COP 3503 10/22/26

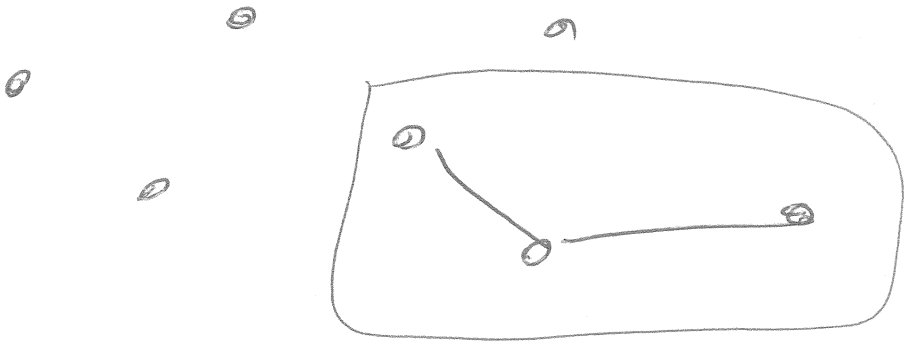
Minimum Spanning Tree Problem

Input: Weighted (non-neg edge weights) Undirected Graph

Output: Subset of edges with a minimal sum of costs (edge weights) s.t. the graph created in connected OR not possible

tree is a graph w/o cycles (always unique path from any vertex to any other vertex in a tree) – always has $|V|-1$ edges.

Spanning - includes all of the graph vertexes



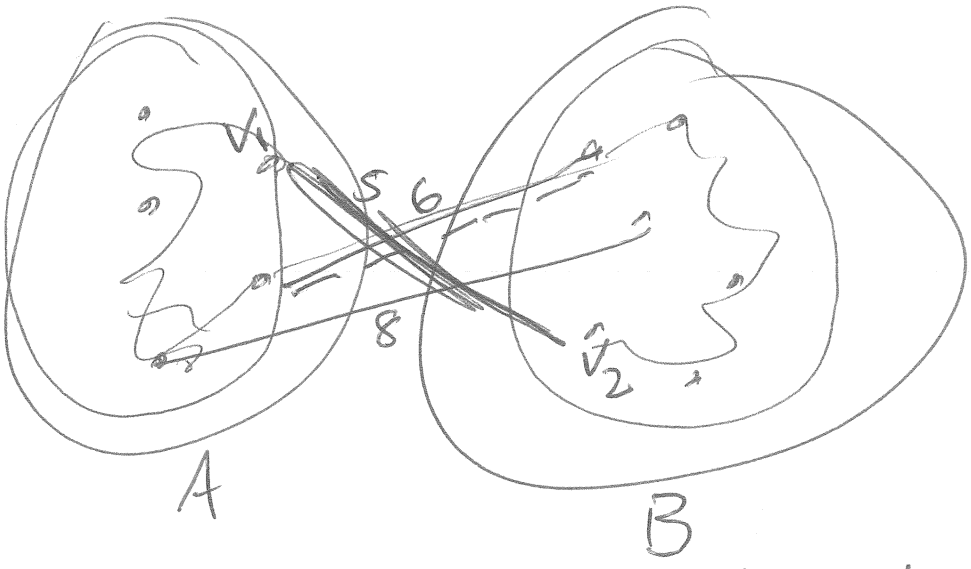
Minimum - Smallest sum of edge weight over all possible spanning trees.

2 Greedy Algs Solve Problem

(1) Prim's (uses PQ)

(2) Kruskal's (uses Disjoint Set)

Crucial Fact About MST

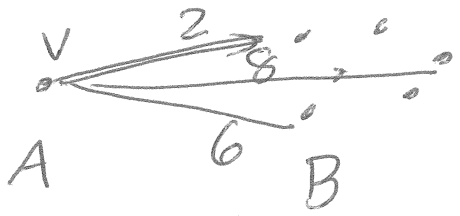


for any partition of vertices A, B $A \cup B = V$ $A \cap B = \emptyset$, the minimum weight edge $(u,v) \in E$ s.t. $u \in A \wedge v \in B$ is <u>SAFE</u> to add to an MST.

Let's say you claim that no MST has this min edge. Instead, you say my competing MST has a different "cross"-edge.

I add that min edge (5) to your graph. This creates a cycle! I can remove cycle by deleting your cross edge!

Prim's Alg Input: G and a starting vertex v.



Prim's we assume starting vertex is in set A.
All others in set B.

boolean array keeps track of set A.

used[v] = true

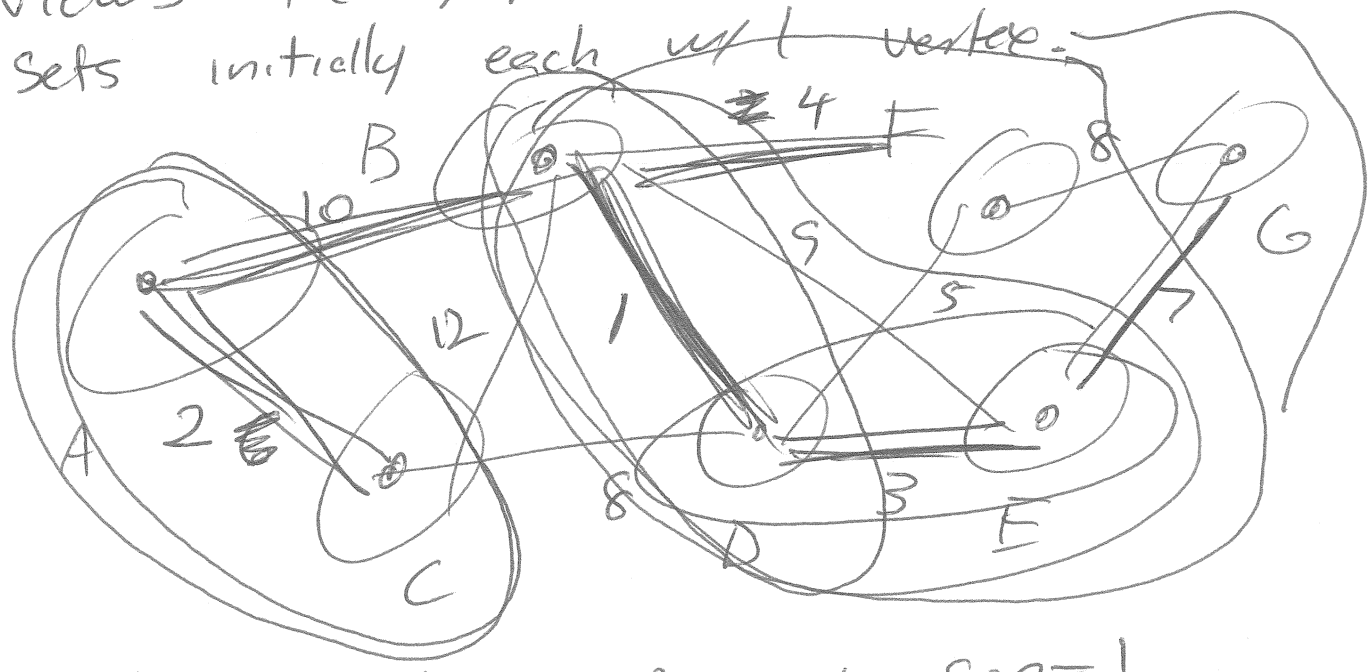
PQ of edges - add all edges that are incident ~~to v~~

while ~~(numEdges < |V|-1 && graph not a~~ ⟹ if(pq.size() == 0) break;

```
edge e = pq.poll();
if used[e.u] && used[e.v] =
    continue;
newV = used[e.u] ? e.v : e.u;
mst += e.w
used[newV] = true
numE++;
for (edge e: graph[newV])
    pq.offer(e);
```

Kruskal's

Views the graph as a partition of |V| sets initially each w/ 1 vertex.



list of ALL edges + SORT!

go through edges

ADD the edge if doing so doesn't cause a cycle!

Stop when you have a spanning tree.

Cycle Detection via Disjoint Set!

Red Blue Tree

$n = 1000$ run mst w/ $R = 1$, $B = 0$

ans = 15

$\Rightarrow$ all Span Trees use ≤ 985 blue

$n = 1000$ run mst w/ $B = 1$, $R = 0$

ans = 200

$\Rightarrow$ all Span Trees use ≥ 200 blue

