Advanced topics in TCS

Exercise sheet 4: **Solution**
Minimum spanning tree, Testing k-connectivity

Christian Konrad

Question 1. Minimum Spanning Tree (MST)

We consider a weighted graph $G = (V, E, w)$, where $w : E \rightarrow \mathbb{N}$ 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. *Solution:*

   $F \leftarrow \emptyset$

   While stream not empty:

   (a) Let e be the next edge in the stream

   (b) **if** $(F \cup \{e\})$ does not contain a cycle **then** $F \leftarrow F \cup \{e\}$

   (c) **else** ($(F \cup \{e\})$ does contain a cycle)

      i. Let C be the edge set of the (unique) cycle in $F \cup \{e\}$

      ii. Let f be an edge of maximum weight in $C \setminus \{e\}$

      iii. **if** $w(f) > w(e)$ **then** $F \leftarrow (F \setminus \{f\}) \cup \{e\}$

   return F

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 $T \cup \{e\}$.*

Hint: Adapt the spanning tree algorithm from the lecture.

Solution:

Proof.
Base case. $F_0 = \emptyset$ and $E_0 = \emptyset$. Observe that F_0 is a MST of an empty graph.
Induction step. Let F_i be a MST in graph G_i. We will only consider the interesting case when $F_{i+1} = (F_i \setminus \{f_{i+1}\}) \cup \{e_{i+1}\}$, where f_{i+1} is the edge of the cycle C_{i+1} that was removed when inserting e_{i+1}. Observe that this implies that $w(e_{i+1}) < w(f_{i+1})$.
Assume for the sake of a contradiction that F_{i+1} is not a MST in G_{i+1}. Then, by Lemma 1, there exists an edge $e \in E_{i+1} \setminus F_{i+1}$ such that $F_{i+1} \cup \{e\}$ contains a unique cycle C with $w(e) < w(f)$ for some edge $f \in C \setminus \{e\}$. Since $e_{i+1} \in F_{i+1}$ and $e \notin F_{i+1}$, we have $e \neq e_{i+1}$ and therefore $e \in E_i$.
We will argue now that $F_i \cup \{e\}$ also contains a cycle C' such that e is not a heaviest edge in C'. This, however, contradicts then the fact that F_i is a MST, since we could swap in F_i the edge e with a heaviest edge in C' and create a spanning tree of less weight.
We consider two cases:

(a) First, suppose that $e_{i+1} \notin C$. Then, $C \subseteq E_i$ and C also constitutes a cycle in $F_i \cup \{e\}$ with the same property that e is not a heaviest edge in this cycle.

(b) Next, suppose that $e_{i+1} \in C$. Then, the symmetric difference $C' = C \oplus (C_{i+1} \setminus \{e_{i+1}\})$ (with $A \oplus B := (A \setminus B) \cup (B \setminus A)$) also forms a cycle that necessarily contains the edges f_{i+1} and e (see Figure 1). Two configurations are possible:

Suppose first that $f \in C'$ (top illustration in Figure 1) . Then we are done since $w(e) < w(f)$.

Next, suppose that $f \notin C'$ (bottom illustration in Figure 1). Then, we necessarily have that $f \in C_{i+1}$ and since the algorithm removed f_{i+1} from F_i instead of f, we have $w(f) \leq w(f_{i+1})$. Since $w(e) < w(f)$, we also have $w(e) < w_{f_{i+1}}$ and e is thus not the heaviest edge.

□

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.

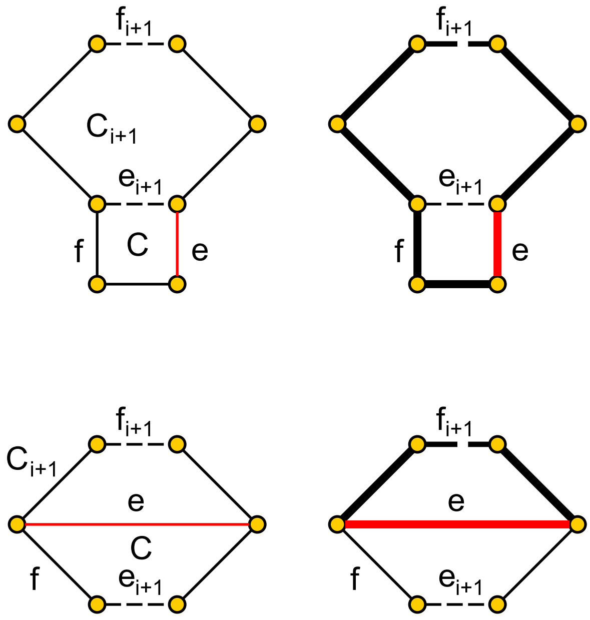

Figure 1: Solution to the MST exercise. Top: Case $f \in C'$. Bottom: Case $f \notin C'$.

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)?

Proof.
Since each set F_i is a spanning forest, we have $|F_i| \leq n - 1$. We thus store at most $k \cdot (n-1)$ edges. Accounting space $O(\log n)$ for the storage of an edge, we obtain space $O(kn \log n)$. Observe that this is a semi-streaming algorithm as long as $k = O(\text{poly} \log n)$. □

2. Prove that the algorithm is correct.

Proof.
Suppose first that (V, F) is k-connected. Then, since (V, F) is a subgraph of

G, we have that G is also k-connected. The algorithm is thus correct when it outputs G is k-connected.

Suppose now for the sake of a contradiction that G is k-connected but the algorithm outputs G is not k-connected. The fact that the algorithm outputs that G is not k-connected implies that (V, F) is not k-connected. Hence, we can remove at most $k - 1$ edges from (V, F) in order to disconnect (V, F), or, in other words, the graph (V, F) contains a cut, i.e., a partitioning of the vertex set into two parts $S, V \setminus S$, such that at most $k - 1$ edges connect V to $S \setminus V$. Observe that this also implies that there exists an index i such that F_i does not contain an edge across the cut $(S, V \setminus S)$. Recall that G is k-connected. Hence, there exists an edge $e \in E \setminus F$ that connects a vertex in S to a vertex in $V \setminus S$. However, this implies that when e arrived in the stream, it would have been inserted into F_i: Since F_i does not contain an edge crossing the cut, adding e to F_i would not create a cycle). This is a contradiction, which completes the proof. $\square$