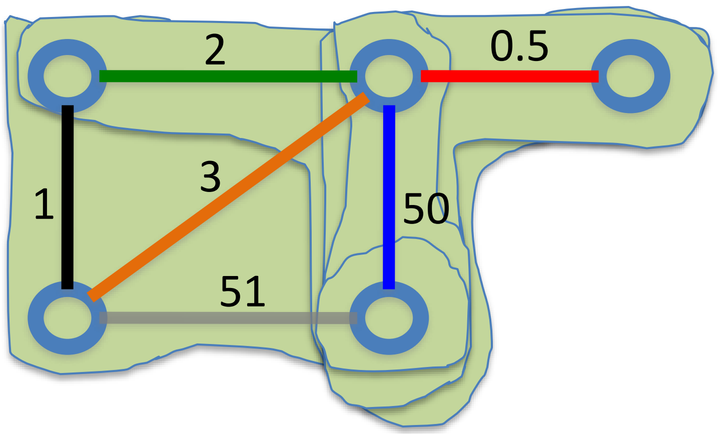
Lecture 22

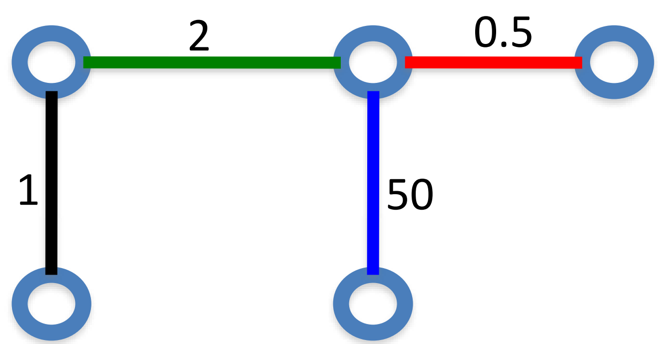
CSE 331

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 = Ø

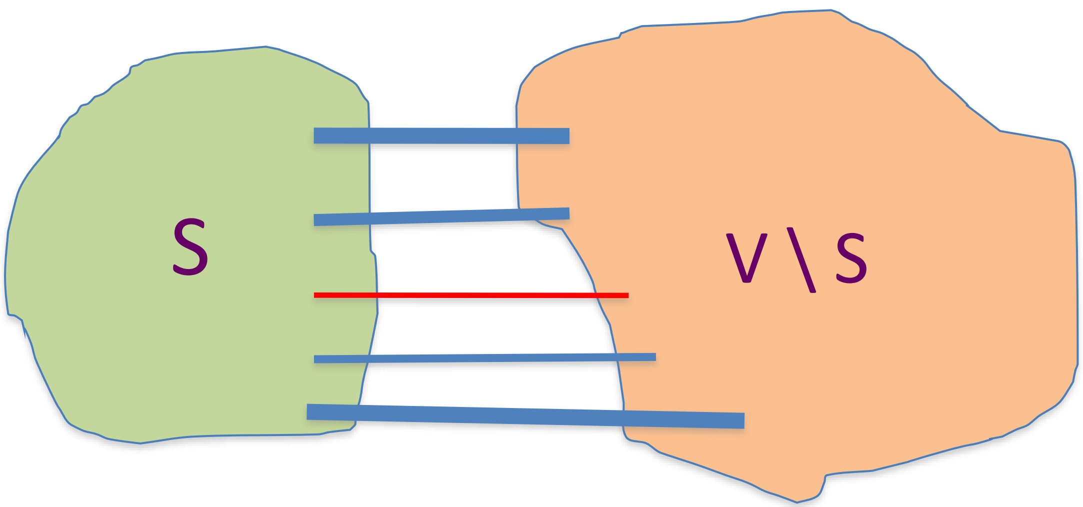
While S is not the same as V

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

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

Agenda

Optimality of Prim’s algorithm

Prove Cut Property Lemma

Optimality of Kruskal’s algorithm

Remove distinct edge weights assumption

On to the board…

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

Kruskal's Algorithm

Theorem 2: Kruskal's algorithm is correct.

(Similar to correctness of Prim's)

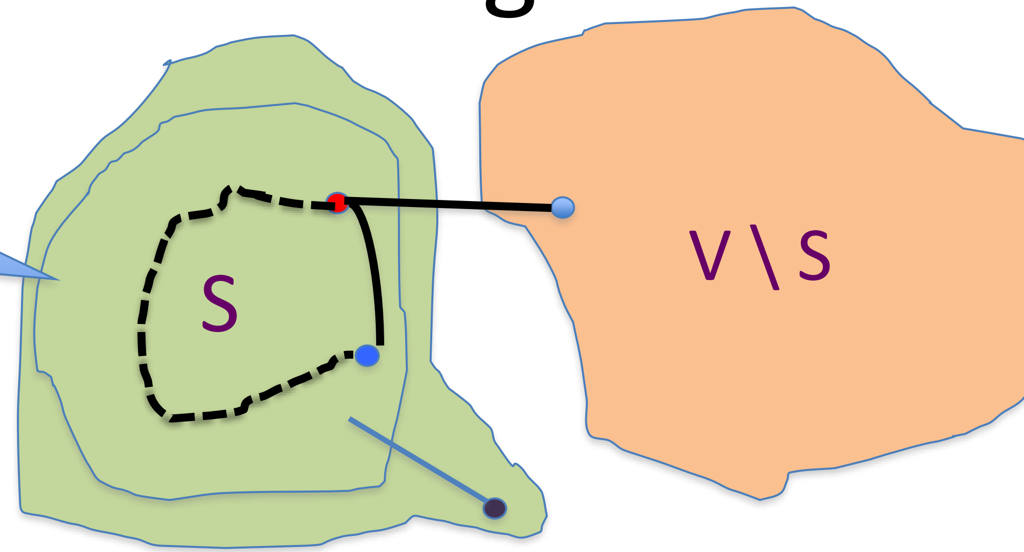
Consider a run of the algorithm when it is about to add edge (u, w) to T.

<u>Goal</u>: show that e is the cheapest "crossing" edge across some cut (S, V\S).

<u>Define S:</u>
Let S be the set of vertices connected to u using only the edges in T (i.e., u has a path to all nodes in S).

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