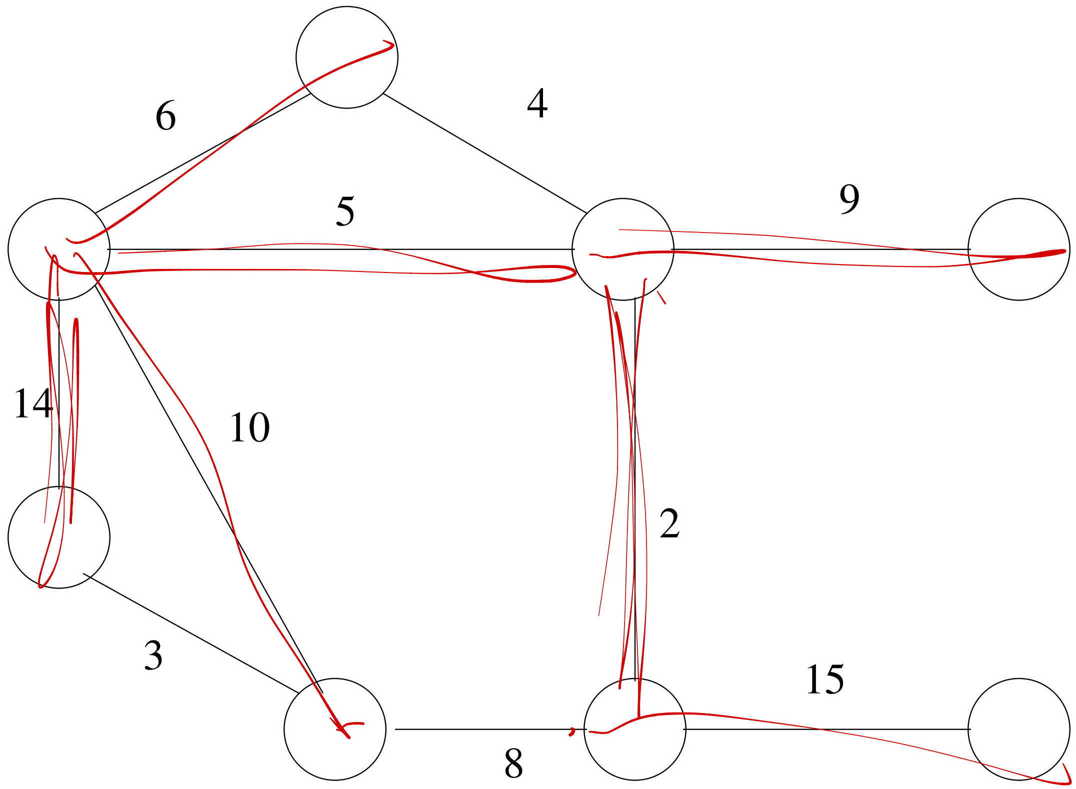
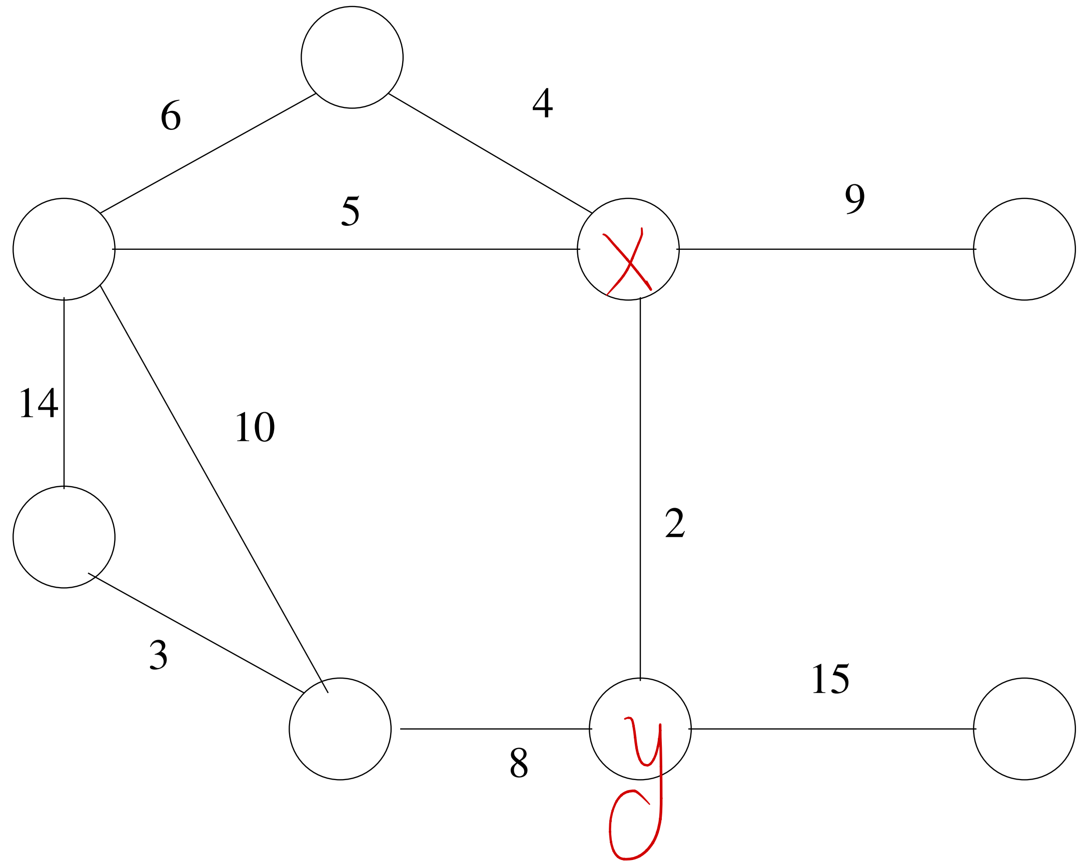
Minimum Spanning Trees

- $G = (V, E)$ **is an undirected graph with non-negative edge weights** $w : E \to Z^+$
- **We assume wlog that edge weights are distinct**
- **A spanning tree is a tree with** $V - 1$ **edges, i.e. a tree that connects all the vertices.**
- **The total cost (weight) of a spanning tree T is defined as** $w(T) = \Sigma_{e \in T} w(e)$
- **A minimum spanning tree is a tree of minimum total weight.**



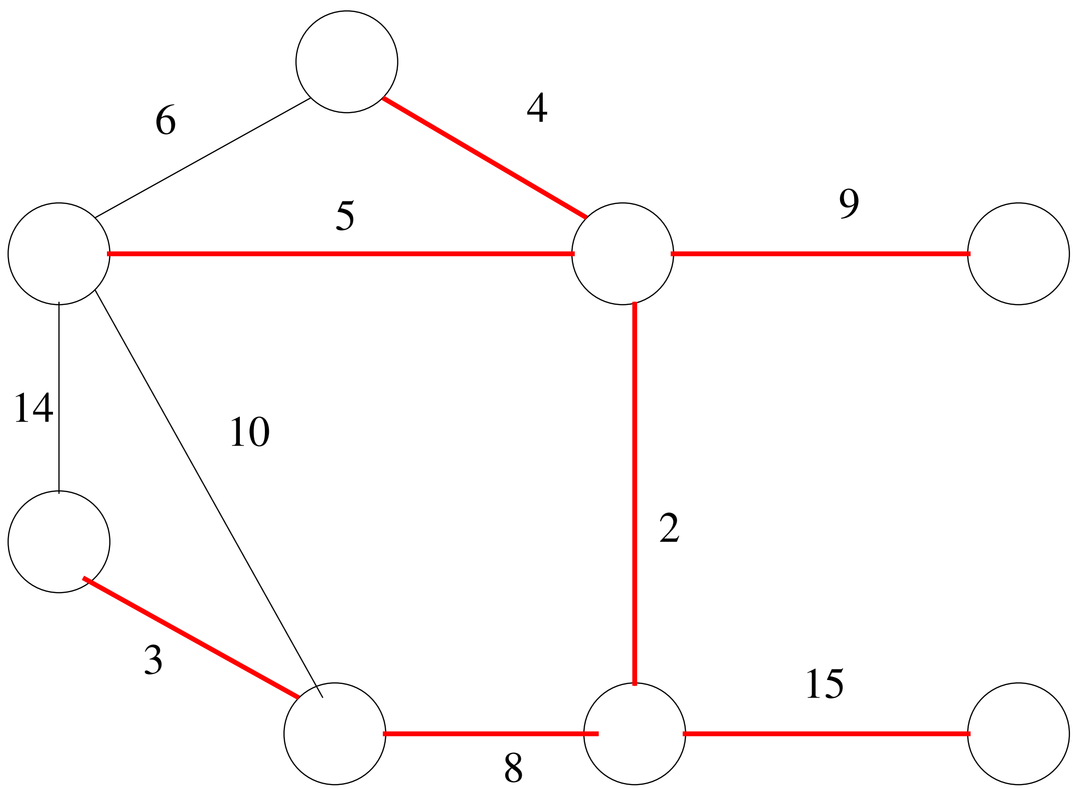
6
4
5
9
14
10
2
3
15
8



6
4
5
9
x
14
10
2
3
15
8
y

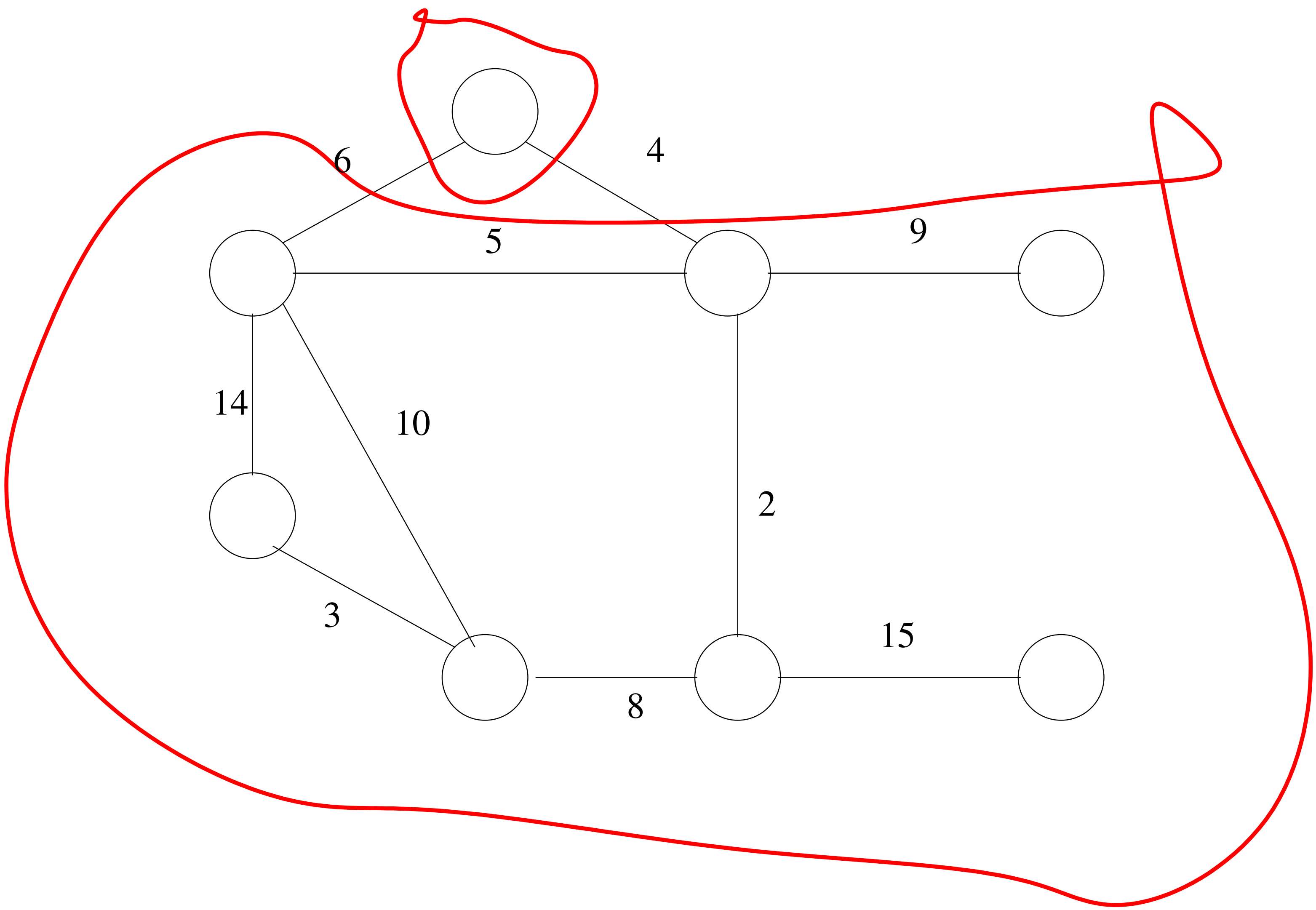
Minimum Spanning Trees

- $G = (V, E)$ **is an undirected graph with non-negative edge weights** $w : E \to Z^+$
- **We assume wlog that edge weights are distinct**
- **A spanning tree is a tree with** $V - 1$ **edges, i.e. a tree that connects all the vertices.**
- **The total cost (weight) of a spanning tree T is defined as** $\Sigma e \in Tw(e)$
- **A minimum spanning tree is a tree of minimum total weight.**



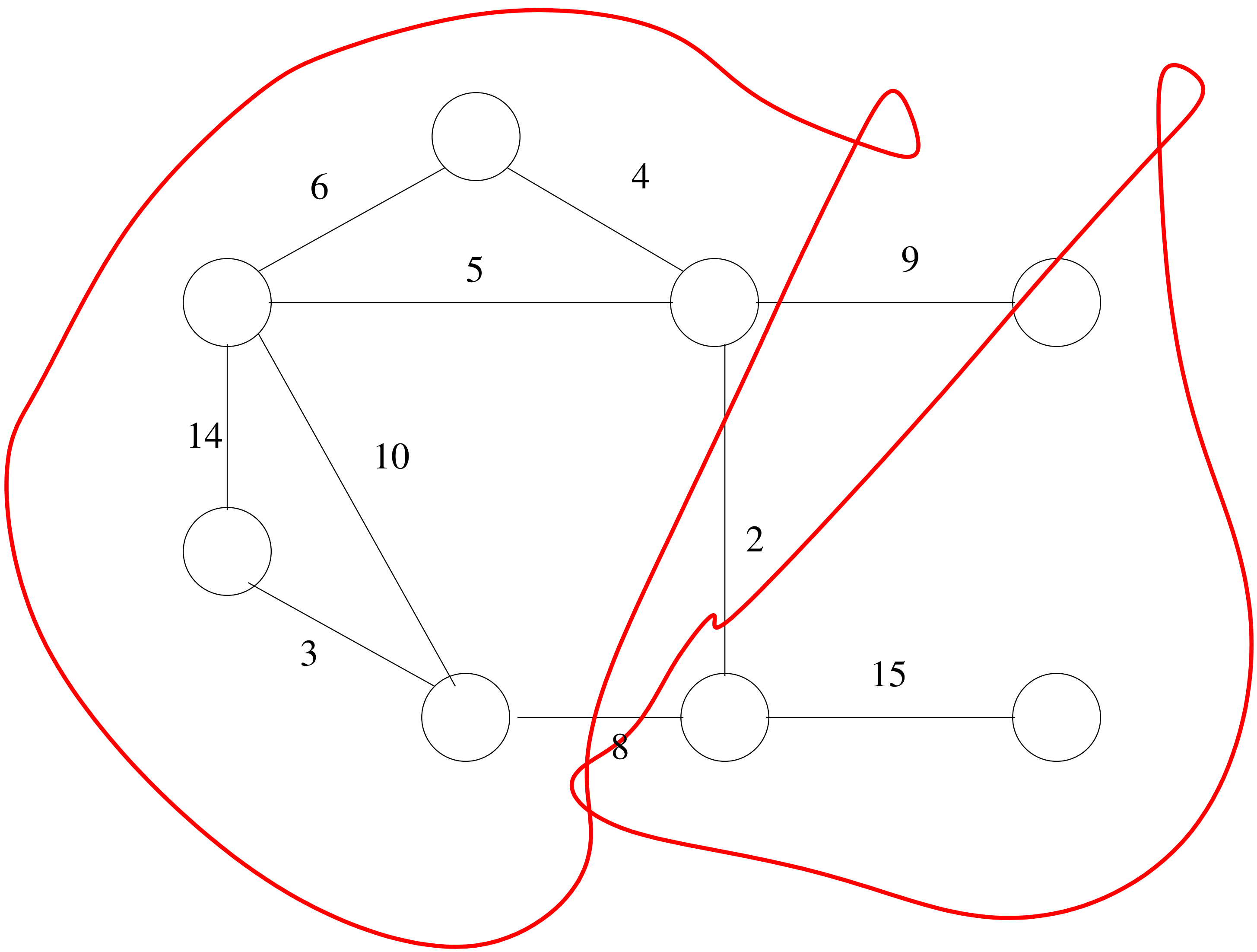
Cuts

- A **cut** in a graph is a partition of the vertices into two sets S and T.
- An edge (u, v) with $u \in S$ and $v \in T$ is said to **cross the cut**.



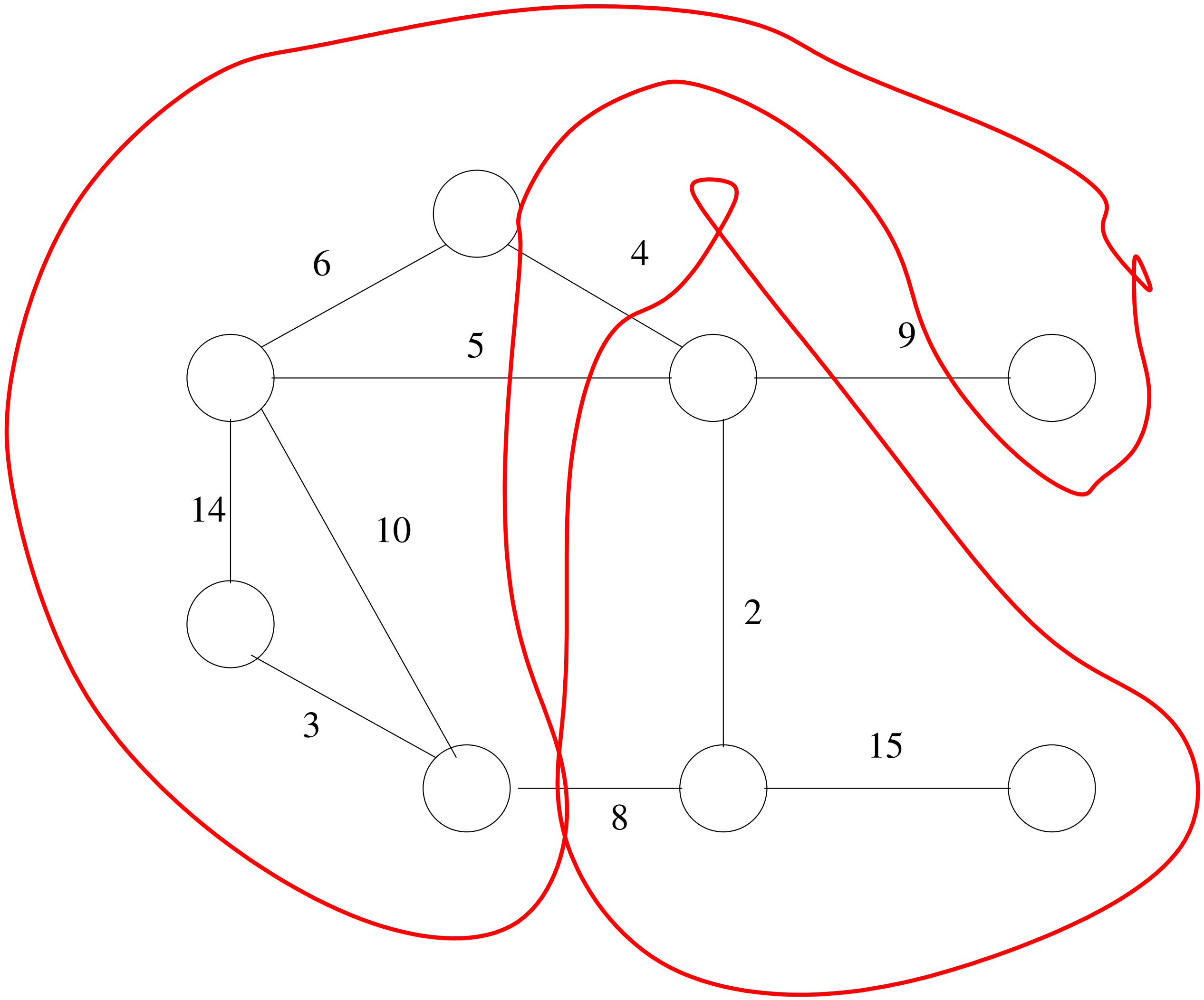
Cuts

- A **cut** in a graph is a partition of the vertices into two sets S and T.
- An edge (u, v) with $u \in S$ and $v \in T$ is said to **cross the cut**.



Cuts

- **A cut in a graph is a partition of the vertices into two sets S and T.**
- **An edge (u, v) with $u \in S$ and $v \in T$ is said to cross the cut.**



Greedy Property

Recall that we assume all edges weights are unique.

Greedy Property: The minimum weight edge crossing a cut is in the minimum spanning tree.

Proof Idea: Assume not, then remove an edge crossing the cut and replace it with the minimum weight edge.

Restatement **Lemma:** Let $G = (V, E)$ be an undirected graph with edge weights w. Let $A \subseteq E$ be a set of edges that are part of a minimum spanning tree. Let (S, T) be a cut with no edges from A crossing it. Then the minimum weight edge crossing (S, T) can be added to A.

Algorithm Idea: Repeatedly choose an edge according to the Lemma, add to MST.

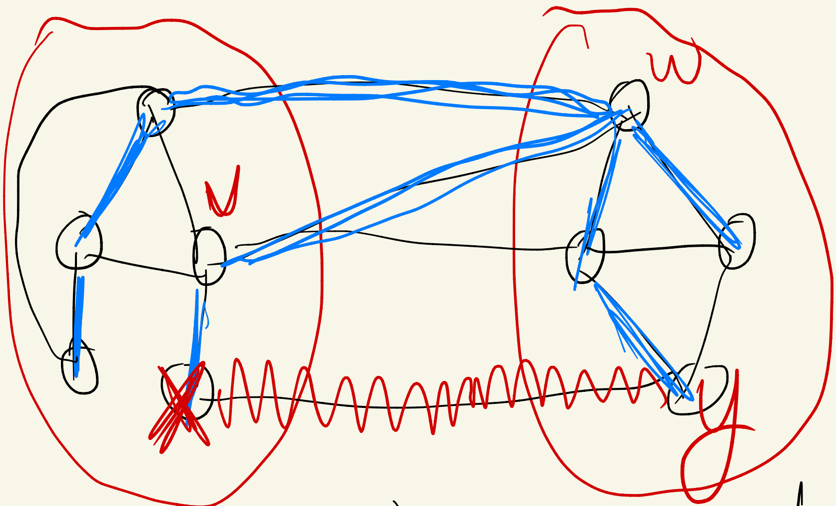
Challenge: Finding the edge to add.

Two standard algorithms:

- Kruskal - consider the edges in increasing order of weight
- Prim - start at one vertex and grow the tree.

Proof

Assume not. Pick a cut S, T.
Let (x, y) be the min wt. edge crossing the cut, and assume there's a mst T that does not contain (x, y)



T ∪ (x, y) contains a cycle.
At least one edge on that cycle crosses the cut.

Call that edge (v,w) and consider

$T' =$

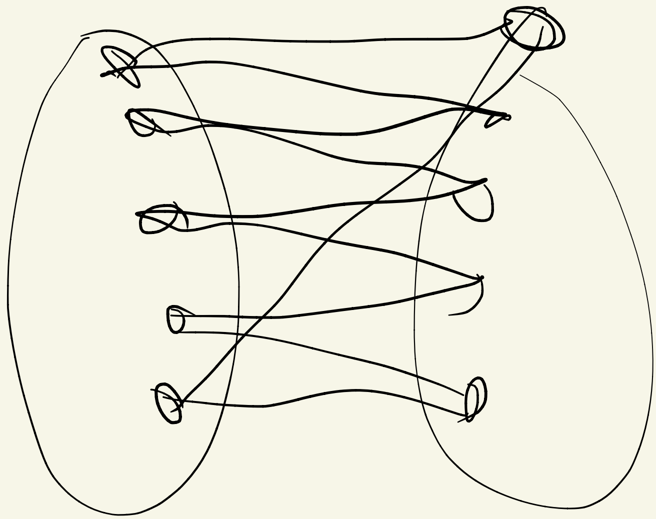
$T - (v,w) \cup (x,y)$.

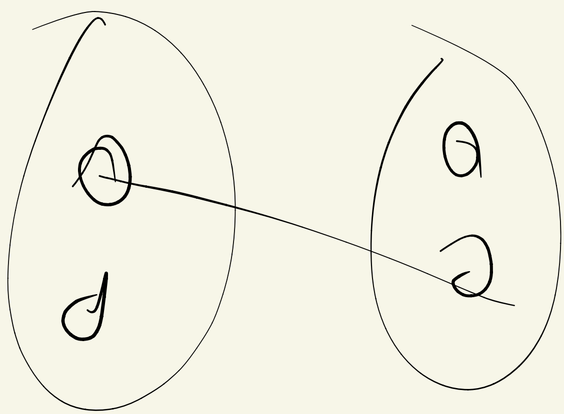
T' is a spanning tree of cost $c(T')$,

$c(T) - c(v,w) + c(x,y)$

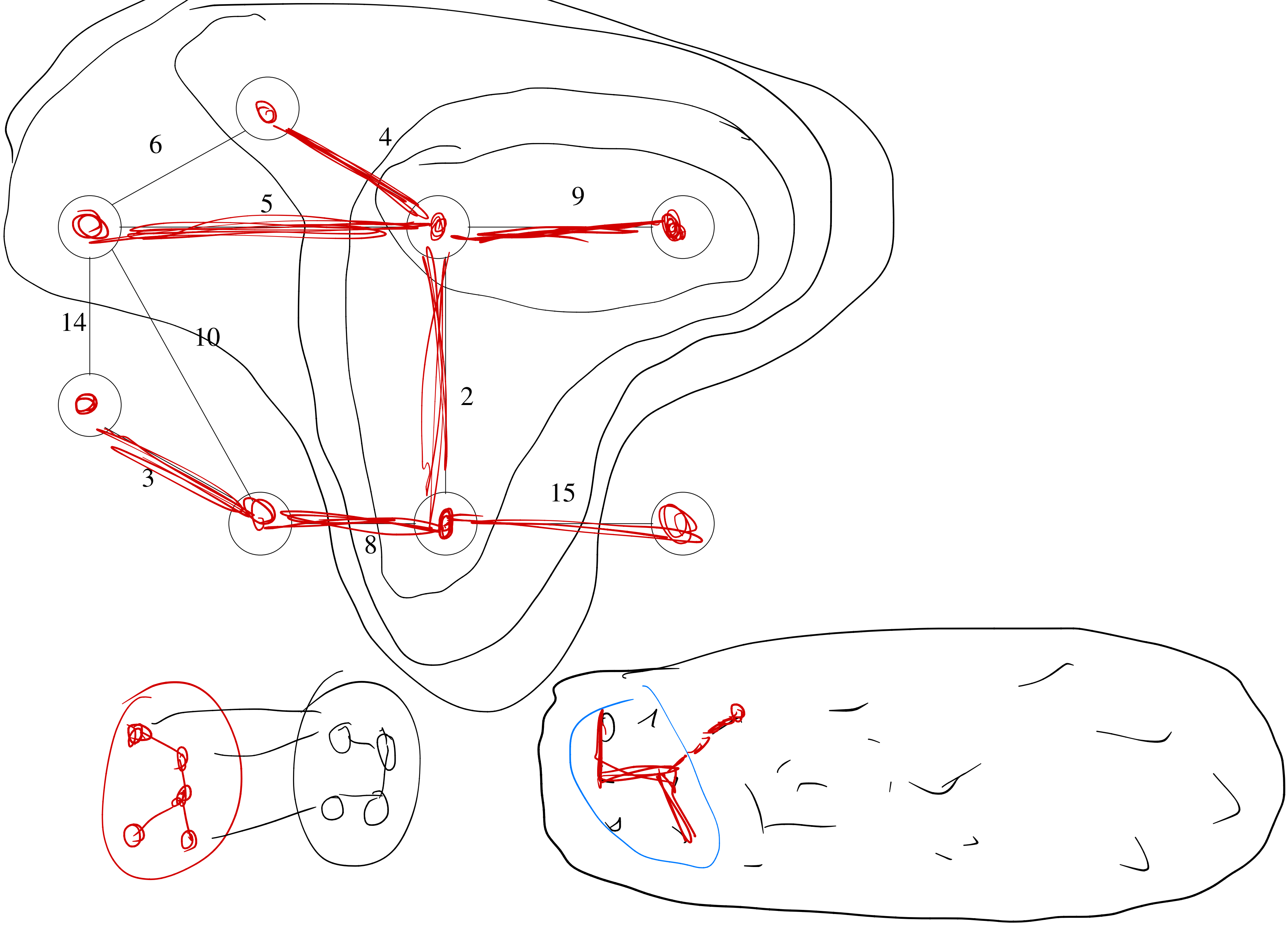
$< c(T)$.

$\therefore$ Contradicts T being the MST. $\boxtimes$



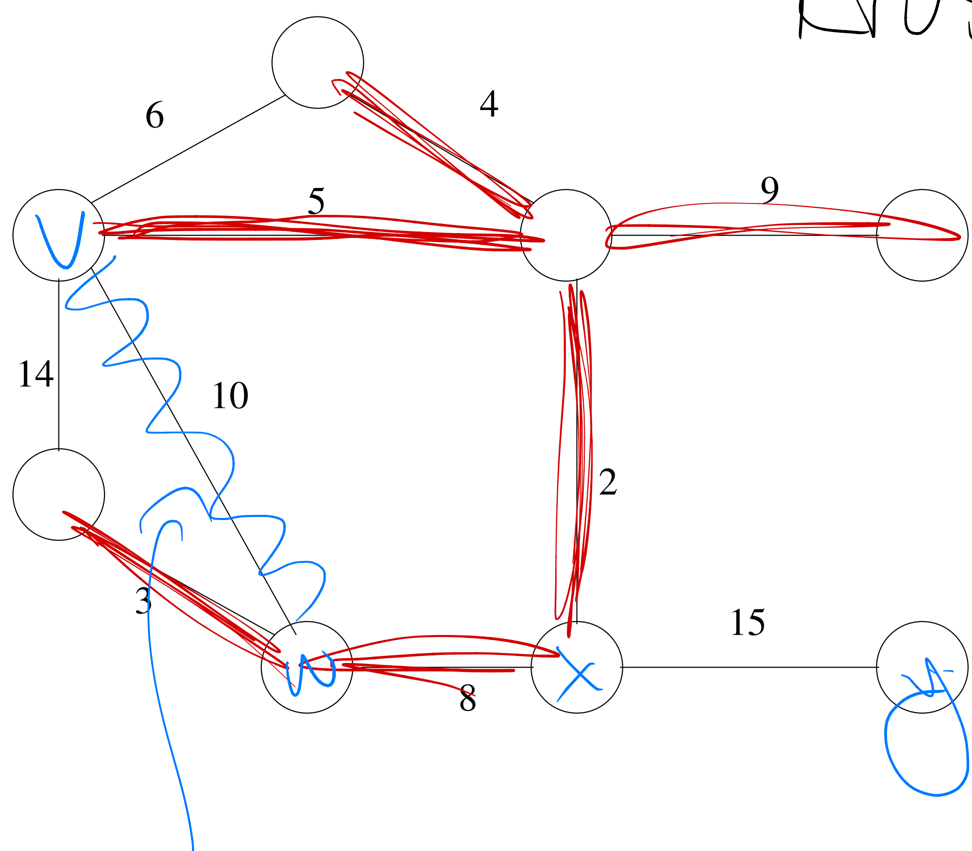


Example: Run both algorithms



Example: Run both algorithms

Kruskal



Are (v,w) already connected in the ~~spanning~~ tree so far

Kruskal's Algorithm: detailed implementation

Idea: Consider edges in increasing order.

Need: a data structre to maintain the sets of vertices in each component of the current forrest

- MAKE-SET(v) puts v in a set by itself
- FIND-SET(v) returns the name of v's set
- UNION(u, v) combines the sets that u and v are in

MST-Kruskal(G, w)

$A \leftarrow \emptyset$
for each vertex $v \in V[G]$
    do MAKE-SET(v)
sort the edges of E **into nondecreasing order by weight** w
for each edge $(u, v) \in E$, **taken in nondecreasing order by weight**
    do if FIND-SET(u) $\neq$ FIND-SET(v)
        then $A \leftarrow A \cup \{(u, v)\}$
            UNION(u, v)
return A

Kruskal Running Time

- V MAKE-SET
- V UNION
- $2E$ FIND-SET

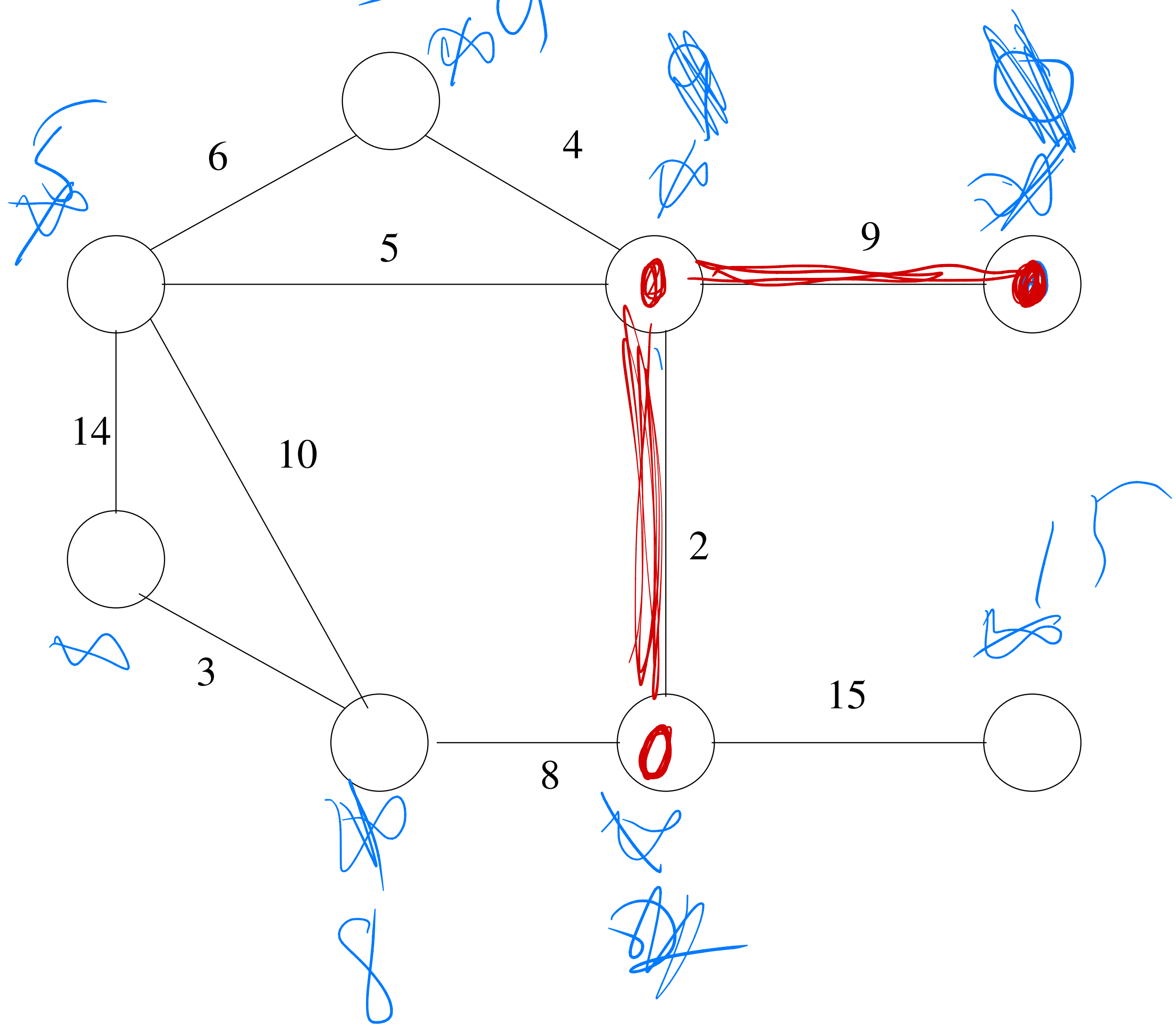
Analysis **After sorting, Kruskal takes** $E \log^* V$ **time (actually slightly better inverse Ackerman time).**

$$\lg^* V = \min\{i : \lg^{(i)} V \le 2\}$$

$$\lg^* 2^{2^{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}} = 5$$

Example

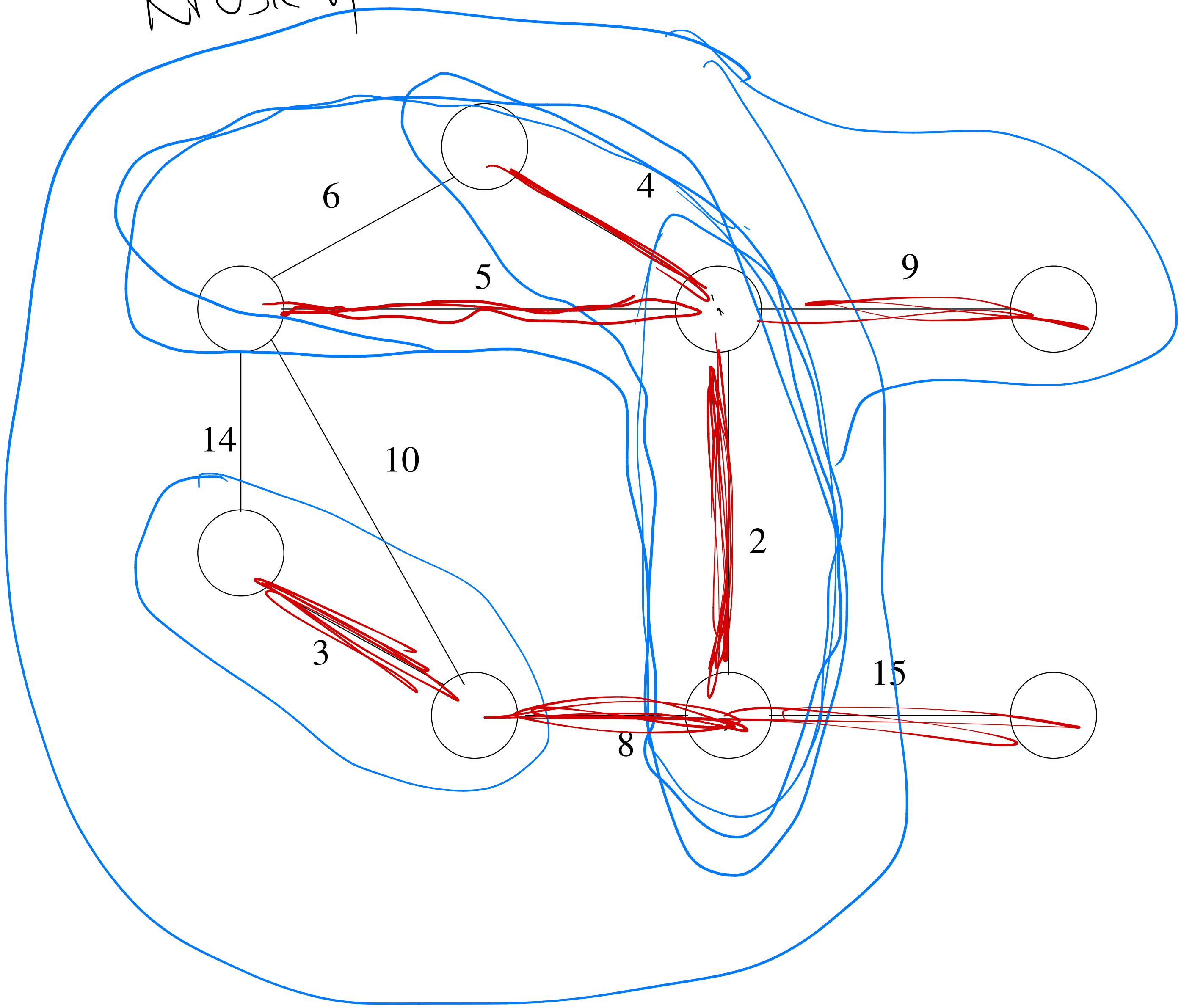
Prim



min wt edge incident crossing the cut

Kruskal

Example



Prim's Algorithm

Idea: Grow the MST from one node going out

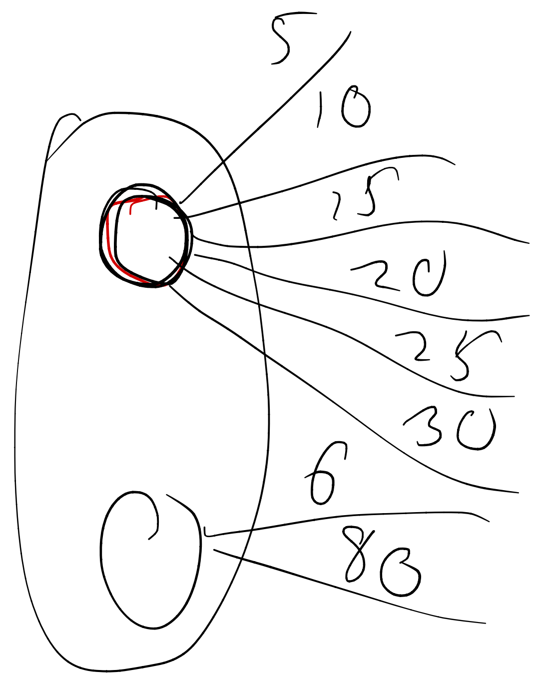
Need: a data structure to maintain the edgdes crossing the cut, and choose the minimum. We will maintain, for each vertex, the minimum weight incident edge crossing the cut

- INSERT(v) puts v in the structure
- EXTRACT-MIN(v) finds and returns the node with minimum key value
- DECREASE-KEY(v, w) updates (decreases) the key of v

MST-Prim(G, w, r)

for each $u \in V[G]$
    do $key[u] \leftarrow \infty$
        $\pi[u] \leftarrow \mathrm{NIL}$
$key[r] \leftarrow 0$
$Q \leftarrow V[G]$
while $Q \neq \emptyset$
    do $u \leftarrow$ EXTRACT-MIN(Q)
        for each $v \in Adj[u]$
            do if $v \in Q$ **and** $w(u, v) < key[v]$
                then $\pi[v] \leftarrow u$
                    $key[v] \leftarrow w(u, v)$

Decrease Key



Analysis

Op	Heap	Fibonacci Heap (amortized)
V Insert	$\lg V$	$\lg V$
V Extract-Main	$\lg V$	$\lg V$
E Decrease-Key	$\lg V$	1
Total	$O(E \lg V)$	$O(E + V \lg V)$

$V \approx 10^6$

$E \approx 10^9$

Example