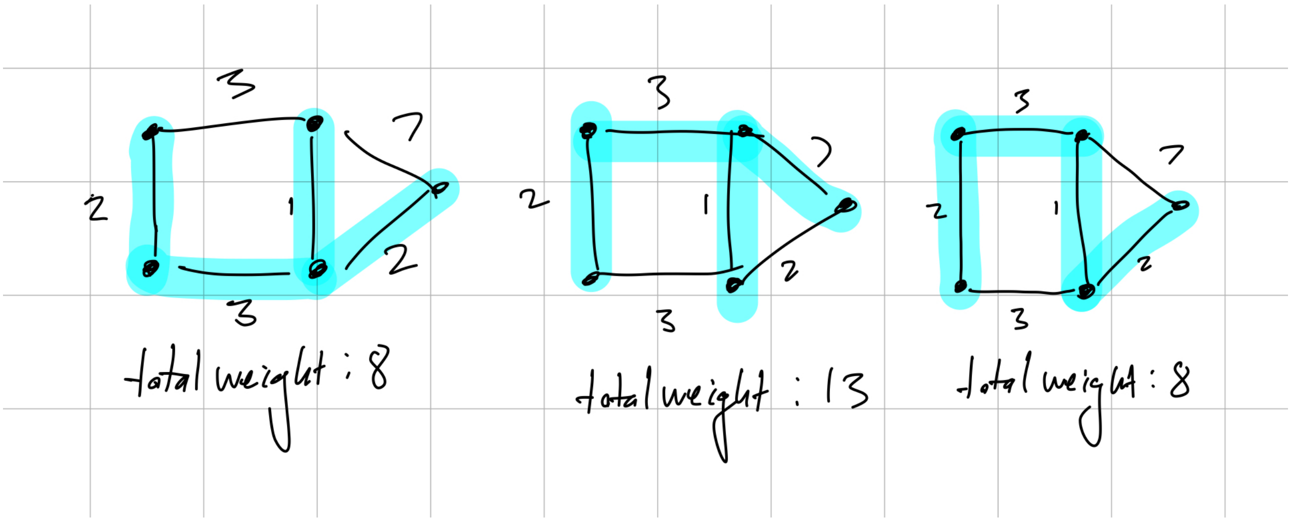
8 Minimum Spanning Trees (Kruskal, Prim)

- Minimum spanning trees
- Scheduling
- Divide and conquer

8.1 Minimum spanning tree

Given a weighted undirected graph, we want to find a spanning tree that minimizes the total weight of the edges in the tree.



Definition 8.1
A **cut** is a bipartition of the vertices ($V = A \cup B$, $A \cap B = \varnothing$).

A cut is **nontrivial** if both A and B are nonempty.

An edge **crosses** the cut if it has one end in A and one end in B.

Lemma 8.2
Assume all edge weights are distinct. Let (A, B) be a nontrivial cut, and let e be the minimum-weight edge crossing this cut.

Then any minimum spanning tree must contain e.

We'll prove this by an **exchange argument**.

Proof. Let T' be a spanning tree that does not contain $e = \{u, v\}$.
Since T' is spanning, there is a path in T' from u to v.
Since e crosses the cut, there must be some edge along the path that crosses the cut, call it e'.

Construct a new spanning tree T from T' by deleting e' and adding e.
This is a graph with the same vertex set and the same number of edges, so it is a spanning tree.

Since the weight of e is less than the weight of e', this change lowers the total edge weight. □

8.1.1 Kruskal's algorithm

We add the lowest edge that doesn't create a cycle to our spanning tree.

Theorem 8.3
Kruskal's algorithm outputs a minimum spanning tree

Proof. Suppose the algorithm adds the edge $e = \{u, v\}$ to the forest F.

Consider the cut induced by the component of u in F (one part of the partition is the component of the forest containing u, and the other part of the partition is every other node (including v)).

Clearly e crosses this cut, and by definition it is the minimum-weight edge with this property.

So by the lemma, any minimum spanning tree must include e.

It remains to show that the algorithm outputs a spanning tree. It clearly does not create a cycle. If the graph were not connected, the algorithm could always add some edge without creating a cycle, so the final putput must be a tree. □

8.1.2 Prim's algorithm

Choose a root, and repeatedly add the non-tree vertex with the lowest attachment cost.

Theorem 8.4
Prim's algorithm outputs a minimum spanning tree

Proof. Suppose the algorithm adds edge $e = \{u, v\}$ to forest F.
Consider the cost induced by the component of the root (from which the algorithm builds the tree).

Clearly e crosses this cut, and by definition it is the minimum weight edge with that property.

So by the lemma, every minimum spanning tree contains e.

Clearly outputs a spanning tree. (As long as the graph is not connected, there is always some edge that the algorithm can choose to include) □

8.1.3 Implementations

- Prim: keep a priority queue $O(m \log n)$
- Kruskal: union-find data structure $O(m \log n)$