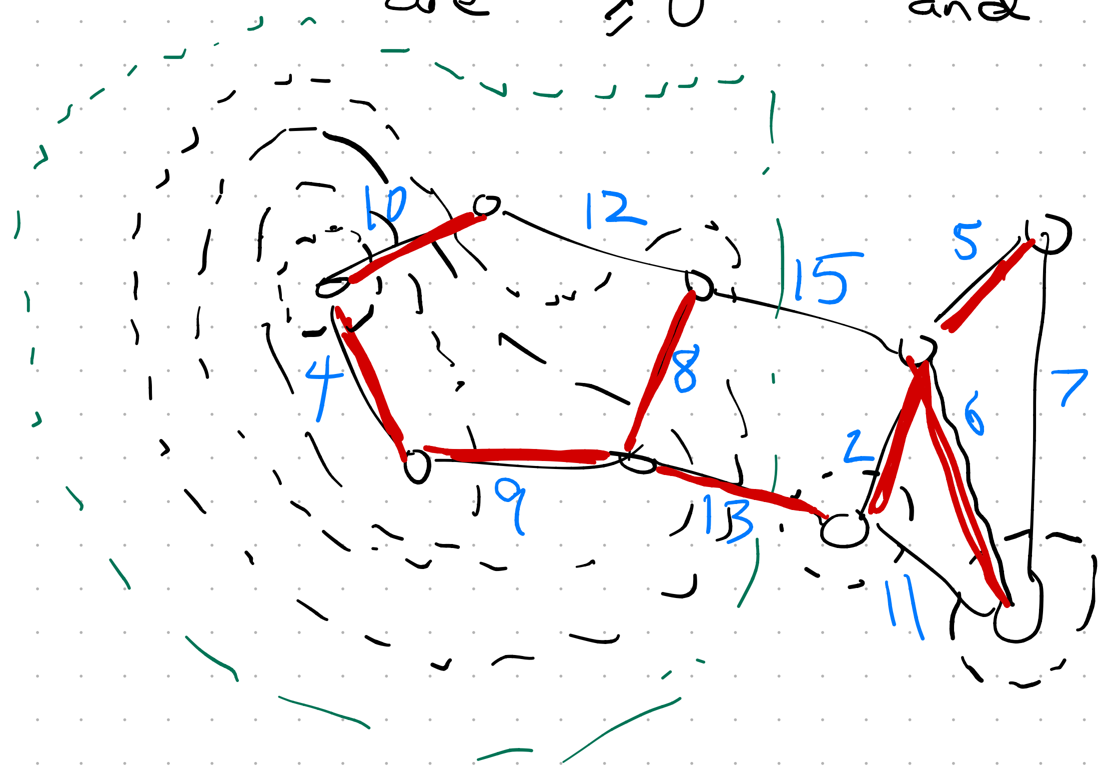
2 Feb — Algorithms for Min Spanning Tree

Reminder:

- G undirected connected graph
  $= (V, E, W)$ — V: vertices, E: edges, W: weights

- For this lecture assume all edge weights are ≥ 0 and distinct.



CUT LEMMA:

The min weight edge crossing any cut must be in the MST when all weights are distinct.

CYCLE LEMMA:

The max weight edge in any cycle must not be in the MST when all weights are distinct.

PRIM'S ALGORITHM:

Choose any vertex, v_1.

Initialize $T = (\{v_1\}, \emptyset)$ — $\{v_1\}$: one vertex, $\emptyset$: no edges

While T is not a spanning tree:

find the min weight edge from $V(T)$ to $V(G) \setminus V(T)$.

insert that edge into T.

// T gains one vertex and one edge

output T.

Proof of correctness: repeatedly apply Cut Lemma.

(Termination proof: $V(T)$ grows by one vertex each iteration, and cannot grow unboundedly.)

KRUSKAL'S ALGORITHM:

Sort edges by increasing weight: $e_1, e_2, \ldots, e_m$.
initialize $E(T) = \emptyset$, $V(T) = V$.
for $i = 1, 2, \ldots, m$:
insert e_i into T unless it creates a cycle.

<u>Correctness:</u> Every omitted edge is the max weight edge in some cycle.

But why does it output a spanning tree???

Loop invariant: at the end of the i^{th} loop iteration the graph

$$(V, E(T) \cup \{e_{i+1}, e_{i+2}, \ldots, e_m\})$$

is connected.

Induction step: every edge we deleted, did not disconnect the graph b/c there was already a path in T connecting its endpoints.

<u>Conclusion:</u> T is a spanning tree, and the <u>complement</u> of its edge set is contained in the complement of the MST's edge set. Since all spanning trees have the same number of edges, T and the MST must coincide.