CS173, Minimum Spanning Tree Algorithms

Tandy Warnow

December 5, 2018

CS 173
Minimum Spanning Tree Algorithms
Tandy Warnow

Today's agenda

Tuesday we did:

- Minimum spanning tree (MST)
- Algorithms for MST construction
- 2-approximation for Triangle-TSP (Travelling Salesman Problem)

Today we will prove Kruskal's algorithm correct

If we have time, we'll also see a 2-approximation for Minimum Vertex Cover!

Spanning Trees

A **spanning tree** of a connected graph $G = (V, E)$ is a subgraph that includes all the vertices and is a tree.

Minimum Spanning Trees (MST)

- Input: Connected graph $G = (V, E)$ and positive edge weights $w : E \rightarrow Z^+$
- Output: Spanning tree $T = (V, E')$ of G that has minimum cost, where $cost(T) = \Sigma_{e \in E'} w(e)$

Finding MSTs

Try one of the greedy algorithms on the complete bipartite graph $K_{3,5}$ with $w(v_i, w_j) = i + j$:

- Keep adding the least weight edges (don't include those that create cycles) - Kruskal's algorithm
- Keep deleting the most costly edges (don't delete bridges)
- Grow a spanning tree, adding least costly edge to an unvisited vertex - Prim's algorithm

Finding MSTs

Running Kruskal's algorithm on $K_{3,5}$ with weight $w(v_i, w_j) = i + j$:

Why does Kruskal's algorithm work?

We will prove that Kruskal's algorithm always returns a minimum cost spanning tree, when the input is a connected graph with positive edge weights.

Fairly straightforward to see that Kruskal's algorithm returns a spanning tree.

Now we show that the spanning tree output by Kruskal's algorithm has minimum cost.

Why does Kruskal's algorithm work?

We show that the spanning tree returned by Kruskal's algorithm has minimum cost for the special case where no two edges have the same weight.

Why does Kruskal's algorithm work?

Let $G = (V, E)$ be a connected graph in which every two edges have different weights.

Let T be the spanning tree returned by Kruskal's algorithm and T^* be a minimum spanning tree.

We prove $T = T^*$ by contradiction!

If $T \neq T^*$, then T^* has at least one edge that is not in T.

Let e be any such edge in T^* that is not in T.

Why does Kruskal's algorithm work?

Remember $e = (x, y) \in E(T^*) \setminus E(T)$

Since e is not in $E(T)$, it must be that when e is considered it would create a cycle if added to T.

If we add e to T we obtain a cycle γ and T has a path P from x to y given by $x, v_1, v_2, \ldots, v_k, y$.

Because $e \notin E(T)$, it must be that $w(e) > w(e')$ for all $e' \in E(P)$ (otherwise Kruskal's algorithm would have added e).

Why does Kruskal's algorithm work?

We have shown: $e = (x, y) \in E(T^*) \setminus E(T)$ and $w(e) > w(e')$ for all edges on the path P between x and y in T.

Consider $T^* - e$ (the graph obtained by deleting the edge e from T^*).

It has two components, A and B, with $x \in A$ and $y \in B$. Also, no edge in T^* besides e has an endpoint in A and another endpoint in B.

Why does Kruskal's algorithm work?

$T^* - e$ has two components, A and B, with $x \in A$ and $y \in B$.

Let f be an in P has an endpoint in A and an endpoint in B.

Note that $f \notin E(T^*)$

Remember we showed $w(f) < w(e)$

Think about $T^{**} = T^* - e + f$ (the graph obtained by deleting e from T^* and adding f)

Why does Kruskal's algorithm work?

Remember: $T^{**} = T^* - e$ had two components A and B, and f also connects these two components.

Therefore T^{**} is a spanning tree for G

Remember we showed $w(f) < w(e)$

Therefore $w(T^{**}) = w(T^*) - w(e) + w(f) < w(T^*)$

This contradicts T^* being a MST!

Hence T is a MST!

Why does Kruskal's algorithm work?

We summarize the proof by contradiction:

- We let T^* be a MST different from T
- We took any edge e from T^* that wasn't in T, and we added it to T
- We argued this created a cycle γ and that e had to be the heaviest edge in γ
- We argued there was an edge f in γ that isn't in T^* and we could add it to $T^* - e$ and get a spanning tree that had smaller weight
- Hence we obtained a contradiction

What we showed

We showed that if all edges have different weights, then there is only one MST, and Kruskal's algorithm produces it.

If edges can have the same weight, a slightly different proof is needed to show that Kruskal's algorithm produces a tree with the same optimal weight.