Lecture 21

CSE 331

Mar 25, 2020

Minimum Spanning Tree Problem

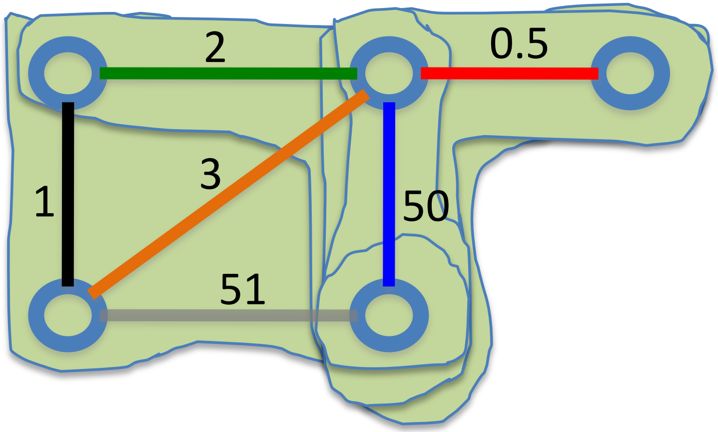
Input: Undirected, connected $G = (V,E)$, edge costs c_e

Output: Subset $E' \subseteq E$, s.t. $T = (V,E')$ is connected
$C(T)$ is minimized

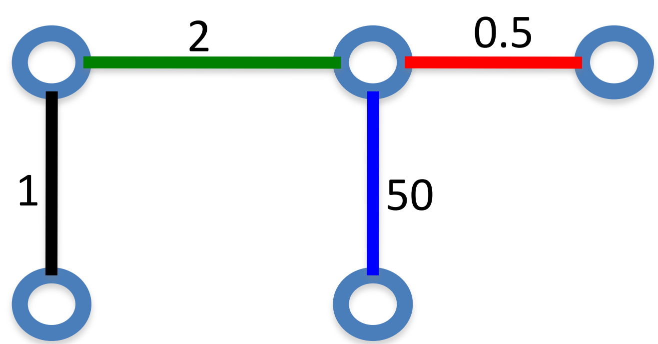
If all $c_e > 0$, then T is indeed a tree

Prim's algorithm

Similar to Dijkstra's algorithm



Robert Prim



Input: $G=(V,E)$, $c_e > 0$ for every e in E

$S = \{s\}$, $T = \emptyset$

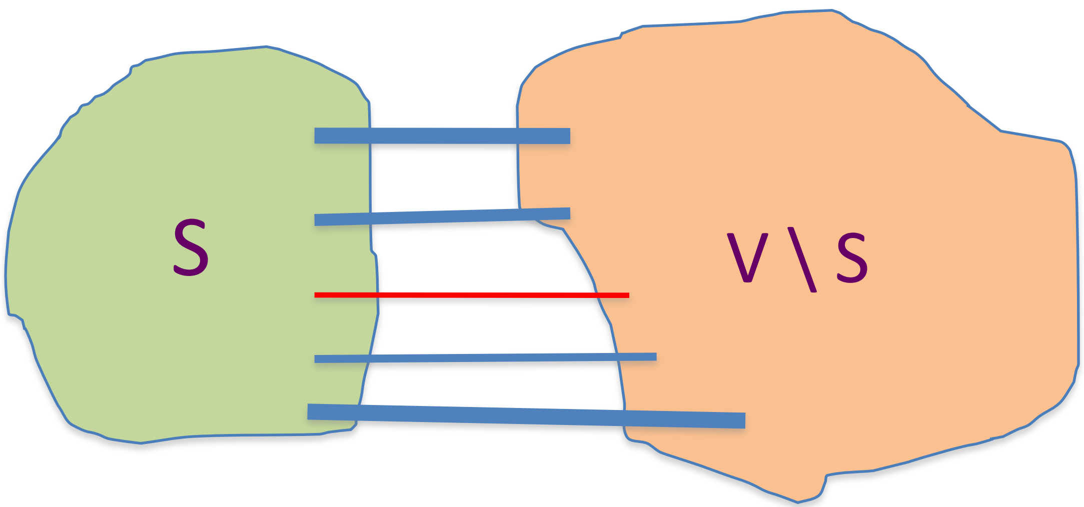
While S is not the same as V

    Among edges $e = (u,w)$ with u in S and w not in S, pick one with minimum cost

    Add w to S, e to T

Cut Property Lemma for MSTs

Condition: S and V\S are non-empty



Cheapest crossing edge is in **all** MSTs

Assumption: All edge costs are distinct

Today’s agenda

Optimality of Prim’s algorithm

Prove Cut Property Lemma

Optimality of Kruskal’s algorithm

Remove distinct edge weights assumption

Kruskal's Algorithm

Joseph B. Kruskal

Input: $G=(V,E)$, $c_e > 0$ for every e in E

$T = \emptyset$

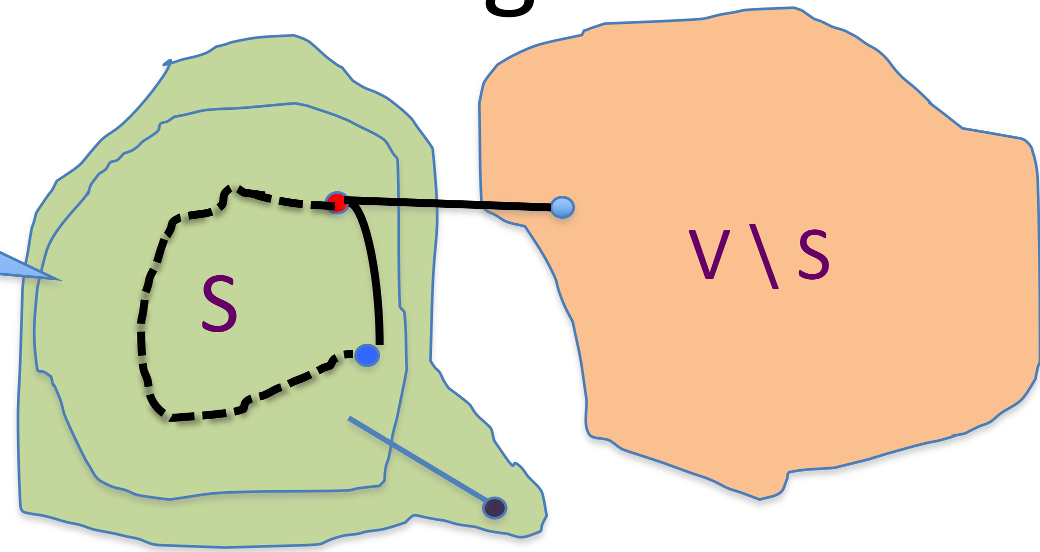
Sort edges in increasing order of their cost

Consider edges in sorted order

If an edge can be added to T without adding a cycle then add it to T

Optimality of Kruskal's Algorithm

Nodes connected to red in (V,T)



Input: $G=(V,E)$, $c_e > 0$ for every e in E

$T = \emptyset$

Sort edges in increasing order of their cost

Consider edges in sorted order

If an edge can be added to T without adding a cycle then add it to T

S is non-empty

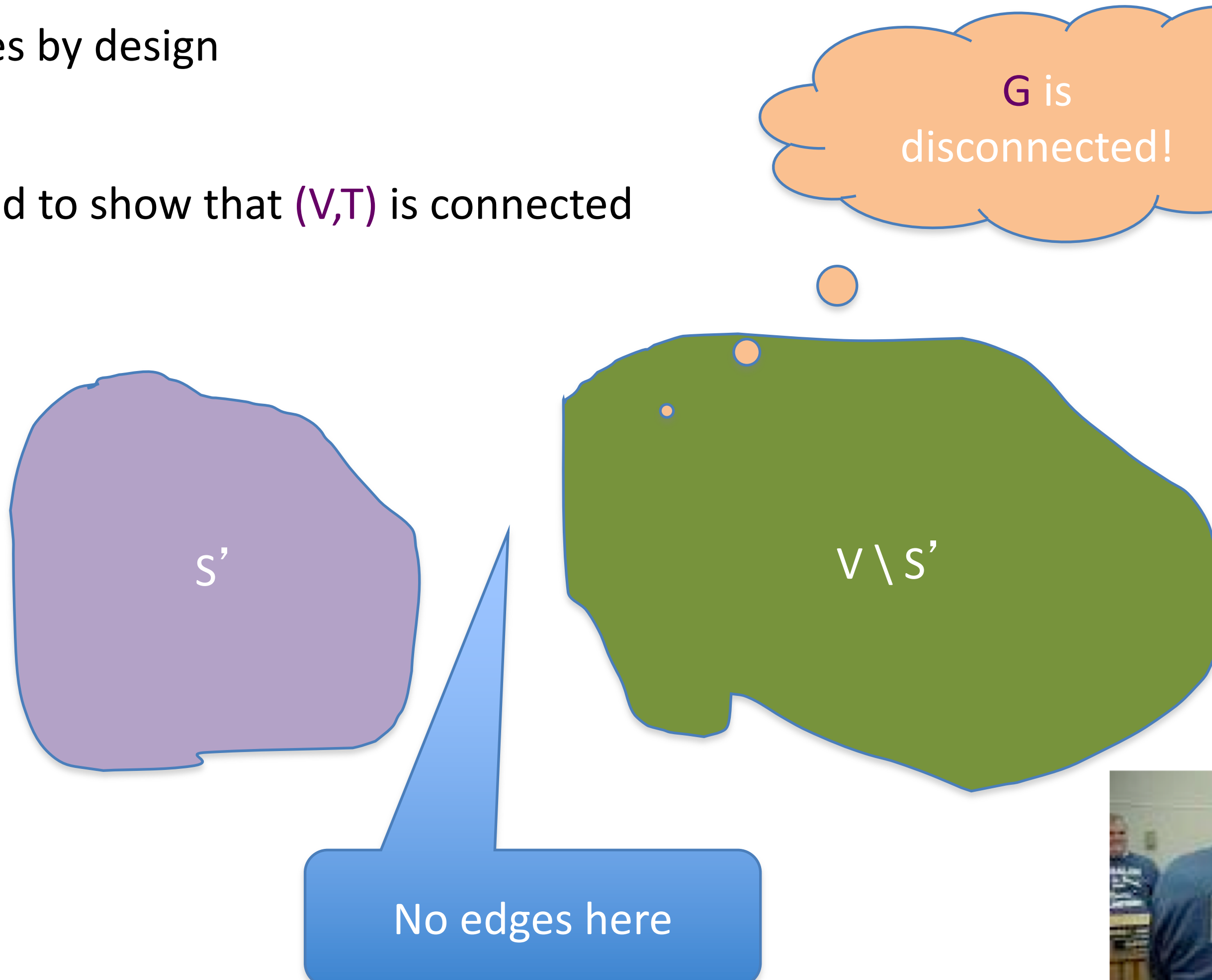
V\S is non-empty

First crossing edge considered

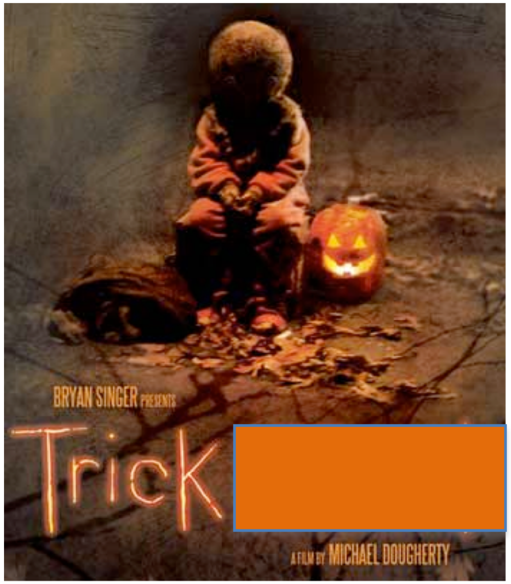
Is (V,T) a spanning tree?

No cycles by design

Just need to show that (V,T) is connected



Removing distinct cost assumption



Change all edge weights by very small amounts

Make sure that all edge weights are distinct

MST for "perturbed" weights is the same as for original

Changes have to be small enough so that this holds

Figure out how to change costs

Run time for Prim's algorithm

Similar to Dijkstra's algorithm

O(m log n)

Input: G=(V,E), c_e> 0 for every e in E

S = {s}, T = Ø

While S is not the same as V

Among edges e= (u,w) with u in S and w not in S, pick one with minimum cost

Add w to S, e to T

Running time for Kruskal's Algorithm

Can be implemented in $O(m \log n)$ time (Union-find DS)

Joseph B. Kruskal

Input: $G=(V,E)$, $c_e > 0$ for every e in E

$O(m^2)$ time overall

$T = \emptyset$

Sort edges in increasing order of their cost

Consider edges in sorted order

If an edge can be added to T without adding a cycle then add it to T

Can be verified in $O(m+n)$ time

Reading Assignment

Sec 4.5, 4.6 of [KT]