Advanced Algorithms 2024 05/11 2024

Sample Solutions 07

Lecturer: Maximilian Probst *Teaching Assistant: Jakob Nogler*

1 Streaming a Minimum Spanning Tree

To construct a maximal spanning forest, we developed a streaming algorithm that uses $\tilde{O}(n)$ memory, handles edge additions and deletions, and in the end, given disjoint subsets $S_1, \ldots, S_t$ of vertices, outputs with high probability an edge crossing the cut $(S_i, V \setminus S_i)$, if one exists, for every $i \in [t]$.

To construct a maximal spanning forest, we ran Boruvka's algorithm for constructing a minimum spanning tree, ignoring the edge weights: we started with every vertex in its own component, then for $\log n$ rounds, found an arbitrary edge leaving each component, and added a maximal amount of those edges to the forest without creating a cycle. To find a minimum spanning tree instead, we need to find the minimum cost edge leaving each component instead of an arbitrary one.

1.1 Part A

For a 2-approximation, we can round the weight of every edge to a power of two, then find the exact minimum spanning tree. With this approach, we only have to consider $k = O(\log n)$ different edge weights (as the original edge weights are polynomially bounded).

We replace every instance of the streaming algorithm with k instances, where the ith only considers edges with weights at least 2^{i-1} but strictly less than 2^i (and rounds the weight of those edges down to 2^{i-1}). To find the cheapest edge leaving each current component, for every power of two we find up to one edge leaving the component of that weight, and then out of those select the minimum weight one.

This approach uses only a $O(\log n)$-factor more memory than the $\tilde{O}(n)$-memory algorithm for constructing a maximal spanning forest, as desired.

1.2 Part B

In part A, we found a 2-approximation of the minimum spanning tree. Now, we find the exact minimum spanning tree using $O(\log^2 n)$ passes, $O(\log n)$ for each of the $O(\log n)$ rounds of Boruvka.

We do this by employing binary search, more specifically a generic technique called *parallel binary search*. We consider some round of Boruvka. Let $S_1, \ldots, S_t$ be the components at the start of the round. For every component i, we have a lower bound $l_{i,j}$ and a upper bound $h_{i,j}$, with initial values $l_{i,0} = 1$ and $h_{i,0} = 2^k - 1$. In the jth subround of the round of Boruvka, we let $m_{i,j} = \left\lfloor \frac{l_{i,j-1}+h_{i,j-1}}{2} \right\rfloor$ and find if there exists an edge of weight $[l_{i,j-1}, m_{i,j}]$ between S_i and $V \setminus S_i$ for every $i \in [t]$ in one pass over the input. For every cluster i for which such an edge existed, we update $l_{i,j} = l_{i,j-1}$ and $h_{i,j} = m_{i,j}$, and for every cluster for which no such edge existed we update $l_{i,j} = m_{i,j} + 1$ and $h_{i,j} = h_{i,j-1}$. This approach finds for every cluster the minimum weight of an outgoing edge (and the edge itself) in $O(\log n)$ passes over the input.

It remains to perform one such pass. For this, we perform a simple modification to the streaming algorithm: each vertex in S_i will only consider edges with weights in $[l_{i,j-1}, m_{i,j}]$.

2 Streaming 3-Connectivity

Let T_1 be a maximal spanning forest of G, T_2 a maximal spanning forest of $G \setminus T_1$ and T_3 a maximal spanning forest of $G \setminus (T_1 \cup T_2)$. Then, G is 3-connected if and only if $T_1 \cup T_2 \cup T_3$ is 3-connected. To see this, note that a graph is 3-connected if and only if every cut has at least 3 crossing edges, and either each of T_1, T_2 and T_3 contain an edge crossing the cut (thus the union contains at least 3 total), or there are less than 3 edges crossing the cut.

To find these three maximal spanning forests, we run three copies of the streaming algorithm for a maximal spanning forest in parallel. To answer if the graph is 3-connected, we extract T_1 from the first instance, and perform edge deletions with edges of T_1 from the second and third instances. Then, we extract T_2 from the second instance, remove its edges from the third instance, and extract T_3 from the third instance.

Finally, to check if the graph $T_1 \cup T_2 \cup T_3$ is 3-connected, we can simply try deleting every combination of two edges, and check if the graph always remains connected. This check can easily be performed with linear memory usage to $|T_1 \cup T_2 \cup T_3| = O(n)$.

3 Communication with a Coordinator

Instead of sending one $O(\log n)$-bit message, each node will send $k = O(\log n)$ independent $b = O(\log n)$-bit messages (for parameters k and b, the exact values of which will be chosen later). This is equivalent (we can just concatenate the messages to form a $O(\log^2 n)$-bit string).

Let $a_i = \sqrt{2}^i$. To generate a b-bit message, the node sets each bit to 1 if they are **not** in S, and set the ith bit ($i \in \{0, 1, \ldots, b-1\}$) to 1 with probability $2^{-\frac{1}{a_i}}$ if they are in S.

Let $c_i \in [k]$ be the number of ANDs of messages where the ith bit is 1. The coordinator will return a_i, where i is the minimum integer such that $c_i \geq \frac{k}{2}$. Thus, if t is selected such that $a_t \leq s \leq a_{t+1}$ (where $s = |S|$), the coordinator returns a 2-approximation of s if $i \in \{t-1, t, t+1, t+2\}$. Note that this is guaranteed to happen if $c_i < \frac{k}{2}$ for all $i < t$, and $c_i \geq \frac{k}{2}$ for all $i > t+1$. To bound the probability of failure, we bound the probability that this does not happen.

We now analyse the probability that $c_i \geq \frac{k}{2}$. Consider the ith bit of some fixed message AND: it is 1 with probability exactly $2^{-\frac{s}{a_i}}$. If $s \leq a_{i-1} = \frac{1}{\sqrt{2}} a_i$, the probability the bit is 1 is at least $2^{-\frac{1}{\sqrt{2}}} \geq 0.6$, and if $s \geq a_{i+1} = \sqrt{2} a_i$, the probability it is 1 is at most $2^{-\sqrt{2}} \leq 0.4$.

Thus, for $i < t$, c_i is a sum of k independent random variables, each 1 with probability at most 0.4, and for $i > t+1$, $k - c_i$ is similarly a sum of k independent random variables, each 1 with probability at most 0.4.

We use Chernoff to bound the probability that the sum of these random variables is greater than $\frac{k}{2}$. Chernoff gives

$$\mathbb{P}\left(c_i \geq \frac{k}{2}\right) = \mathbb{P}\left(c_i \geq \left(1 + \frac{1}{4}\right)\mathbb{E}[c_i]\right) \leq \exp\left(-\frac{1}{16}\mathbb{E}[c_i]/(2 + \frac{1}{4})\right) \leq \exp\left(\frac{1}{90}k\right)$$

Thus, by a union bound, the probability of failure will be at most $b\exp(-\frac{1}{90}k)$ and selecting $k \geq 90(\log(n^2) + \log(b)) = O(\log n)$ is sufficient. The only requirement for b is that $a_t \leq s \leq a_{t+1}$ holds for some $t + 1 \leq b - 1$, thus we can simply select b to equal $2\lceil \log n \rceil + 2$ as $s \leq n$.