Minimum Spanning Tree (MST)

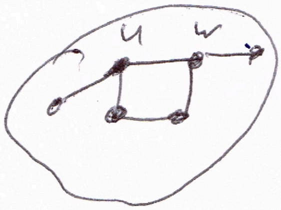
<u>Input:</u> $G=(V,E)$ $\quad c_e > 0 \quad \forall e \in E$

<u>Output:</u> $T \subseteq E$ s.t. $G' = (V, T)$ is connected.

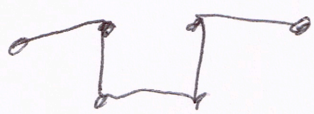
<u>goal:</u> minimize $cost(T) = \sum_{e \in T} c_e$

<u>Proposition:</u> If G' has min cost, the G' is tree.

<u>proof:</u> by contradiction. assume G' is not a tree. Then we know G' has a cycle. Remove an edge from the cycle: $H = G' \setminus \{(u,w)\}$



H is connected through the path.

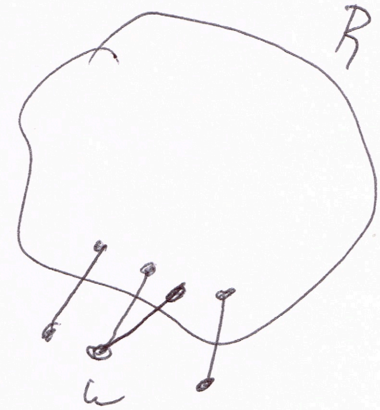


$$Cost(H) = cost(G') - c_{(u,w)} < Cost(G')$$

$\therefore$ G' is not optimal

Prim's Algo:

build R iteratively
pick w with the
minimum cost crossing edge.
Add w through that edge



Kruskal's algo:

- sort the edges by cost
- add edges in order unless it creates a cycle in R.

Reverse Delete algo:

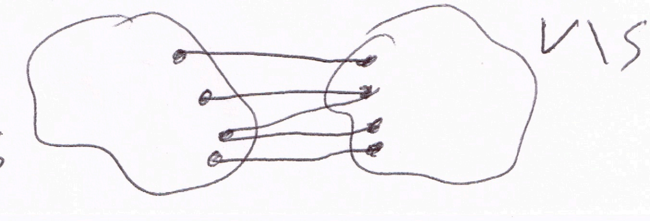
- sort edges in decreasing orde of cost
- remove edges unless it disconnects the graph.

Correctness of Kruskal's + Prim's algos

We'll use the cut property Lemma $\boxed{\text{To be proven later}}$

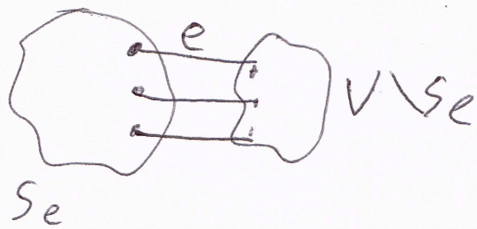
$\forall S \subset V$ s.t. $S \neq \emptyset$

The cheapest crossing edge is in all MST's of G



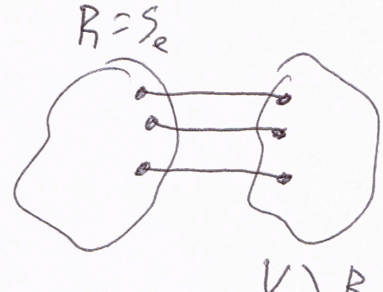
Assume cut property is true (we will prove this later)

<u>idea for both proofs</u>: For every edge e that is added by the algos, argue that there exists a cut S_e s.t. e is the cheapest crossing edge



Correctness of Prim's algo

By definition, Prim's picks the cheapest crossing edge.



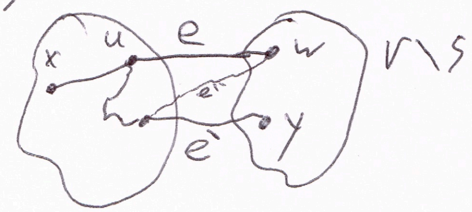
At any step of the algo, $S_e = R$ algo takes the cheapest crossing edge. This edge must be in any MST.

$\therefore$ The algo only adds edges that must be in any MST.

- no cycles since e never connects nodes both in R
- Connected since G is guarenteed to be connected.

Correctness of kruskal's algo

When the algo adds edge $e=(u,w)$ to T,
let S_e be the set of all verticies connected to u by edges in T before e is added.
$S = CC(u)$



Cut property conditions

(i) $S_e \neq \emptyset$ $(u \in S)$ ✓

(ii) $S_e \subset V$ $(w \notin S)$ ✓

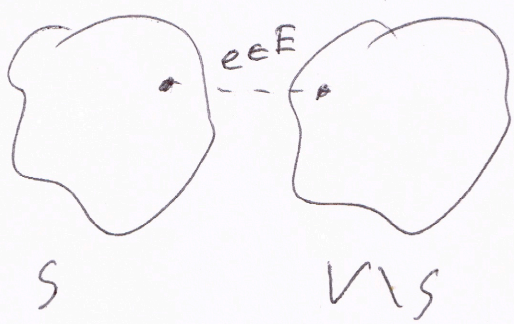
(iii) e is the cheapest crossing edge ✓

(iv) final output (V,T) is connected.

(iii) if e' is cheaper, the algo must have already considered it, ~~it~~ ~~it connected~~ It must not exist since it doesn't cause a cycle. For it to cause a cycle, $y \in S$ and it's not a crossing edge. y can be w.

(iv) Final output is connected

By contradiction, assume (V,T) is not connected, must $\exists e \in E$ s.t. ...



All edges are considered and added if it doesn't create a cycle.
e doesn't create a cycle since there are no other crossing edges in T. ✗