# Lec02-Kruskal

KT 4.5 / 4.6

Tuesday, August 27, 2024 12:38 PM

We just saw how the greedy Prim's algorithm for MST always works. Other greedy algorithms also work.

## Greedy MST rules

Prim's: start any node, add frontier edge (edge on boundary of growing tree) with smallest weight

Kruskal's: add edges in increasing order of weight, skipping any that would create a cycle

Reverse-Delete: Start with all edges. Delete in decreasing order of weight, skipping any that would disconnect the graph.

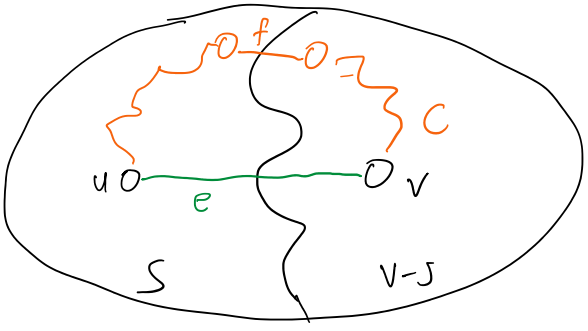
Reminder: In last section, proved smallest edge in any Cut must be in MST.

**Thm (Cycle Prop):** Let $C$ be a cycle in $G$. Let $e=(u,v)$ be the edge with max weight. Then $e$ is NOT in any MST of $G$.

Intuition: can always replace edge with another lower-cost edge in $C$.

proof: Suppose $e \in T$, where $T$ is a MST.

Deleting $e$ partitions $T$ into two sets $S \ni u$ and $V-S \ni v$.




Cycle $C$ must have another edge $f$ that crosses over from $S$ to $V-S$

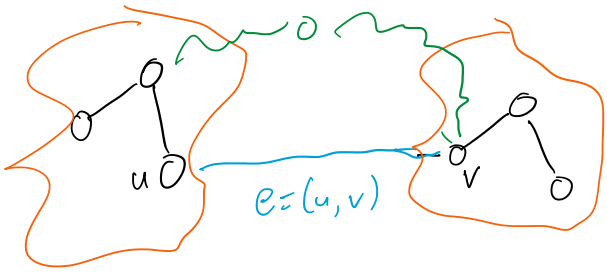
$cost(f) < cost(e)$, so we should replace $e$ with $f$ in $T$, getting a lower-cost spanning tree. →|← contradiction

i.e. the heaviest edge on any cycle is never in any MST, because can replace w/ another edge and still have a spanning tree.

Makes both Kruskal + Reverse-Delete work.

**Thm** Reverse-Delete produces a MST.

**proof.**

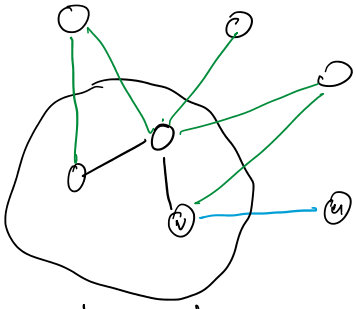



Let e be next edge removed. By construction, e's removal would disconnect the graph, so must be another path b/t u & v.

So we must be removing the largest edge in cycle, which is never in the MST anyway. ▨

**Thm** Kruskal produces a MST

**proof.** Consider the point when edge $e=(u,v)$ is added.




S = nodes in v's connected component in T before adding e.

u is in V−S (otherwise would be a cycle)

We get a tree because we never create cycles, and if T is disconnected, can always add another edge w/o creating a cycle

All rejected edges would be the max weight edge in a cycle, so it is correct they never be added to a MST.

---

Revisiting unique edge weights

## Revisiting unique edge weights

In all three methods, we don't say what to do if two edge weights are the same. Instead, we made them unique by adding small unique values. How can we be sure this works?

**Claim:** Given graph $G$ with edges $\{e_1, \dots, e_m\} = E$ and weights $d(e_i) \geq 0$,

let $\Delta = \frac{1}{2} \min_{T_i, T_j} \{ |cost(T_i) - cost(T_j)| \text{ s.t. } cost(T_i) \neq cost(T_j) \}$ over all spanning trees $T_i$

↑ half the smallest difference b/t any two non-equal tree costs

$\forall \varepsilon_1, \dots, \varepsilon_m$ s.t. $\varepsilon_i > 0$ & $\sum_i \varepsilon_i < \Delta$, let $\hat{d}(e_i) = d(e_i) + \varepsilon_i$.

Then any MST $\hat{T}$ with these modified weights is also a MST of the original graph $G$ with weights $d(e_i)$.

**proof.** Let $cost(\hat{T})$ be the cost with original weights

$\hat{cost}(\hat{T})$ be cost with modified weights

Suppose $\exists$ MST $T$ of $G$ s.t. $cost(T) < cost(\hat{T})$.

Then $\hat{cost}(T) < cost(T) + \Delta < cost(\hat{T}) < \hat{cost}(\hat{T})$

↑ by def of $\varepsilon_i$'s ↑ by def of $\Delta$ ↑ by def of $\hat{cost}$

So $\hat{T}$ is not a MST with the modified weights, a contradiction. $\rightarrow|\leftarrow$

$\Rightarrow cost(T) = cost(\hat{T})$

$\Rightarrow \hat{T}$ is a MST even with original weights

---

Correctness ← done

Implementation / Data Structures ← need lecture

Runtime Asymptotics ← specific to implementation