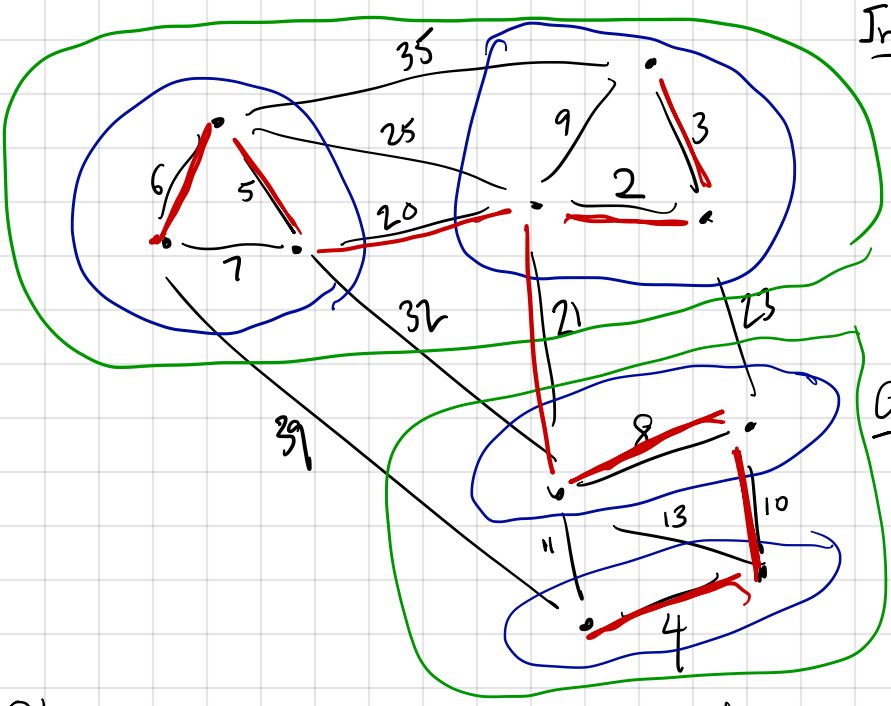
5 Feb. 2018 The Minimum Spanning Tree Problem.

First solved by Borůvka working on the rural electrification of Moravia.



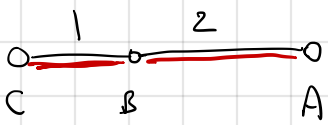
Input: Connected undirected graph.

Edge costs: a positive number $c(e)$ for each edge e.

Goal: Find a connected subgraph with minimum combined edge cost.

Observation: Minimum cost connected subgraph will always be a tree. (If it contained a cycle, you could remove any edge, preserve connectedness while reducing cost.)

Borůvka's Algorithm: Connect everyone to its nearest neighbor. Contract connected components to single vertices, recurse on the remaining graph.



Tie-Breaking Rule: If a node has 2 edges of equal cost leaving it, and they both have the minimum cost, the algorithm is allowed to pick either one.

From now on (for rest of lecture) assume all edge costs are distinct — for simplicity, not because correctness depends on that assumption.

Kruskal's Algorithm. Iterate through edges in order of increasing cost, always choosing the cheapest edge that doesn't create a cycle.

Reverse Delete. Iterate through edges in order of decreasing cost, always deleting the most costly edge that doesn't disconnect the graph.

Prim's Algorithm. (Analogous to Dijkstra, but creates a tree instead of a path.)

Maintain a set S of "discovered" vertices. (Vertices that have been added to tree so far.)

Pick arbitrary vertex r, initialize $S = \{r\}$.

Until $S = V$:

| let (u,v) be cheapest edge from S to $V \setminus S$.
| insert (u,v) into T
| insert v into S.

end

output T.

All four of these algorithms are correct!!

Why is the spanning tree problem so conducive to correct greedy algorithms?

Recall. We are assuming all edge costs are distinct.

CUT PROPERTY For partition of the vertices V into two non-empty sets A, B, the cheapest edge from A to B must belong to every minimum spanning tree.

CYCLE PROPERTY For every cycle C in the graph, the most costly edge of C cannot belong to any minimum spanning tree.

E.g. Every time Boruvka's Algorithm includes an edge in the tree, the inclusion of that edge is justified by the cut property. Similarly for Prim's.

Kruskal's Algorithm. Iterate through edges in order of increasing cost, always choosing the cheapest edge that doesn't create a cycle.

Reverse Delete. Iterate through edges in order of decreasing cost, always deleting the most costly edge that doesn't disconnect the graph.

CYCLE PROPERTY For every cycle C in the graph, the most costly edge of C cannot belong to any minimum spanning tree.

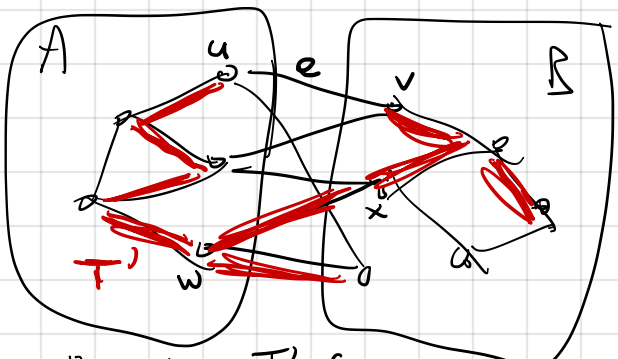
E.g. Kruskal only omits an edge e if it forms a cycle with edges that were already picked. The other edges of that cycle are cheaper than e, else they wouldn't have all been selected before e when iterating through edges in increasing cost order.

Exercise. (Not to hand in) Reverse Delete algorithm can also be justified using cycle property.

CUT PROPERTY For partition of the vertices V into two non-empty sets A, B, the cheapest edge from A to B must belong to every minimum spanning tree.

Proof of cut property. By exchange argument.

Suppose $e = (u,v)$ is minimum cost edge crossing the cut.



To form T, trace the path in T' from u to v until it crosses from $w \in A$ to $x \in B$. Remove (w,x) from T' and insert (u,v) to create T.

Exchange argument must show how to transform any tree T' that doesn't include e into T that includes e and $\text{cost}(T) \leq \text{cost}(T')$.