Minimum Spanning Trees

Minimum Spanning Tree Problem

- Input : G(V, E)

$$c : E \to \mathbb{N}$$

- Output : $T \subseteq E$ such that G(V, T) is connected and $\sum_{e \in T} c(e)$ is minimised.

Minimum Spanning Tree Problem

- Input : G(V, E)

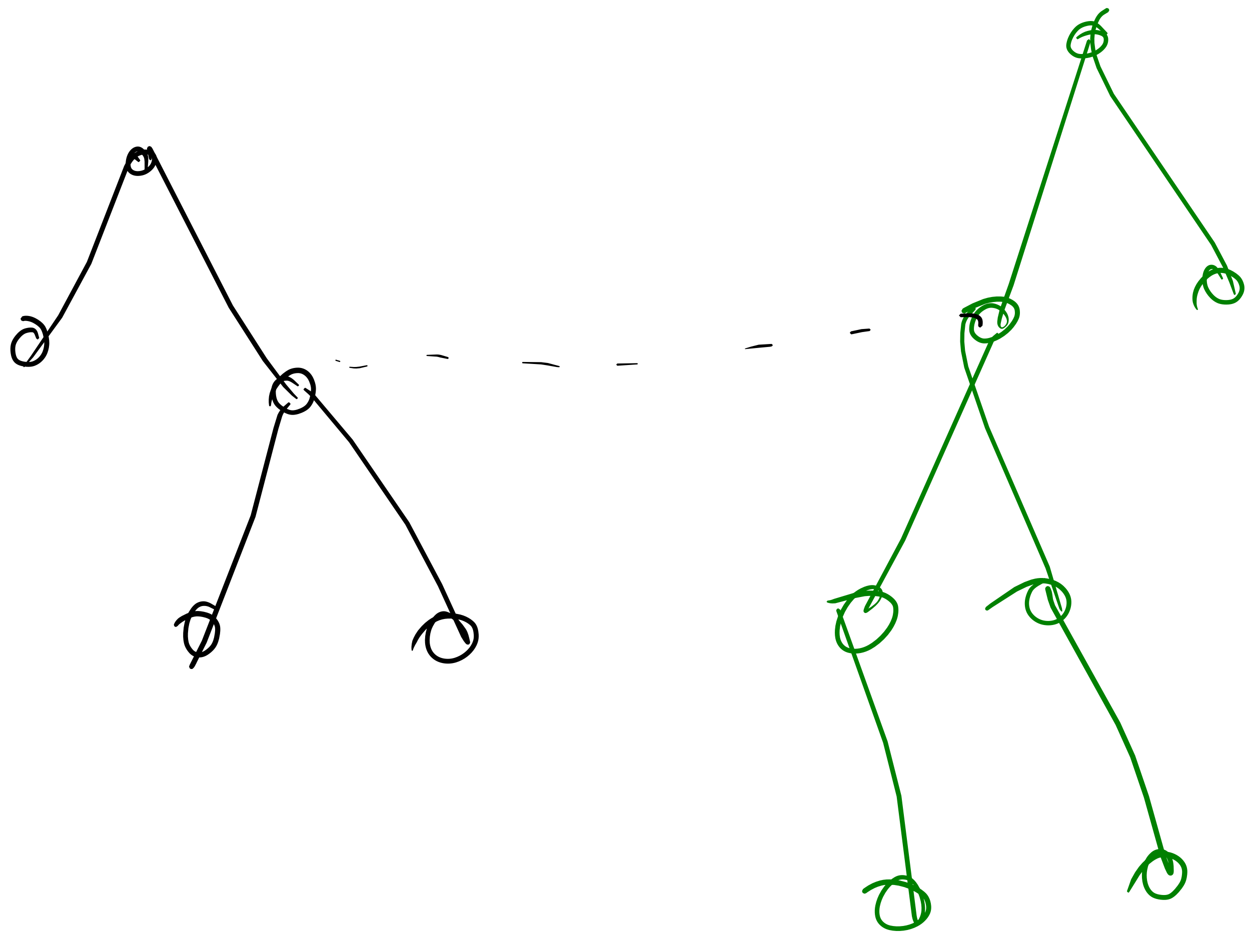
  $$c : E \to \mathbb{N}$$

- Output : $T \subseteq E$ such that G(V, T) is connected and $\sum_{e \in T} c(e)$ is minimised.

Can be solved using greedy strategy

Kruskal's Algorithm

- Insert edges in order of increasing cost so that no cycles are formed.



Prim's Algorithm

- Start with a root node s and try to greedily grow a tree from s outward.

- At each step, add the node that can be attached as cheaply as possible to the partial tree we already have.

Kruskal's Algorithm

A = $\emptyset$

Add each vertex v to a separate component of A – O(n)

Sort the edges of E by weight – m log m

For each edge (u,v) in order

if u and v are not in the same component – O(m) times

A = $A \cup \{u, v\}$ – O(n) times

Kruskal's Algorithm

Implementation and Running Time :

- sort the edges

- checking whether two elements are in the same component and merge (O(log n) time using Union Find data structure)

$O(n + m \log m + m \log n) = m \log n + n$

Prim's Algorithm

```
For each u ∈ V
        key[u] = ∞
        π[u] = NIL
key[r] = 0
Q = V
while Q ≠ ∅
        u = ExtractMin(Q), Add(u, π(u))
        For each v ∈ Adj[u]
                if v ∈ Q and w(u,v) < key[v]
                    π[v] = u
                    key[v] = w(u,v)
```

$O(n)$

Prim's

~~Kruskal's~~ Algorithm

Implementation and Running Time :

- Find the minimum element from Q (n times)

- Update the value of key (m times)

Prim's ~~Kruskal's~~ Algorithm

Implementation and Running Time :Using binary heap

- Find the minimum element from Q (n times) - n log n

- Update the value of key (m times) - m log n

  - O((m+n) log n)

~~Kruskal's~~ Prim's Algorithm

Implementation and Running Time :Using Fibonacci heap

- Find the minimum element from Q (n times) - n log n

- Update the value of key (m times) - m

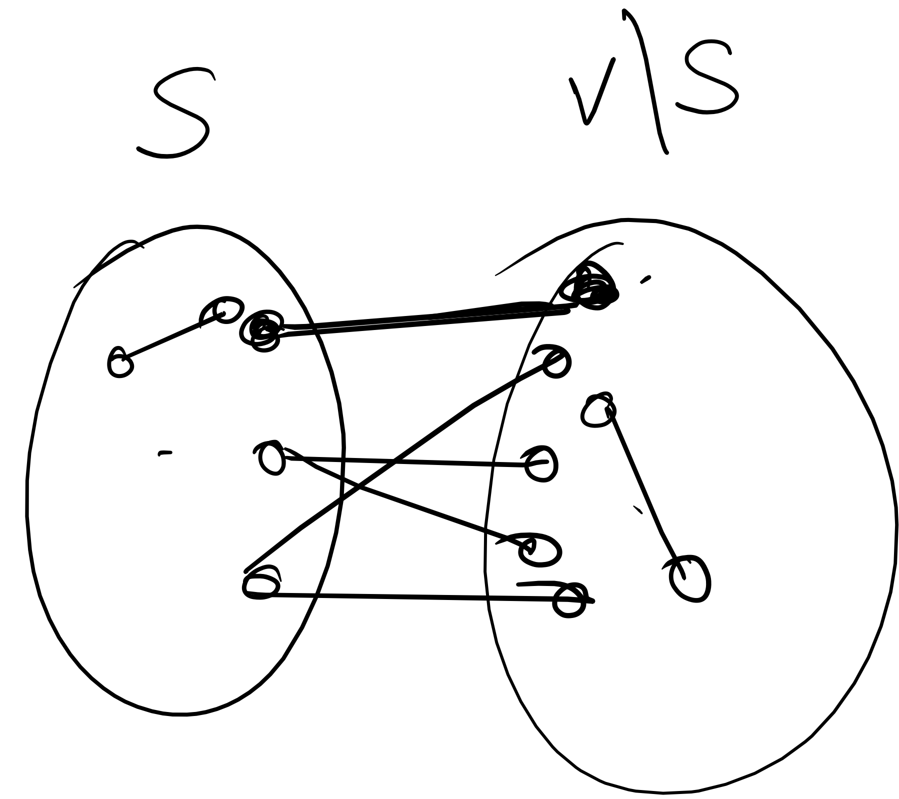
  - O(m+n log n) — fib heaps

$O((m+n) \log n)$ — Binary heaps.

Proof of correctness

- some edges are always safe to be added to a MST

Proof of correctness



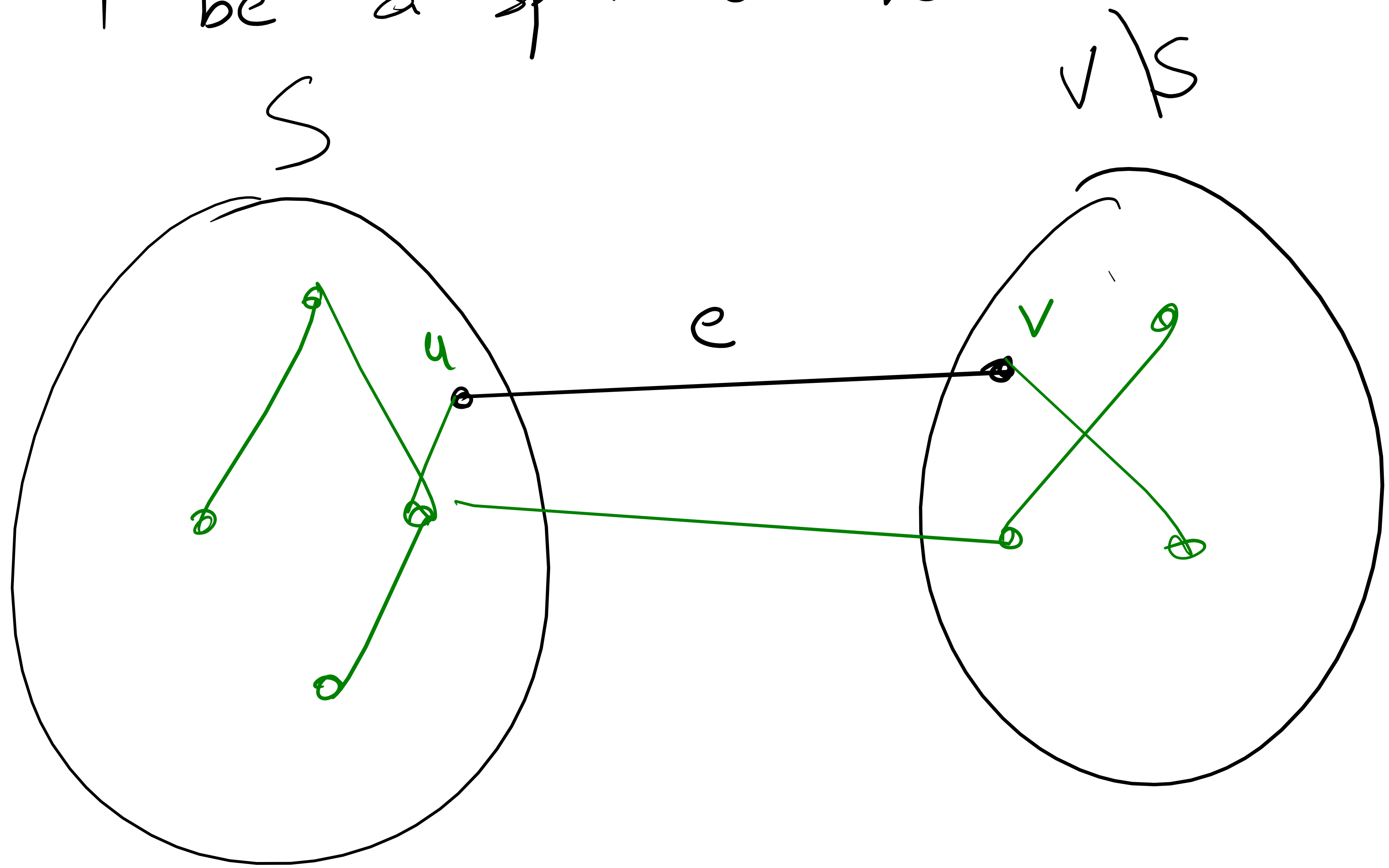
Cut Property

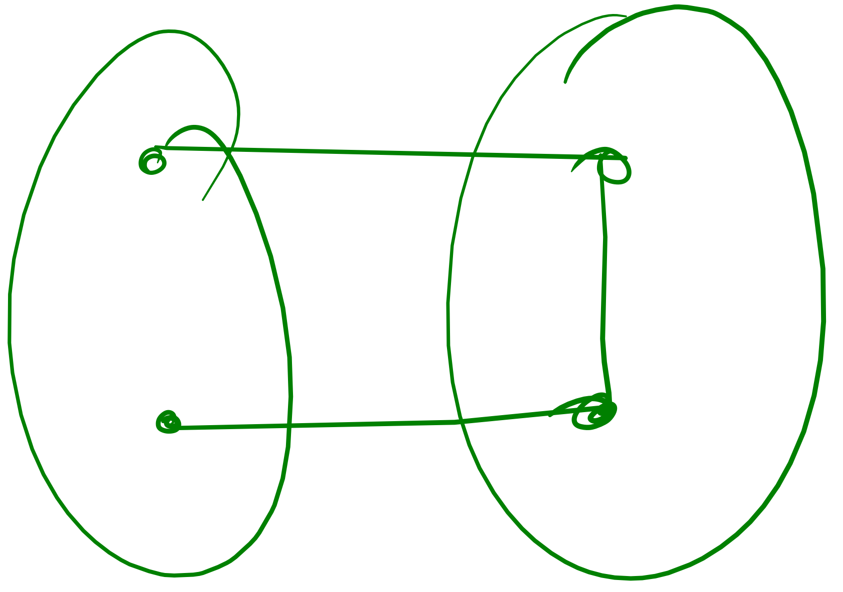
- Assume all edge costs are distinct
- Let $S \subset V, S \neq \emptyset$
- Let e be the minimum cost edge with one end in S and the other edge in V-S
- Every MST contains the edge e

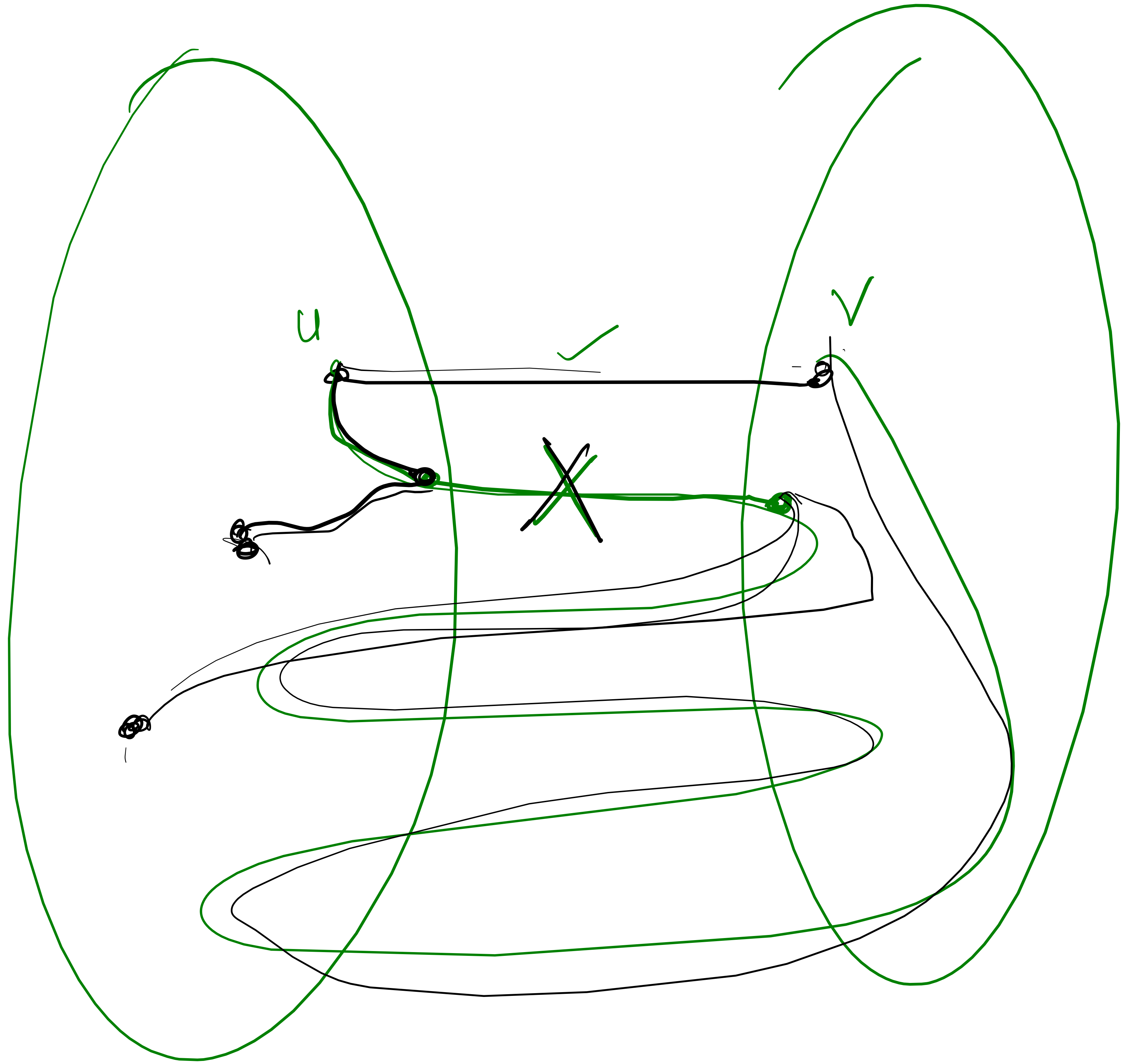
$(S, V \setminus S)$

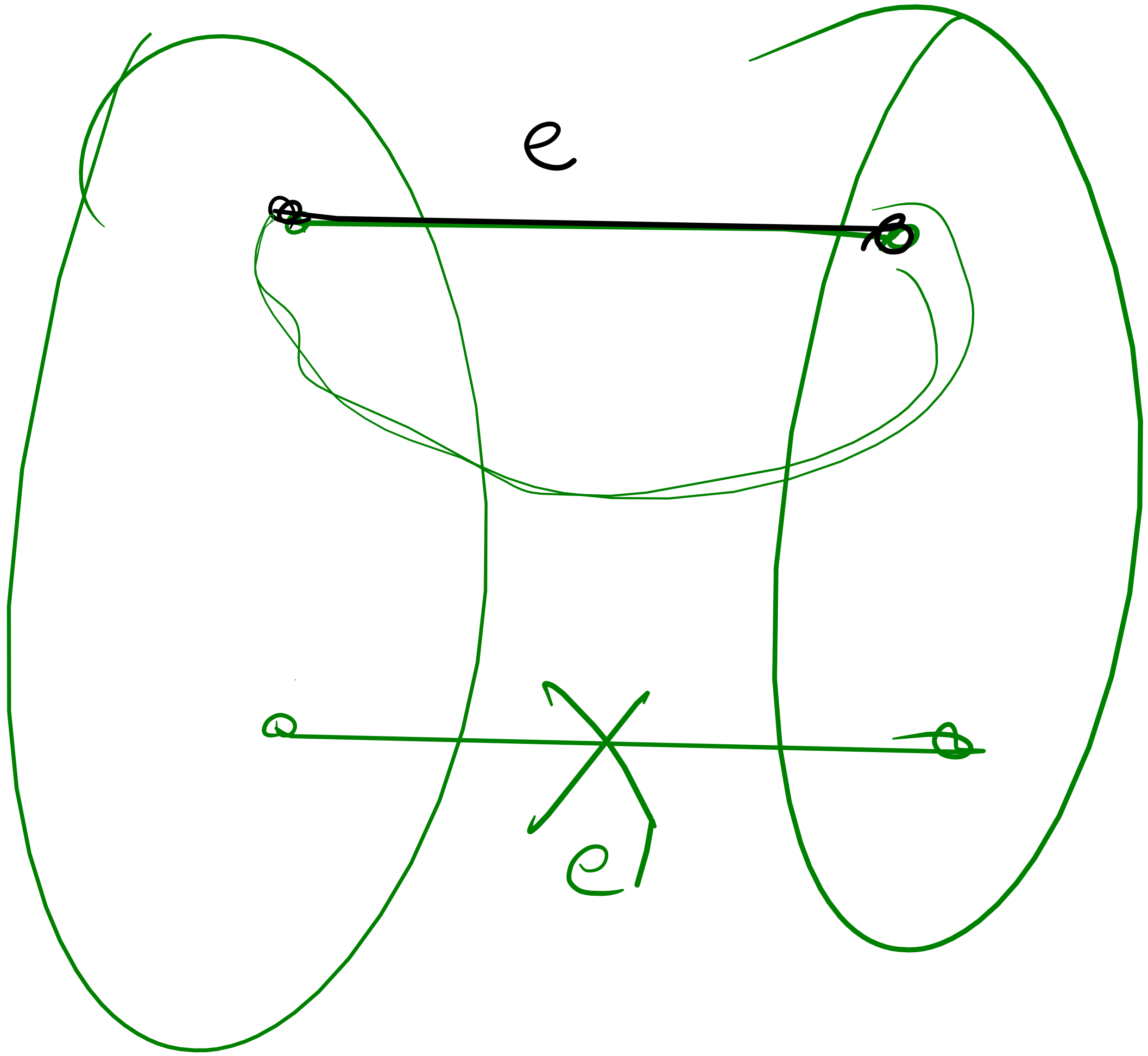
Let $e=(u,v)$ be the ~~mid~~ min edge across the cut.

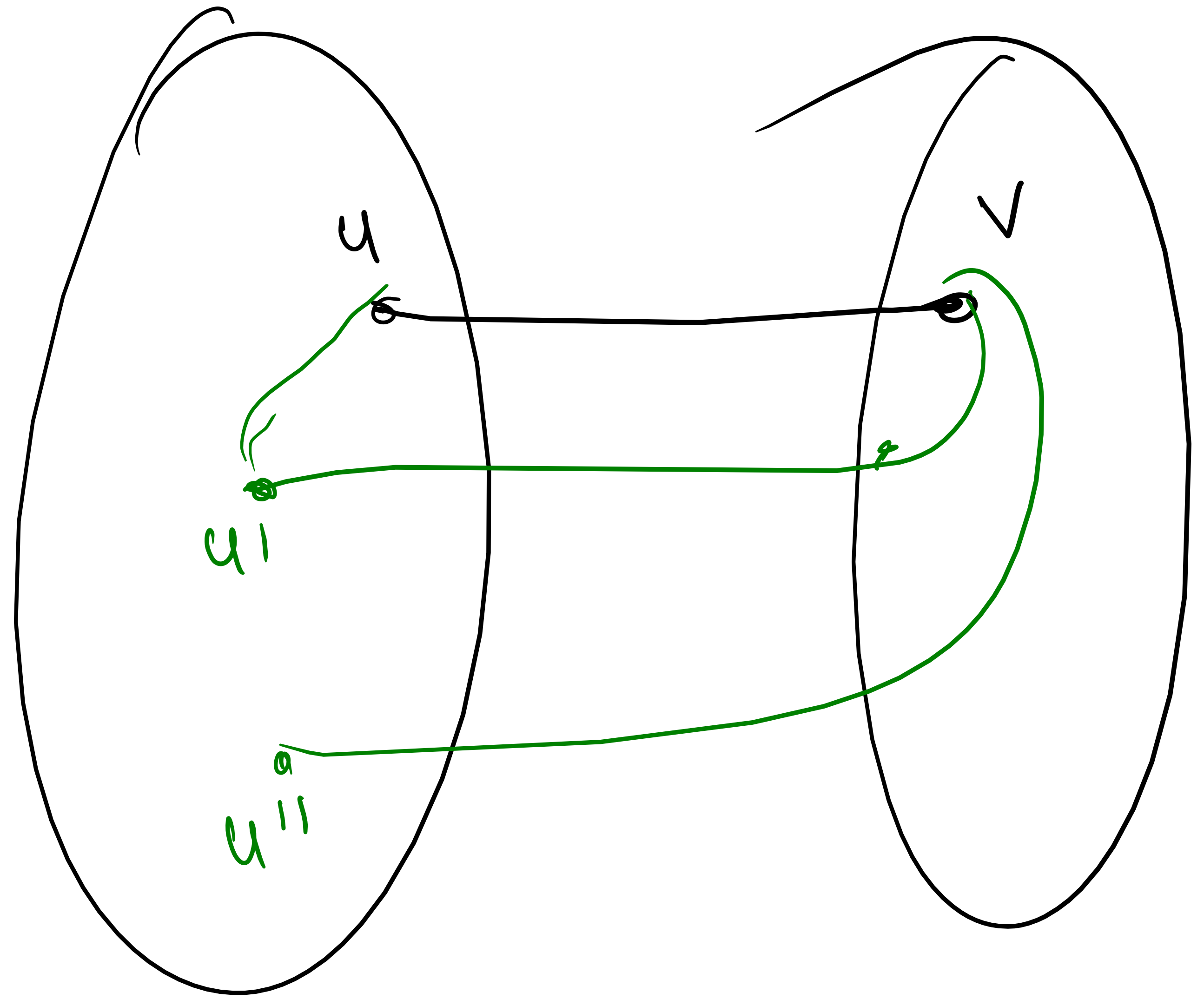
Let T be a sp. tree that does not contain e.






u
v


e
e


u
v
u'
u''

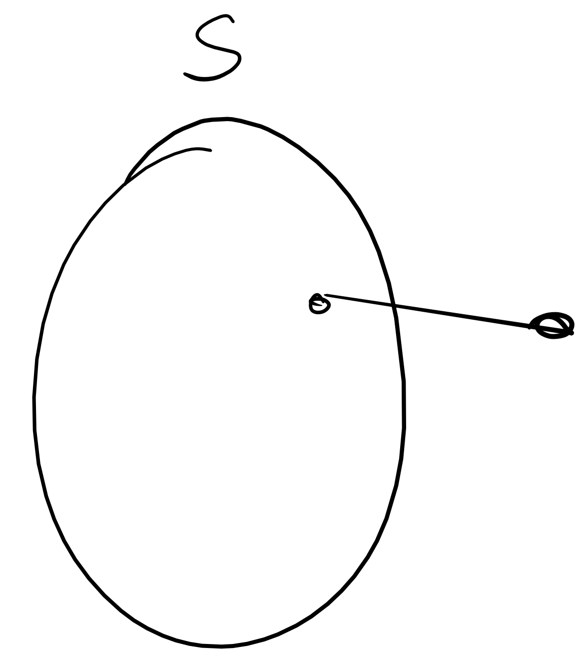
Optimality of Prim's.

It is enough to show that an sub. edge added by algo is safe.

Let (u,v) be an edge added. $u \in$ already computed partial solution, S.

(u,v) is the min-cut edge across $(S, V\setminus S)$.

$\Rightarrow (u,v)$ is safe.



Optimality of Kruskal's.

(u,v) — edge added.

S : vertices that are reachable from u in the partial sol.

$\Rightarrow$ (u,v) is the min edge across $(S, V\setminus S)$.