Lec02-Kruskal

Tuesday, August 25, 2026 12:04 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

WRONG: start any node s. Add frontier node (node connected to T by frontier edge) that is closest to s. (Dijkstra)

WRONG: start with all edges
Pick an arbitrary cut of the graph. Delete the max weight edge.
Repeat until there are no cycles in the graph.

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

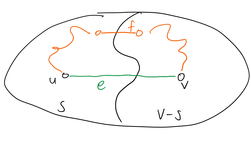
Thm (Cycle Prop): Let C be a cycle in G. Let $e=(u,v)$ be the edge with max weight. Then e is NOT in any MST of G.

proof? Prim's works. In order to add an edge to a growing tree, that edge must be the lowest cost frontier edge, so suppose we add e. Then there must be some cut $(S, V-S)$ such that e is the minimum edge crossing that cut.
WLOG, say $u \in S$, $v \in V-S$.
But in the cycle C, another edge f must cross the Cut, and $cost(f) < cost(e)$, leading to a contradiction.
Therefore, Prim's will never choose e, so e is not in the MST. ?

WRONG, because we never proved Prim's returns all possible MSTs.

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

proof. Suppose $e \in T$, where T is an MST.
Deleting e partitions T into two sets $S \ni u$ and $V-S \ni v$.



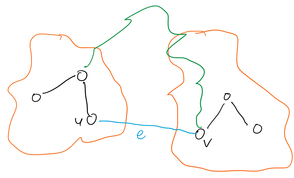
Cycle C must have another edge f that crosses the Cut.
$cost(f) < cost(e)$, so we should replace e with f in T, getting a lower-cost spanning tree.

$\rightarrow|\leftarrow$ contradiction.

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

Makes both Kruskal & Reverse-Delete work.

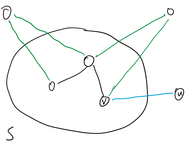
Thm Reverse-Delete produces a MST.



proof. Let e be next edge removed.
By construction, e's removal does NOT disconnect the graph, so there must be another path b/t u & v.
$\Rightarrow$ e is the largest edge in a cycle, which is never in a MST anyway.

Thm Kruskal produces a MST.

proof. Consider the point when edge $e=(u,v)$ is added.
Let S = nodes in v's connected component. (before adding e)
Then $u \in V-S$ (otherwise e would create a cycle).



All rejected edges would create a cycle, and since we add edges in increasing order of weight, they would be the max edges in a cycle.
$\Rightarrow$ rejected edges aren't in any MST.

At end of Kruskal, we have a spanning tree and only rejected edges that aren't in a MST $\Rightarrow$ we have a MST.

Revisiting unique edge weights

In all 3 methods, we don't say what to do if two edge weights are the same. Instead we assumed they weren't or added random noise as a tie-breaker. 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 $\Delta = \frac{1}{2} \min_{T_i, T_j} \left\{ |cost(T_i) - cost(T_j)| \text{ s.t. } cost(T_i \neq T_j) \right\}$ over all spanning trees T_i

(Δ: half the difference between 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 the 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.
Let $\hat{cost}(\hat{T})$ be the cost with modified weights.

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

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

($cost(T) < cost(T) + \Delta$: by def. of ε_i's; $cost(T) + \Delta < cost(\hat{T})$: by def. of Δ; $cost(\hat{T}) < \hat{cost}(\hat{T})$: by def of $\hat{cost}$)

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

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

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

Question: instead of adding noise, could you reframe our logic to deal with non-unique weights instead?

How does this work when a graph has multiple MSTs?

Correctness ← done

Implementation / Data structures ← next lecture

Runtime asymptotics ← specific to implementation