to e in the unique cycle in $T \cup \{e\}$.*

Question 2. Deciding k-Connectivity

We say that a graph G is *k-connected* if we need to remove at least k edges from G in order to disconnect G.

Consider the following algorithm for deciding k-connectivity of a graph:

1. $F_1, F_2, \ldots, F_k \leftarrow \varnothing$
2. For each edge e in the stream: If there is an $i \in \{1, \ldots, k\}$ such that $F_i \cup \{e\}$ has no cycle then add e to F_i (if there are multiple such i then pick only one, ties can be broken arbitrarily)
3. Post-processing: Let $F = \bigcup_{i=1}^{k} F_i$

   If (V, F) is k-connected then **return** "G is k-connected", otherwise **return** "G is not k-connected"

Algorithm 1.

1. How much space does Algorithm 1 use (as a function of n and k)?
2. Prove that the algorithm is correct.