Lec08-minimum-spanning-arborescence

KT 4.9

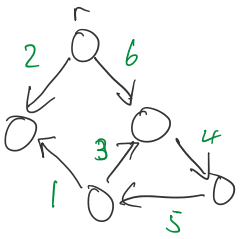
Tuesday, September 12, 2023 1:53 PM

We saw Prim's & Kruskal for MST on undirected graphs
What if the graph is directed?

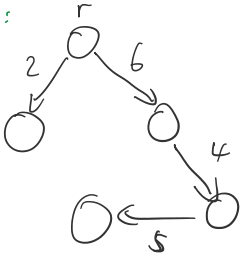
Define: An arborescence of a directed graph $G=(V,E)$ rooted at r is a subgraph of G with no cycles s.t. $\exists$ a path from r to every other vertex.

Problem: Given weights $w_{uv} \geq 0$ on each edge $(u,v) \in E$ and a vertex r, find the minimum cost arborescence (MCA) rooted at r.

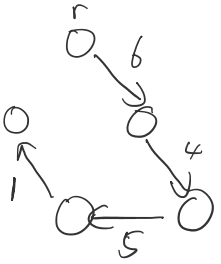
Can we modify Prim's? Each time at lowest cost frontier edge



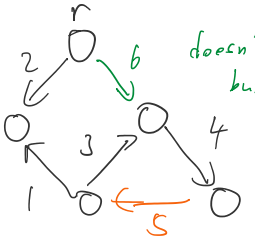
Prim's:



MCA



What about Kruskal?

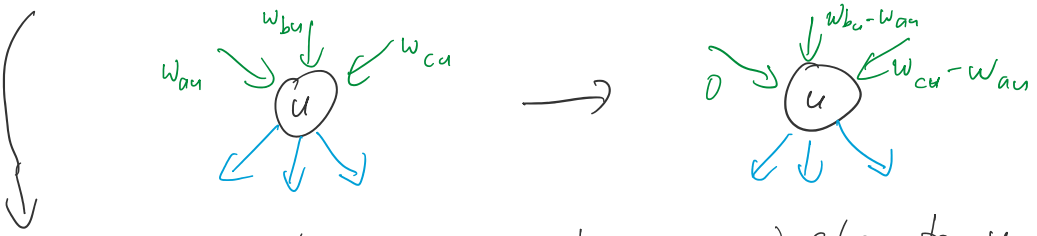


doesn't create cycle but also not an arborescence

creates cycle (how to detect)

Difficulty: Cycles & connectedness more complicated for directed graph.

Notice: Let w_{au} be the smallest weight of an in-edge of $u \neq r$. Subtracting w_{au} from all in-edges doesn't change MCA.



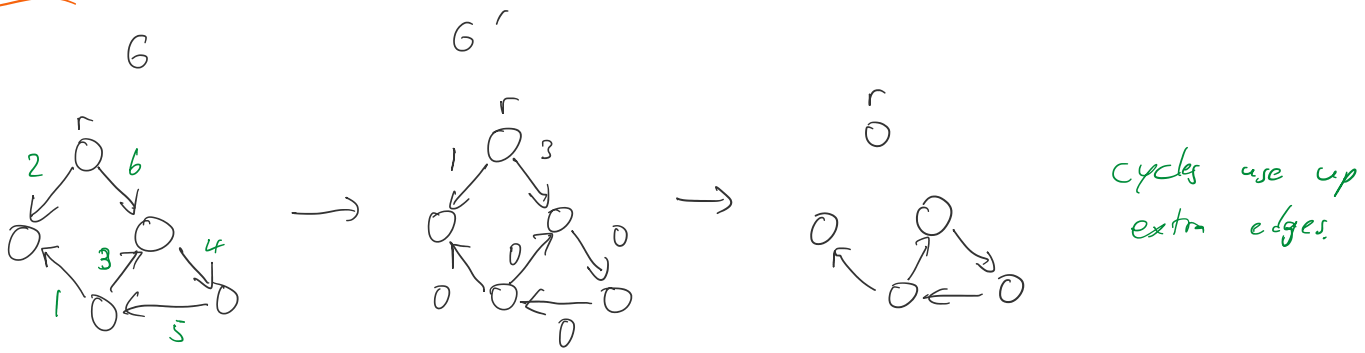
because every arborescence must have an in-edge to u, so we subtracted w_{au} from every one.
Also doesn't change which edge is lightest.

Attempt at alg'
1. Transform G into G' by subtracting smallest incoming edge weight from all incoming edge.
2. Choose T by selecting arbitrary 0-weight in-edge for each node $u \neq r$.

If T is an arborescence, it is optimal.

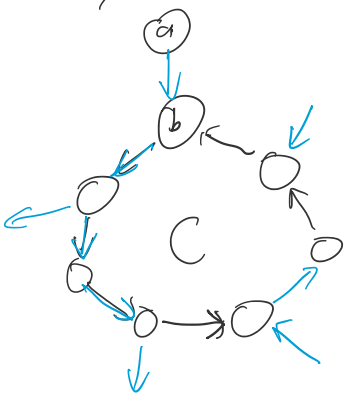
Why? T has cost 0. And no MCA has lower cost.

Problem? Cycles



Fix: Analyze cycles

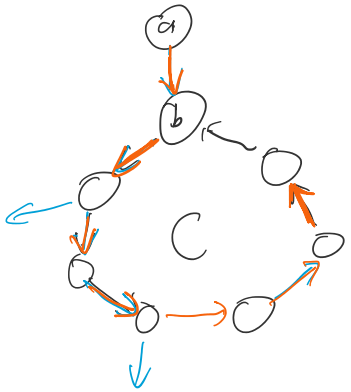
Let C be a cycle in T & let T^* be an optimal MCA



T^* may enter C several times

Want $\tilde{T}^*$ another MCA to enter C only once.

- Let (a,b) be the edge entering C closest to the root in T^*.
- Remove all edges entering C except (a,b)
- Add all edges in C except the one going to b



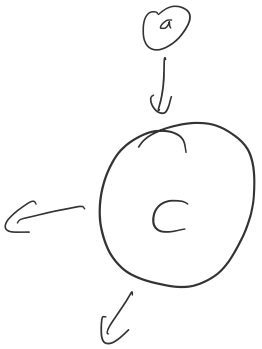
Verification: $\text{cost}(\tilde{T}^*) \leq \text{cost}(T^*)$ since we replaced edges with 0 weight edges

Also, still an arborescence, since we can still get to every node in the cycle, and the only edges deleted are entering C.

Thm There is an MCA that only enters C once.

Thm There is an MCA that only enters C once.

$\Rightarrow$ don't need to keep track of internal structure of C and can treat as a supernode.



Idea: Find the MCA in this new, smaller graph recursively.

Procedure MCA:

- Transform G into G' by subtracting min in-edge weight from all in-edges of nodes.
- Choose T by selecting an arbitrary 0-weight in-edge for every node.
- If no cycles, done. (can check w/ BFS in $O(n)$ time)
- If cycle C, collapse to super node C' to get new graph G''.
- Find MCA on G'' (recursively)
- Expand supernodes back into almost cycles (lacking only last edge) to T'
- Return tree T'

Correctness: The MCA procedure works.

proof. By induction on size of graph G.

When G has 2 nodes, can't be any cycles. (base case)

By induction, find MCA on G'' (since $|G''| < n$).

$\Rightarrow$ gives MCA on G' by expanding cycle supernode.

$\Rightarrow$ gives MCA on G since cost transformations don't change MCA.

Runtime: Cost transformation + selecting 0-weight edges $O(|E|)$

Checking for cycles $O(|V|)$

Each recursive step shrinks graph by ≥ 1 node.

$\Rightarrow$ $O(|V|)$ recursive steps.

$\Rightarrow O(|V|)$ recursive steps.

$\Rightarrow O(|E||V|)$ runtime.