Task Discussion
Return of the Jedi

Daniel Graf

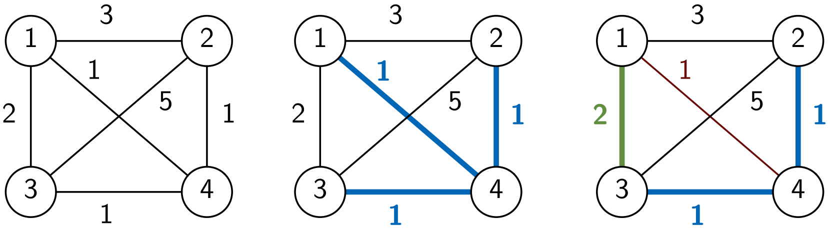
ETH Zürich

December 16, 2015

Problem Statement

First hurdle: Cut through the long story and find the algorithmic problem.

- **Given:** a complete, weighted, undirected graph $G = (V, E)$ on n vertices
- **Wanted:** the cost of the **second cheapest spanning tree** (the second cheapest spanning tree might not be unique and/or might have the same cost as the MST)
- **Subtasks:**
  - 40 points: $n \leq 100$
  - 40 points: $n \leq 1000$ and one (specific) MST is a star
  - 20 points: $n \leq 1000$



First Algorithm (40 points)

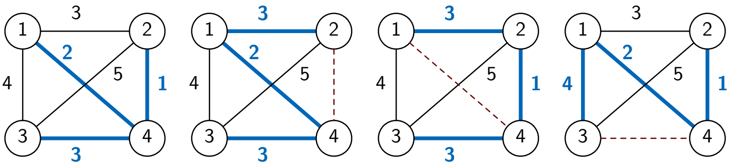
Lemma

The second cheapest spanning tree differs from a cheapest spanning tree by at least one edge.

This immediately gives us the following algorithm

- Take any minimal spanning tree T.
- For every edge e of T: compute the MST for $G' = (V, E \setminus \{e\})$ and minimize.

Runtime: $\mathcal{O}(n \cdot (n^2 \log n)) \rightarrow$ 40 points



c(T) = 6

c(T') = 8

c(T') = 7

c(T') = 7

Second Algorithm (80 points)

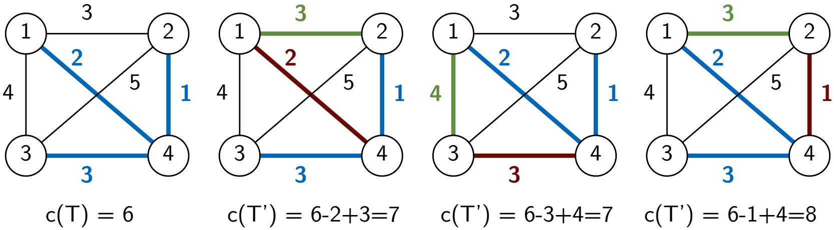
Lemma

There is a second cheapest spanning tree that differs from a cheapest spanning tree by exactly one edge. (Proof follows later)

Look at any MST T. There are two ways of using this lemma:

- If we remove the right edge $e \in T$, adding the cheapest edge that reconnects the graph gives the answer.

$\rightarrow$ we can solve the star testcase in $\mathcal{O}(n^2)$ with this.



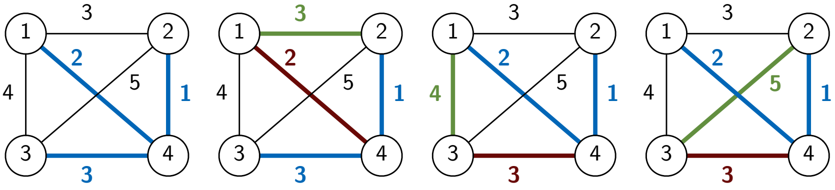
Second Algorithm (80 points)

Lemma

There is a second cheapest spanning tree that differs from a cheapest spanning tree by exactly one edge. (Proof follows later)

Look at any MST T. There are two ways of using this lemma:

- If we remove the right edge $e \in T$, adding the cheapest edge that reconnects the graph gives the answer.
- If we add the right edge $e \notin T$, removing the most expensive edge on the forming cycle gives the answer.



c(T) = 6 c(T') = 6+3-2=7 c(T') = 6+4-3=7 c(T') = 6+5-3=8

Third Algorithm (100 points)

Problem: How to find the most expensive edge on the forming cycle quickly?
Solution: Precompute it for all (v, w)-paths in the MST T in time $\mathcal{O}(n^2)$.
Then we can answer for each edge (v, w) in $\mathcal{O}(1)$.

- First, find the MST T.
- For every vertex $v \in V$ run a DFS over T starting at v.
  We find the most expensive edge on the path from v to u for all $u \in V$.
- Then, try adding all edges $e = (v, w) \notin T$ and look up the cost of the most expensive edge between v and w in T in time $\mathcal{O}(1)$ per edge.

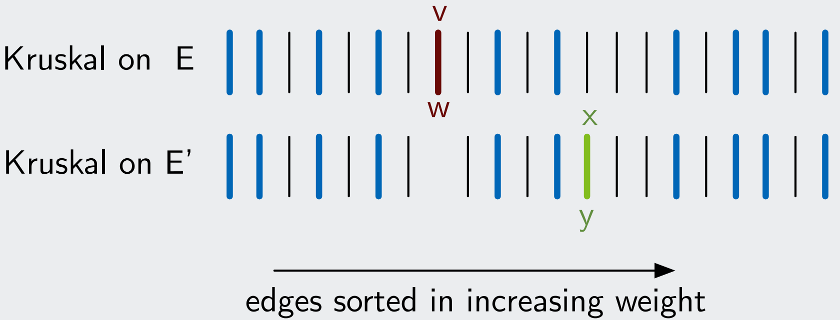
Implementation Details: If you use BGL's weight map on an `adjacency_list` to lookup the edge costs you get an overhead which causes a timelimit. Use `adjacency_matrix` or your own copy (e.g. in a `vector<vector<int> >`) for fast constant time access. (also applies to the 80 point subtask).

Single Swap Lemma

Lemma

There is a second cheapest spanning tree that differs from a cheapest spanning tree by exactly one edge.

Proof (by picture).



∎

Single Swap Lemma

Lemma

There is a second cheapest spanning tree that differs from a cheapest spanning tree by exactly one edge.

Proof (in words).

For the sake of reaching a contradiction, let T be a MST and T' a second cheapest spanning tree such that $|T \cup T'| - |T \cap T'| > 2$.
Let (v, w) be the cheapest edge that is in T but not in T'.
We now run Kruskal's algorithm in parallel on both E and $E' = E \setminus \{(u, v)\}$ (with ties broken the same way when sorting the edges).
Let (x, y) be the first edge that is only added in one of the two runs.
We must have $c((v, w)) \leq c((x, y))$ and $(x, y) \in T'$ and adding (x, y) to T creates a cycle that contains (v, w).
After adding (x, y) in the run on E', the connected components created up to this point are the same as on E and so Kruskal will also choose the same edges afterwards.
In the end, T and T' only differ by (v, w) and (x, y) – a contradiction. $\square$