Greedy Algorithms

The Student's Problem:

You have n HW assignments, $a_1, \dots, a_n$ with deadlines $d_1, \dots, d_n$.

Each assignment takes 1 hour.

Goal: find a schedule that maximizes # of submitted assignments (on time).

For example:

a_1	a_2	a_3	a_4	a_5	a_6
1	1	2	2	4	5

a_1 a_3 a_5 a_6.
a_2

strategy: At any point in time, pick the assignment with closest deadline that has not expired.

Claim: This strategy gives an optimal solution

Proof: Suppose not. Let S be an optimal solution

G be the greedy solution.

$$S = (a_{i_1}, a_{i_2}, \ldots, a_{i_t})$$
$$G = (a_{j_1}, a_{j_2}, \ldots, a_{j_{t'}})$$

we Assumed $S \neq G$.

There's a first place k where S & G differs.

$i_1 = j_1, \; i_2 = j_2, \; \ldots, \; i_{k-1} = j_{k-1}$

Can we modify S to S' s.t. $i_k = j_k$

Case 1: a_{j_k} doesn't appear in S. a_{j_k} did not expired at time k. $\Rightarrow$ replace a_{i_k} with a_{j_k}. S'.

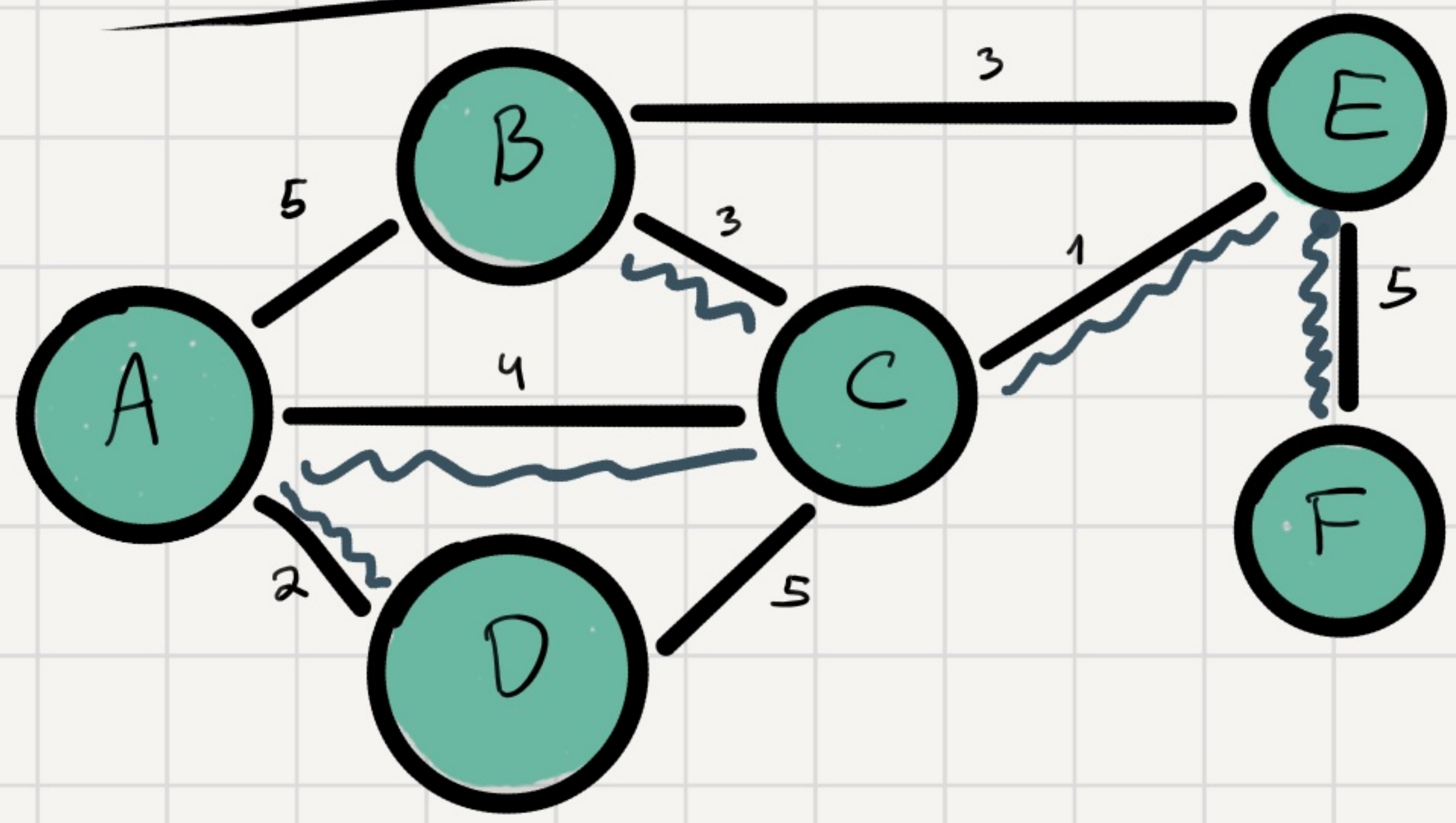
Case 2: $\exists l > k$ $\quad a_{i_l} = a_{j_k}$ $\quad$ swap(a_{i_k}, a_{i_l}).

$\Rightarrow S'$ that is optimal.

To be more formal S is an optimal solution that maximizes the length of longest prefix agreeing with G.

$\Rightarrow S'$ that agrees on a longer prefix.

Minimum Spanning Trees



Input: An undirected graph $G=(V,E)$

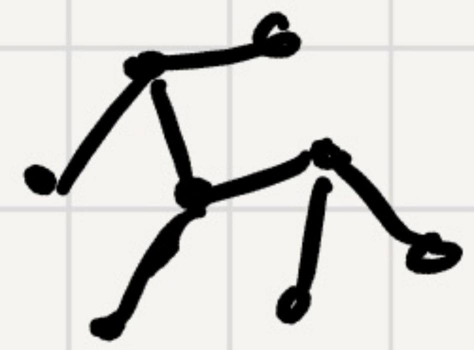
Weights $w: E \to \mathbb{N}$.

Output: A tree T, $V(T)=V(G)$, $E(T) \subseteq E(G)$ that minimizes

$$w(T) = \sum_{e \in E(T)} w(e).$$

and touches all nodes in the graph

Def'n: A tree is a connected graph w. no cycles.

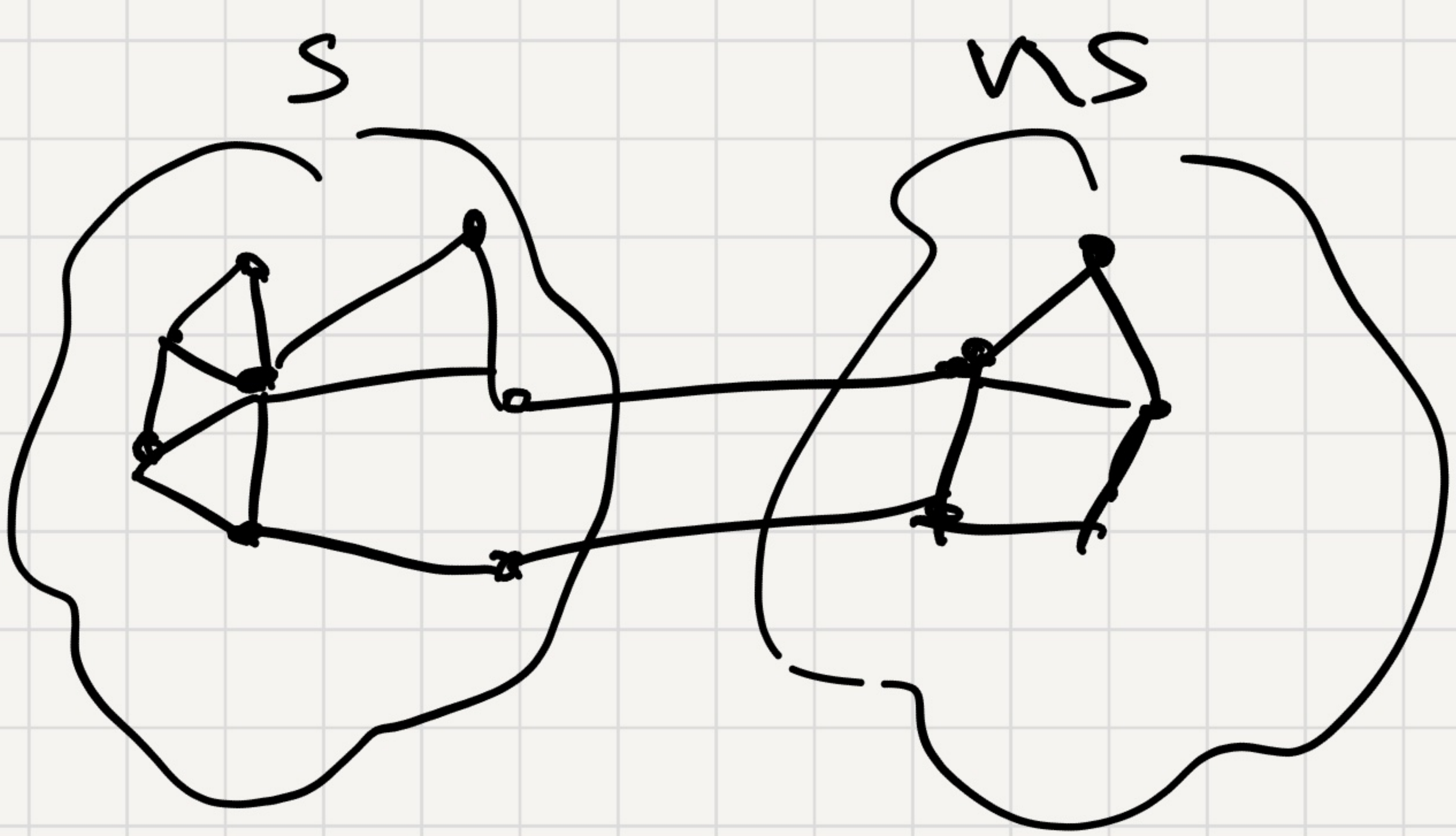


Claim: If G is a connected graph. TFAE:

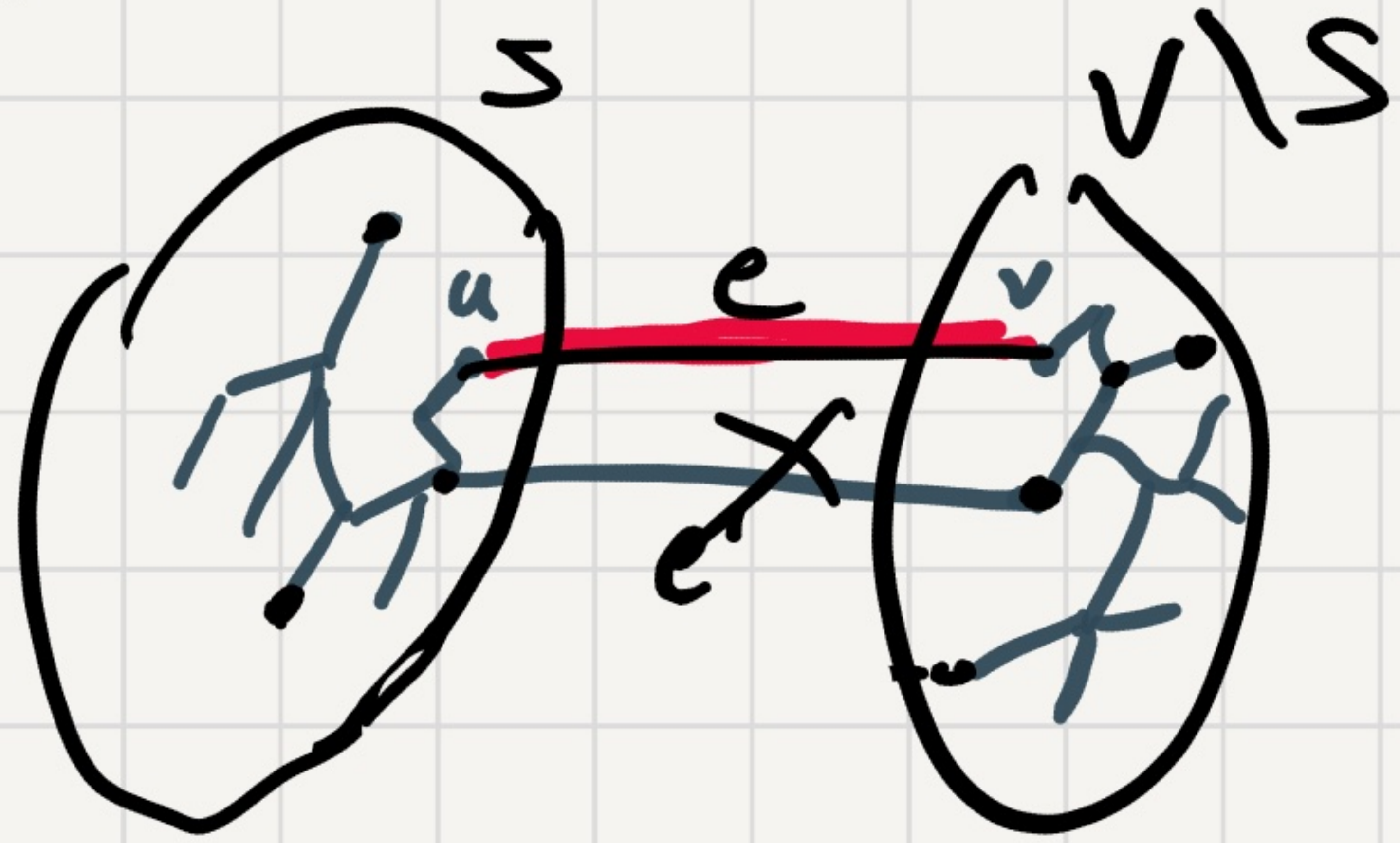
(a) G is acyclic

(b) $|E(G)| = |V(G)| - 1$.

Def'n: A cut in a graph $G = (V, E)$ is any pair of the form $(S, V \setminus S)$ for $S \subseteq V$.



Claim: The cheapest edge in any cut appear in some MST



Let T be some MST.

Proof: Assume $e \notin T$. There's a path in T between u & v

$\Rightarrow \exists$ a cycle in $T \cup \{e\}$

Let's remove e' from T

$$T' = T \cup \{e\} \setminus \{e'\}.$$

$\#$ of edges in T' $= |V|-1$.

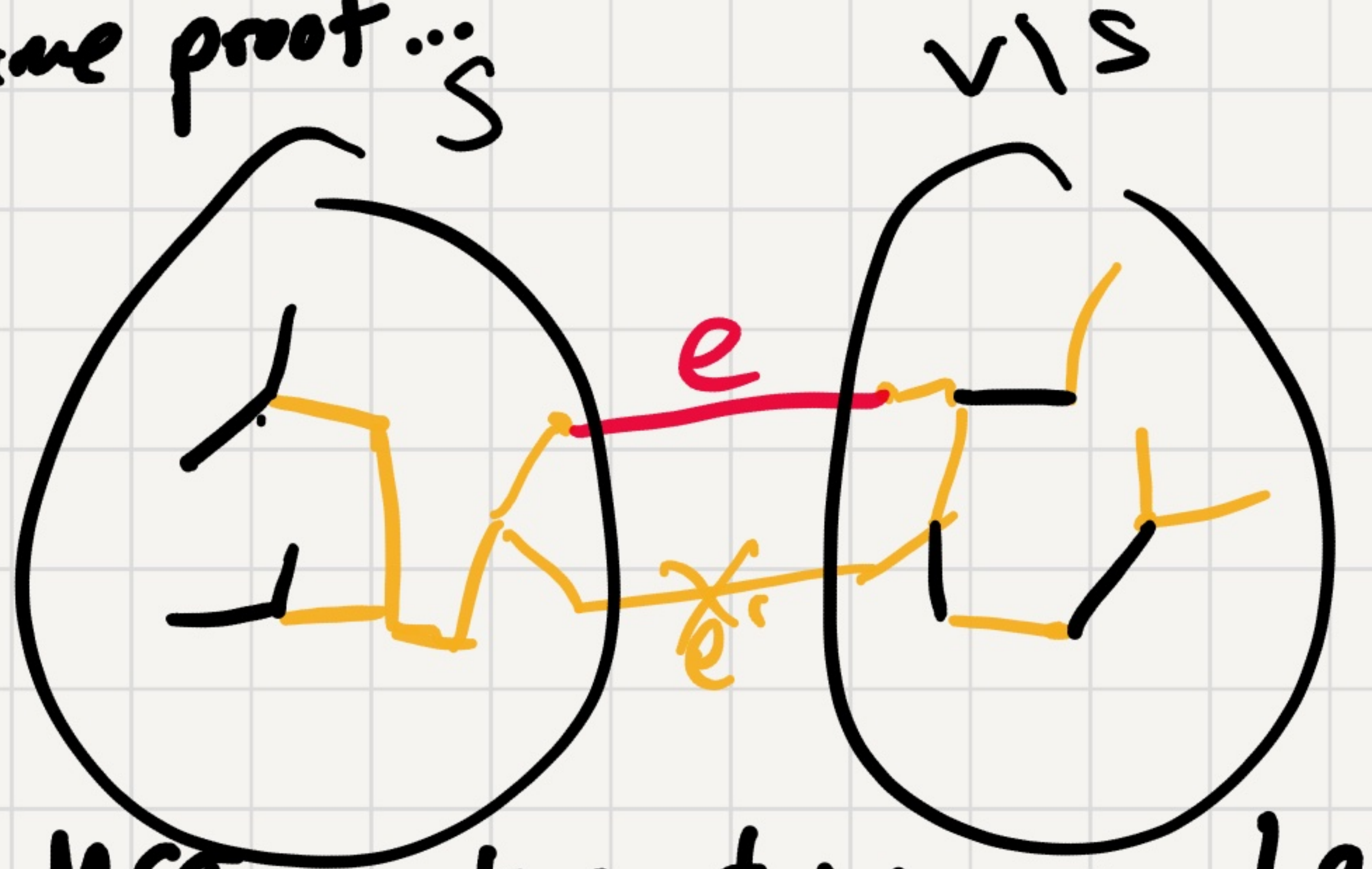
$\Rightarrow T'$ is connected. $\Rightarrow T'$ is a tree.

$$w(T') = \sum_{e \in E(T')} w(e) = w(T) \underbrace{- w(e') + w(e)}_{\leq 0} \leq w(T). \blacksquare$$

More refined statement. Suppose we already picked $X \subseteq E$ s.t. $\exists$ MST T that contains all edges in X.

Claim: Suppose $X \subseteq E$ and $\exists$ MST T s.t. $X \subseteq E(T)$.
Suppose also that X doesn't cross $(S, V \setminus S)$.
If e cheapest edge across $(S, V\setminus S)$ then $X \cup \{e\}$ is contained in some MST T'.

Proof: Exactly the same proof...



X - black edges

T - orange edges

Let T be a MST not containing e. Let e' be an edge in the cut $(S, V\setminus S)$ on the tree path between u & v.
Take $T' = T \setminus \{e'\} \cup \{e\}$. Then $w(T') \leq w(T)$ and T' is an MST containing $X \cup \{e\}$. ■

Meta Algorithm

$X = \emptyset$

Repeat iteratively:

- Pick a cut s.t. X doesn't cross the cut.
- Add the edge e with smallest weight in the cut to X.

Kruskal (G, w)

1. $X \leftarrow \emptyset$
2. sort edges according to their weight.
3. For all edges $e \in E$ (by the above order)
   {if adding e to X creates a cycle $\Rightarrow$ skip }
   Else $X \leftarrow X \cup \{e\}$.

   $|E|$ times.

   time $O(\log |V|)$

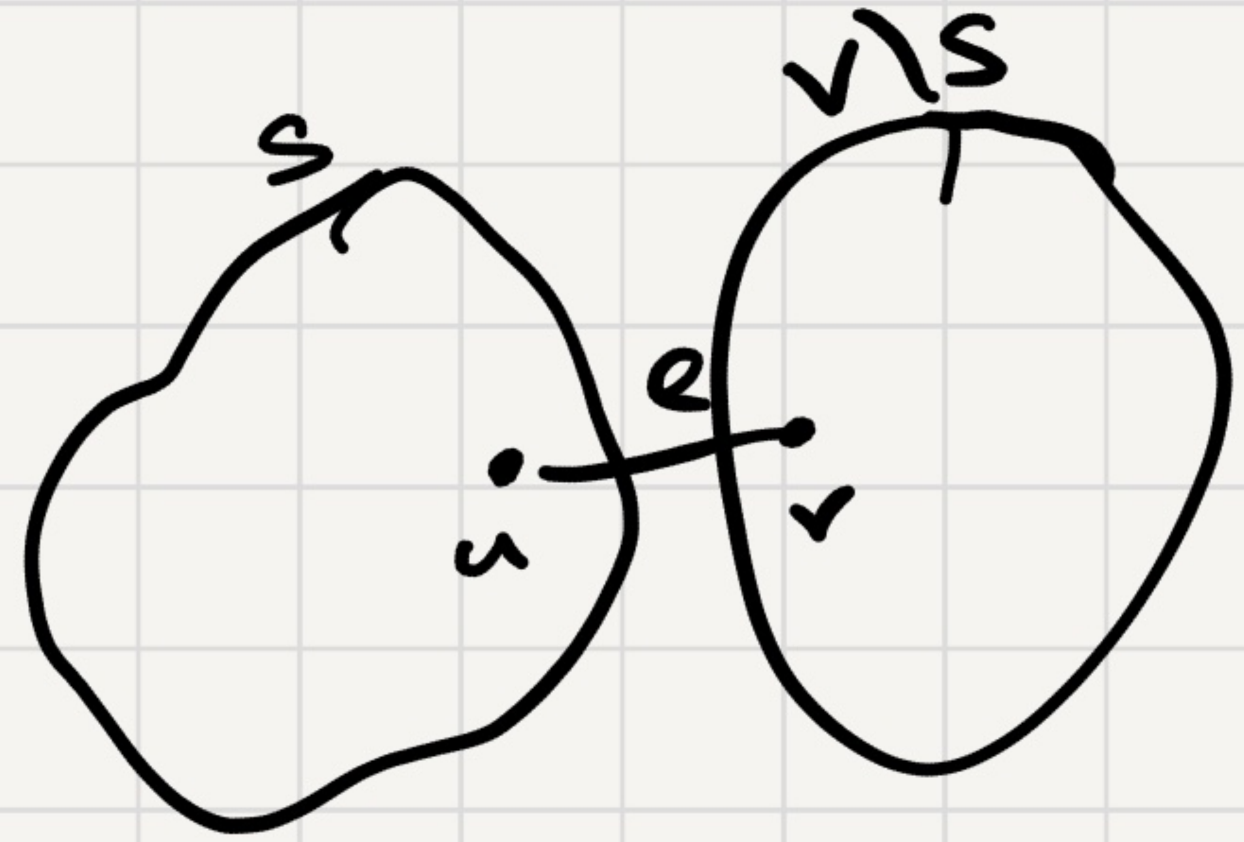
   union-find.
4. Output X.

Running time: next time. Using union-find

$O((|V| + |E|) \log |V|)$.

Claim. Kruskal works correctly:
At any point in time X is a subset of $E(T)$ of some MST T.

Proof: Base case:

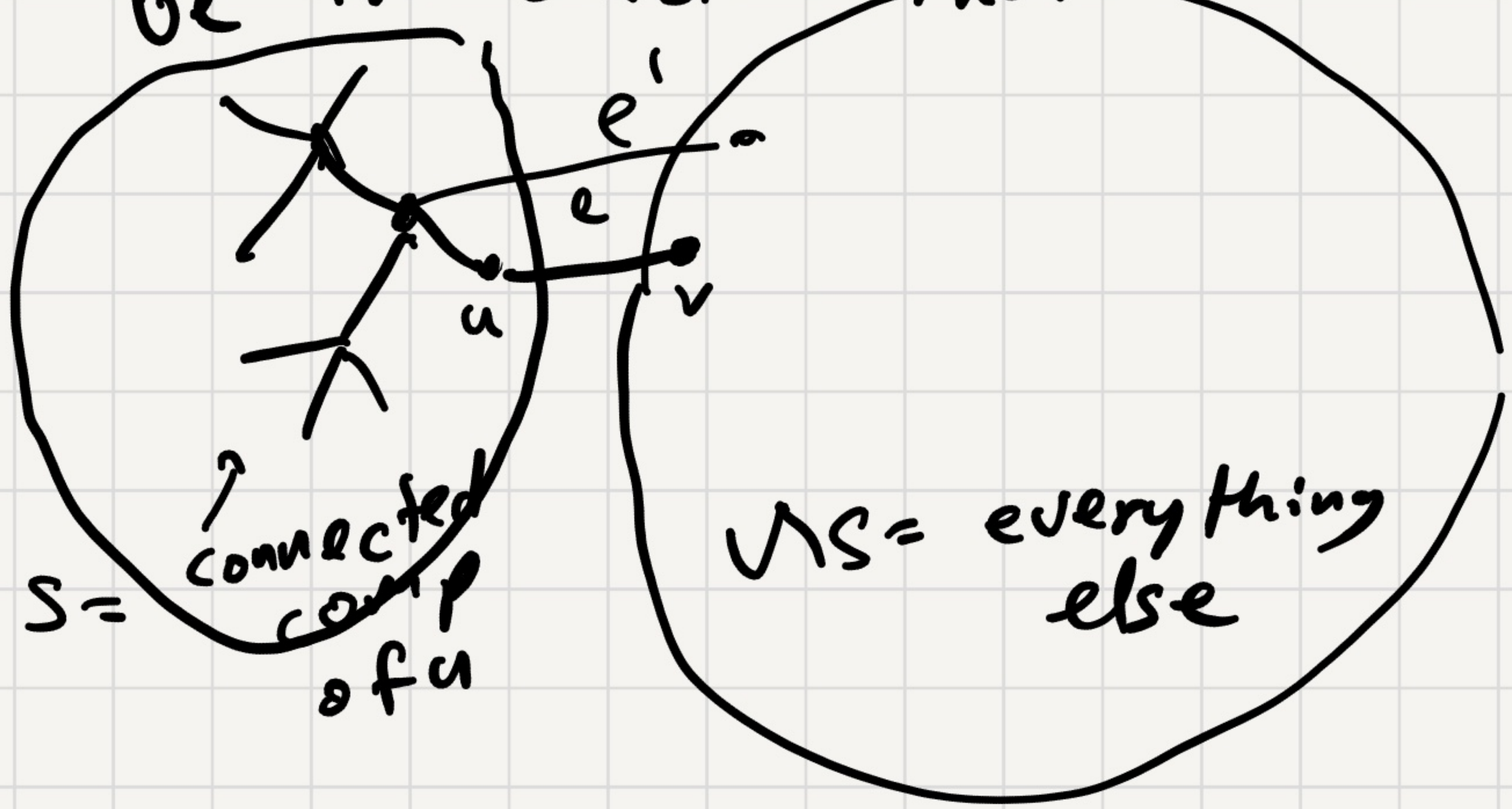


adding the minimal weight edge to X is safe.

Inductive step:

If Kruskal adds e to X then it must be the case that $X \cup \{e\}$ has no cycles.

X is a collection of trees - a forest.

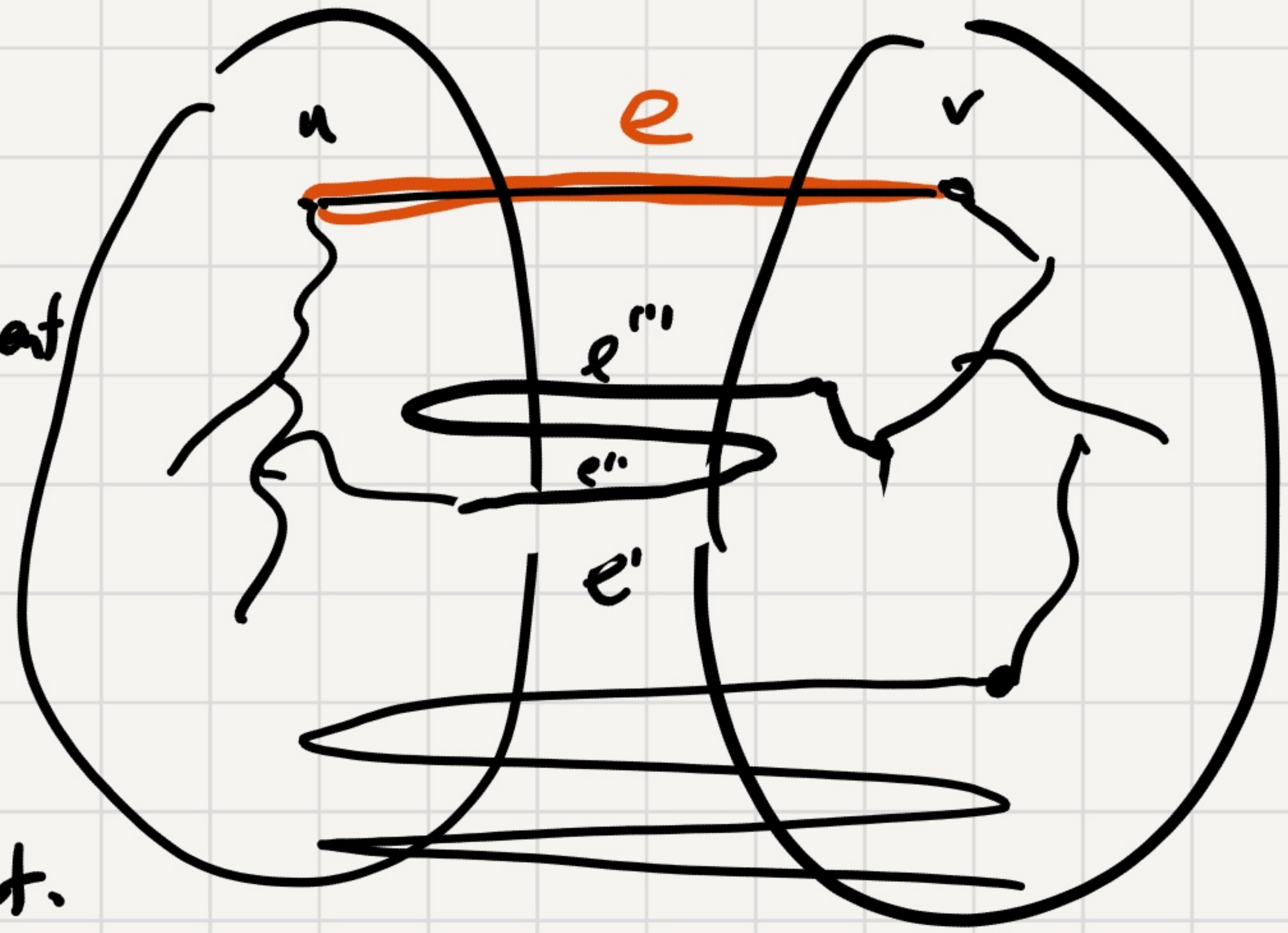


$w(e') \geq w(e)$.

∎

More nuanced view of the exchange argument

there could be multiple edge that belong to T & cross the cut.



We look at the edge on shortest tree path from u to v. Since u & v are at different ends of the cut there must be an edge crossing. (or multiple - the number of those is odd. Think why?).

Remove any one of those edges e' and add e. This gives you an MST T'.