31 January. More Greedy Algos.

Plan

1. Finish Greedy Stays Ahead Analysis
2. Announcements
3. Minimum Spanning Tree Problem

Earliest Finish Time

1. Sort jobs by finish time
2. Schedule = $\{\}$

Iterate through jobs in sorted order $j = 1 \dots n$

- if job j does not conflict w/ Schedule
  - $\hookrightarrow$ Schedule $\leftarrow$ Schedule $\cup \{j\}$

Return Schedule.

Optimal schedule $O^* = \langle [s_1^*, f_1^*], [s_2^*, f_2^*], \ldots, [s_\ell^*, f_\ell^*] \rangle$

EFT schedule $S = \langle [s_1, f_1], [s_2, f_2], \ldots, [s_k, f_k] \rangle$

"If EFT stays ahead, then EFT is optimal"

Earliest Finish Time Lemma

For all $j \in \{1, \ldots, k\}$

$$f_j \leq f_j^* \implies S \text{ is also optimal.}$$

Optimal schedule $O^* = \langle [s_1^*, f_1^*], [s_2^*, f_2^*], \ldots, [s_\ell^*, f_\ell^*] \rangle$

EFT schedule $S = \langle [s_1, f_1], [s_2, f_2], \ldots, [s_k, f_k] \rangle$

"If EFT stays ahead, then EFT is optimal"

Earliest Finish Time Lemma

For all $j \in \{1, \ldots, k\}$

$$f_j \leq f_j^* \implies S \text{ is also } \underline{\text{optimal}}.$$

"EFT stays ahead of O^*"

Greedy Stays Ahead Lemma.

For all $j \in \{1, \ldots, |S|\}$ $\quad f_j \leq f_j^*$

"EFT stays ahead of O^*"

<u>Greedy Stays Ahead Lemma</u>.

For all $j \in \{1, ..., |S|\}$ $\quad f_j \leq f_j^*$

By <u>induction</u> on jobs added by EFT.

<u>Base Case</u>. EFT adds jobs by earliest finish time.

"EFT stays ahead of O^*"

Greedy Stays Ahead Lemma.

For all $j \in \{1, \dots, |S|\}$ $\quad f_j \leq f_j^*$

By induction on jobs added by EFT.

Base Case. EFT adds jobs by earliest finish time.

$\Longrightarrow$ $[s_1, f_1] \in S$ chosen because $f_1 = \min_{i \in [n]} f_i$

"EFT stays ahead of O^*"

Greedy Stays Ahead Lemma.

For all $j \in \{1, \ldots, |S|\}$ $\quad f_j \leq f_j^*$

By induction on jobs added by EFT.

Base Case. EFT adds jobs by earliest finish time.

$\Longrightarrow$ $[s_1, f_1] \in S$ chosen because $f_1 = \min_{i \in [n]} f_i$

$\Longrightarrow$ $f_1 \leq f_1^*$ ✓

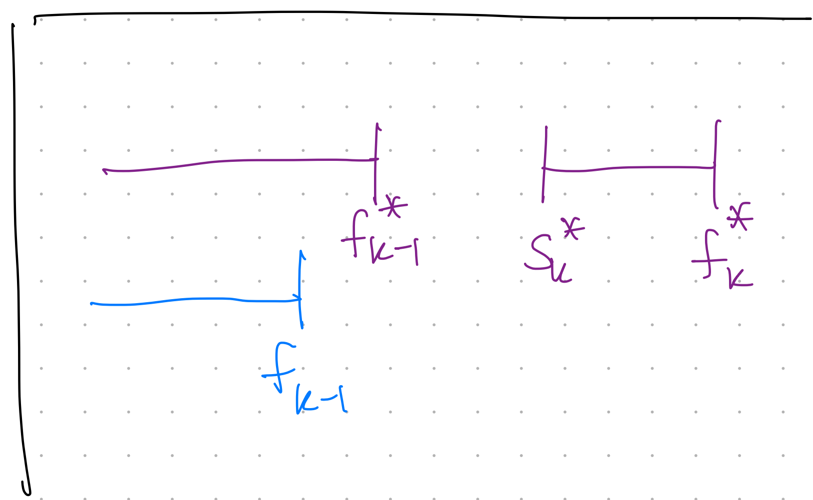
"EFT stays ahead of O^*"

Greedy Stays Ahead Lemma.

For all $j \in \{1, \ldots, |S|\}$ $\quad f_j \le f_j^*$

Inductive Hypothesis. For all $i < k$, $\quad f_i \le f_i^*$

By IH, $\underbrace{f_{k-1} \le f_{k-1}^*}_{\text{By IH}} < s_k^*$. $\quad \Rightarrow \quad f_{k-1} < s_k^*$



"EFT stays ahead of O^*"

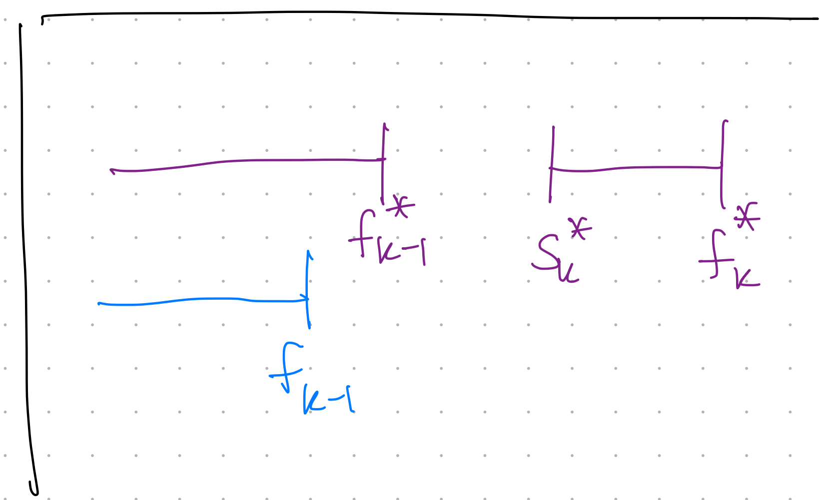
Greedy Stays Ahead Lemma.

For all $j \in \{1, \dots, |S|\}$ $\quad f_j \le f_j^*$

Inductive Hypothesis. For all $i < k$, $\quad f_i \le f_i^*$

By IH, $f_{k-1} \le f_{k-1}^* \le s_k^*$.

So $[s_k^*, f_k^*] \in O^*$ does not conflict $\langle [s_1, f_1] \cdots [s_{k-1}, f_{k-1}] \rangle$



"EFT stays ahead of O^*"

Greedy Stays Ahead Lemma.

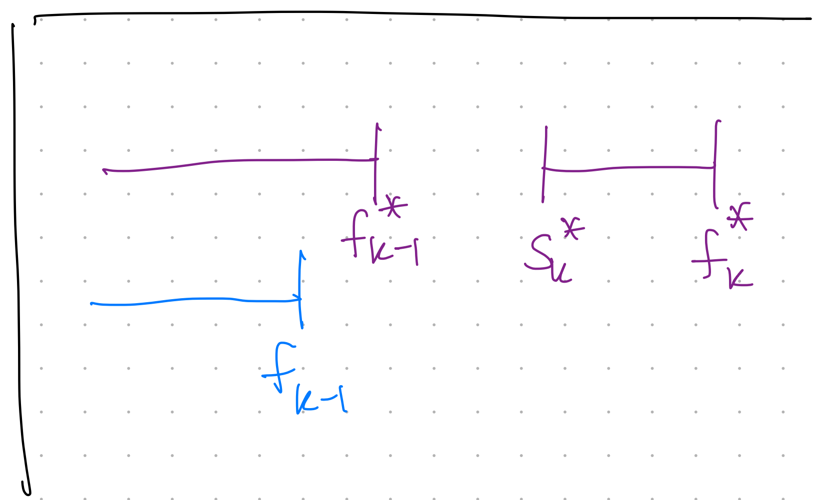
For all $j \in \{1, \ldots, |S|\}$ $\quad f_j \leq f_j^*$

Inductive Hypothesis. For all $i < k$, $\quad f_i \leq f_i^*$

By IH, $f_{k-1} \leq f_{k-1}^* \leq s_k^*$.

So $[s_k^*, f_k^*] \in O^*$ does not conflict $\langle [s_1, f_1] \cdots [s_{k-1}, f_{k-1}] \rangle$

But $[s_k, f_k]$ is non-conflicting job of earliest finish time.



"EFT stays ahead of O^*"

Greedy Stays Ahead Lemma.

For all $j \in \{1, \ldots, |S|\}$ $\quad f_j \leq f_j^*$

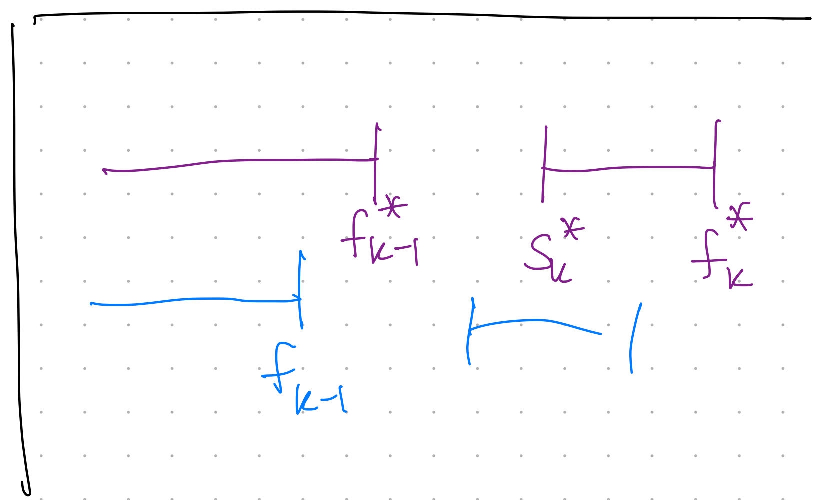
Inductive Hypothesis. For all $i < k$, $\quad f_i \leq f_i^*$

By IH, $f_{k-1} \leq f_{k-1}^* \leq s_k^*$.

So $[s_k^*, f_k^*] \in O^*$ does not conflict $\langle [s_1, f_1] \cdots [s_{k-1}, f_{k-1}] \rangle$

But $[s_k, f_k]$ is non-conflicting job of earliest finish time.

$\Rightarrow \quad f_k \leq f_k^*$



"If EFT stays ahead, then EFT is optimal"

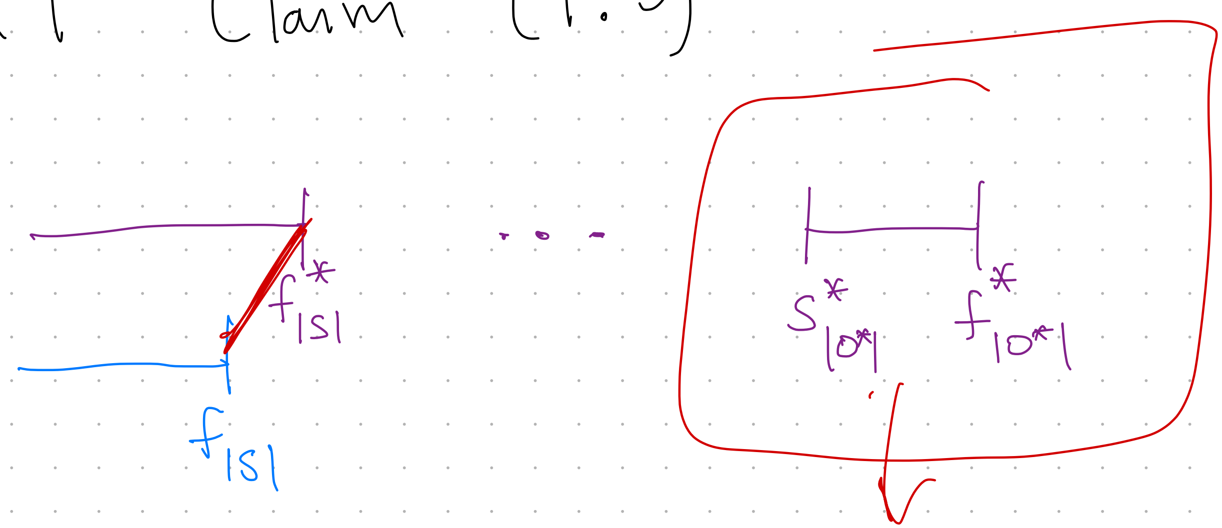
Earliest Finish Time Lemma

For all $j \in \{1, \ldots, k\}$

$$f_j \le f_j^* \implies S \text{ is also } \underline{\text{optimal}}.$$

Similar Argument.

See KT Claim (4.3)



Announcements

* Enrollment cap lifted
* HW 1 due Thurs, 11:59 PM
  ↳ At most 3 slip days (Sun, 11:59 PM)

Greedy Graph Algorithms

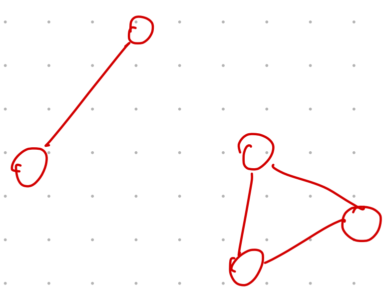
Minimum Spanning Tree

Given a connected, undirected, weighted, graph $G = (V, E, w)$

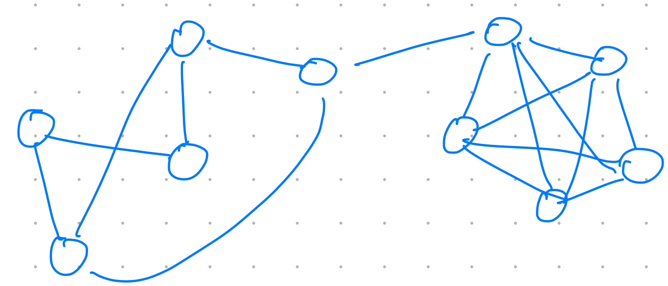
Minimum Spanning Tree

Given a connected, undirected, weighted graph $G = (V, E, w)$

- A graph is connected if $\forall u, v \in V$ $\exists$ a path $p = (e_1, ..., e_{|p|})$ connecting u & v.



Not connected



Connected

Minimum Spanning Tree

Given a connected, undirected, weighted graph $G = (V, E, w)$

- A graph is connected if $\forall u, v \in V$ $\exists$ a path $p = (e_1, \ldots, e_{|p|})$ connecting u & v.

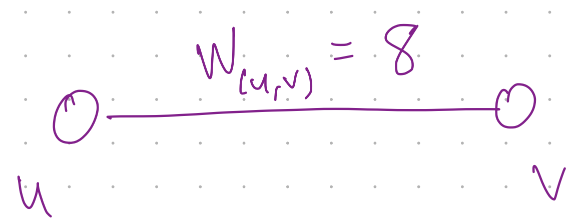
- A graph is undirected if $(u,v) \in E \Leftrightarrow (v,u) \in E$

That is, (u,v) and (v,u) represent the same edge

Minimum Spanning Tree

Given a connected, undirected, weighted graph $G = (V, E, w)$

- A graph is connected if $\forall u, v \in V$ $\exists$ a path $p = (e_1, \dots, e_{|p|})$ connecting u & v.
- A graph is undirected if $(u,v) \in E \iff (v,u) \in E$
- Each edge $e \in E$ has an associated Nonnegative weight $w_e \geq 0$



Minimum Spanning Tree

Given a connected, undirected, weighted graph $G = (V, E, W)$

Find a minimum spanning tree $T = (V, E' \subseteq E)$

Minimum Spanning Tree

Given a connected, undirected, weighted graph $G = (V, E, w)$

Find a minimum spanning tree $T = (V, E' \subseteq E)$

minimum total weight

$$w_T = \sum_{e \in T} w_e$$

Minimum Spanning Tree

Given a connected, undirected, weighted graph $G = (V, E, W)$

Find a minimum spanning tree $T = (V, E' \subseteq E)$

minimum total weight

connects all $u, v \in V$

$$w_T = \sum_{e \in T} w_e$$

Minimum Spanning Tree

Given a connected, undirected, weighted graph $G=(V,E,W)$

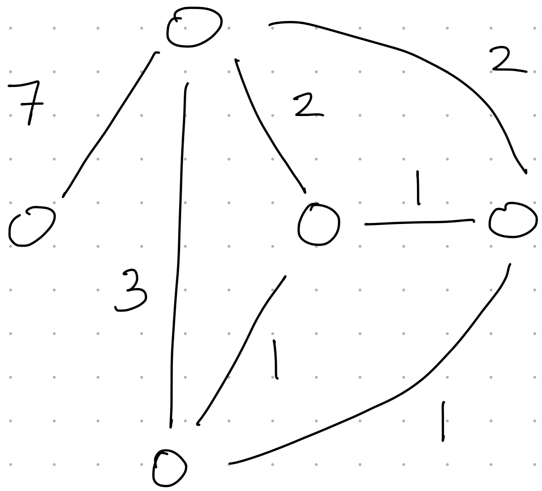
Find a minimum spanning tree $T=(V,E'\subseteq E)$

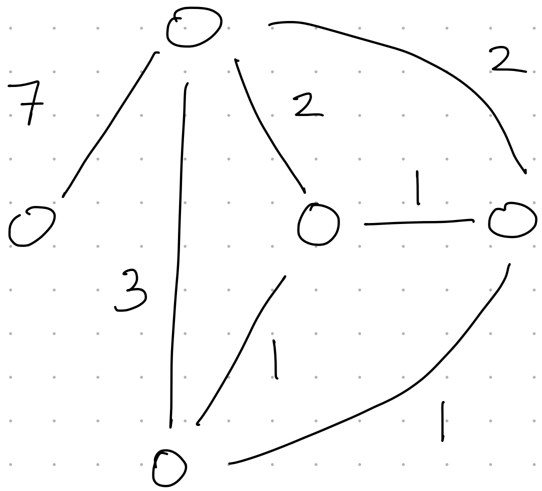
minimum → minimum total weight

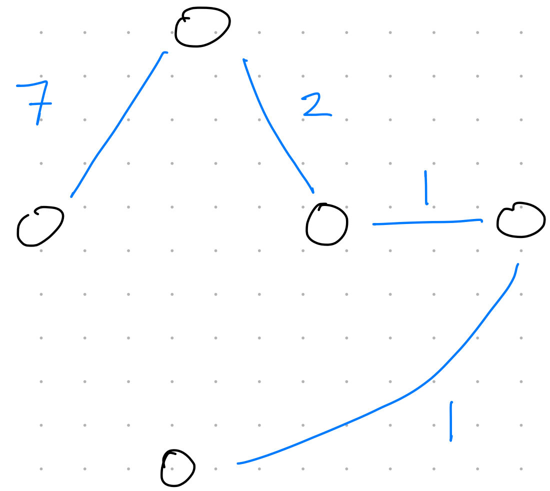
spanning → connects all $u,v \in V$

tree → graph with no cycles

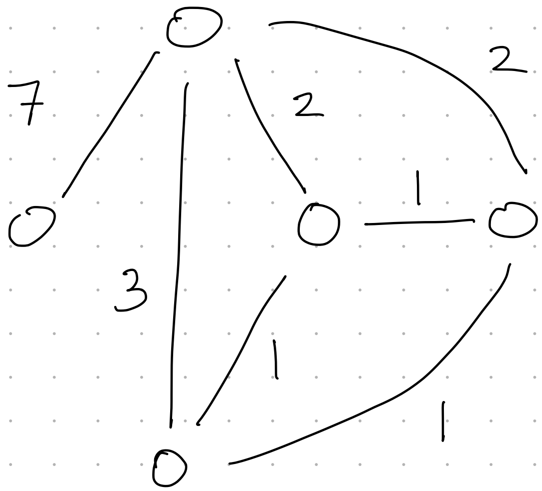
$$W_T = \sum_{e \in T} w_e$$

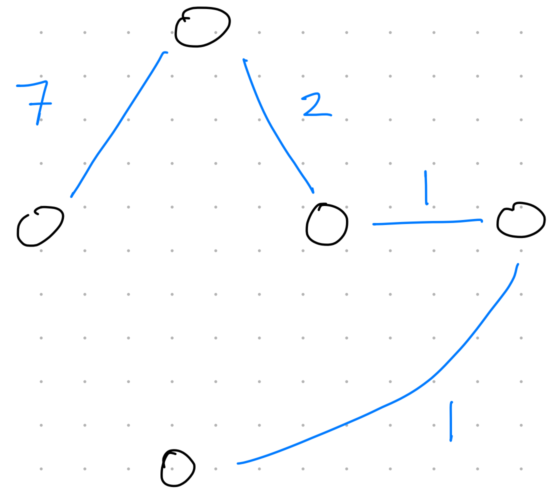

7
2
2
1
3
1
1



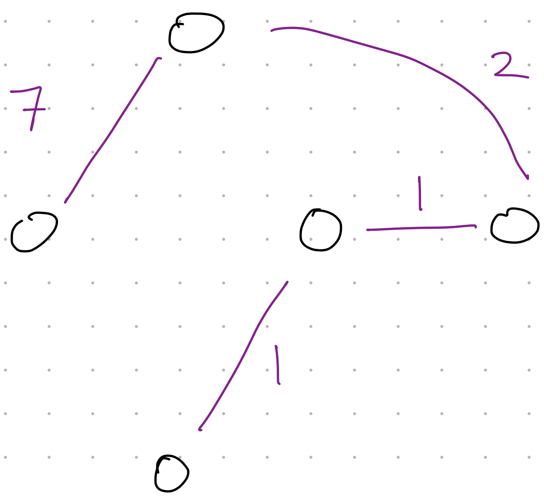


$W_T = 11$

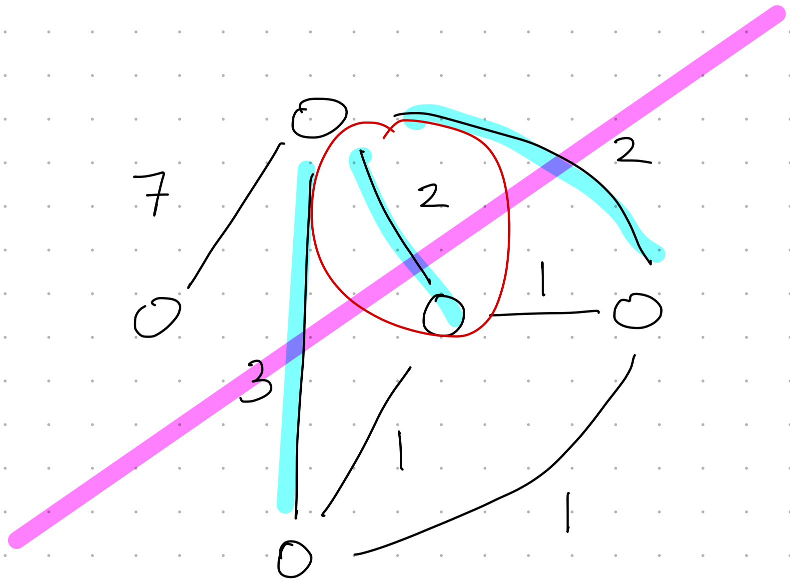


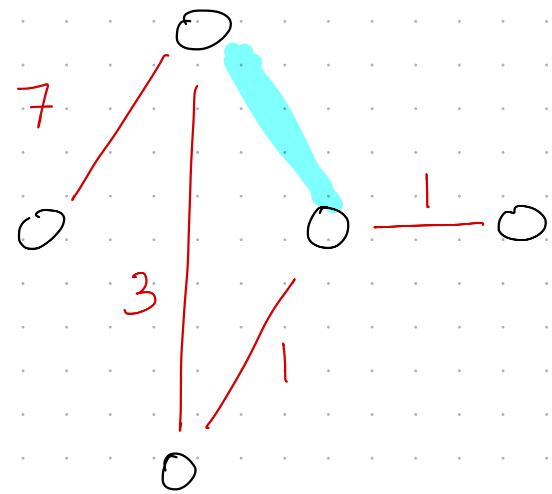


$W_T = 11$



$W_T = 11$



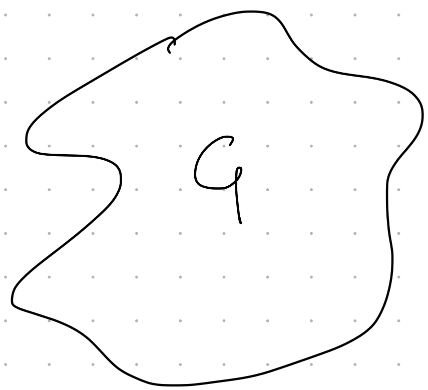


$W_T = 12$

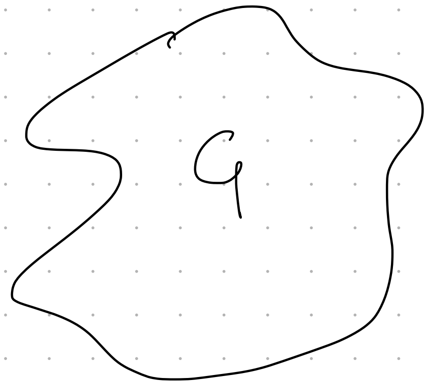
Given a tree T, how do we know that

T is / is not a min spanning tree?

Cut Lemma. Given a graph, $G = (V, E, W)$.



Cut Lemma. Given a graph, $G = (V, E, W)$.

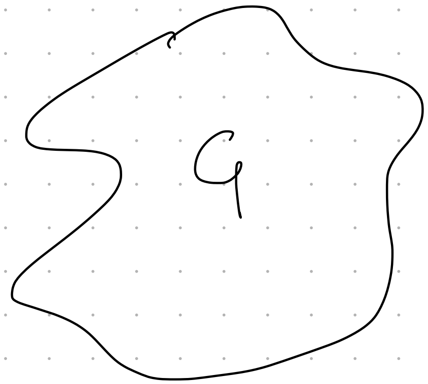


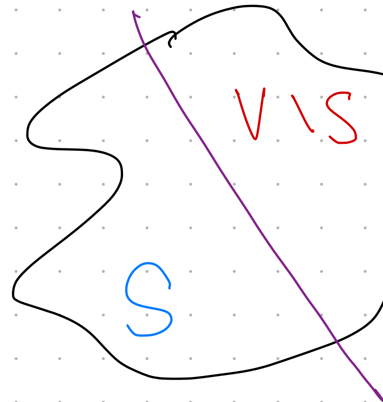
Simplifying Assumption

Edge weights are distinct.

$\forall e, e' \in E, \quad w_e \neq w_{e'}$

Cut Lemma. Given a graph, $G = (V, E, W)$.



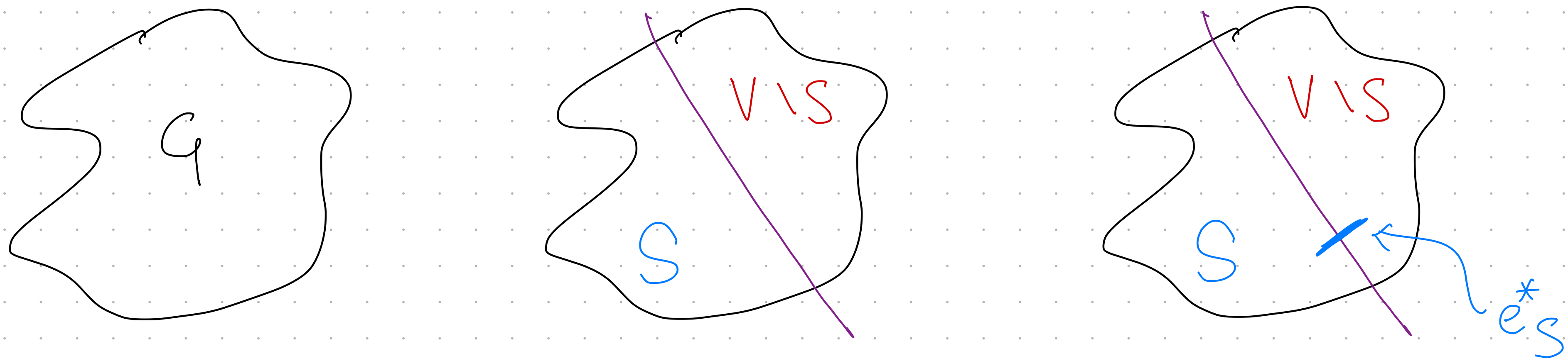


Consider *any* non-empty cut $(S, V \setminus S)$.

partition of vertices into two sets

$S \subseteq V$, $V \setminus S \subseteq V$

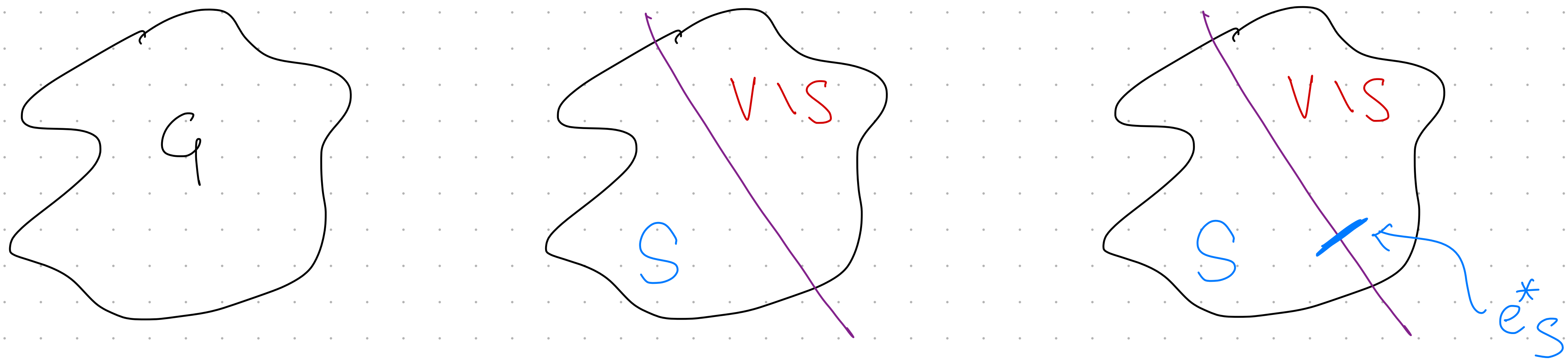
Cut Lemma. Given a graph, $G = (V, E, W)$.



Consider any non-empty cut $(S, V \setminus S)$.

Denote by e_S^* the minimum-weight edge crossing the cut.

Cut Lemma. Given a graph, $G = (V, E, w)$.



Consider *any* non-empty cut $(S, V \setminus S)$.

Denote by e_S^* the minimum-weight edge crossing the cut.

$$\boxed{\text{The MST of } G \text{ contains } e_S^*.}$$

Significance of Cut Lemma?

Given $T = (V, E')$

if there exists some cut $(S, V \setminus S)$

and T does not include e_S^*, then

Significance of Cut Lemma?

Given $T = (V, E')$

if there exists some cut $(S, V \setminus S)$

and T does not include e_S^*, then

T is not an MST.

Significance of Cut Lemma?

Given $T = (V, E')$

if there exists some cut $(S, V \setminus S)$

and T does not include e_S^*, then

T is not an MST.

So, any algorithm for solving MST must "collect" e_S^* for every non-empty $(S, V \setminus S)$

Proof of Cut Lemma. By exchange argument.

Start with a spanning tree T_0 that does not contain some e_S^*.

Proof of Cut Lemma. By exchange argument.

Start with a spanning tree T_0 that does not contain some e_S^*.

Show how to exchange e_S^* for an edge e in T_0 s.t. the tree weight drops.

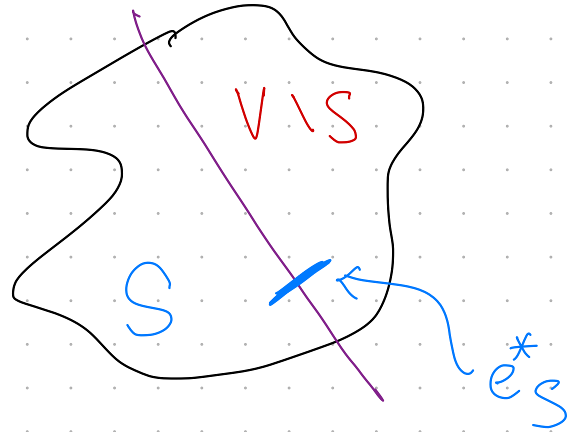
Proof of Cut Lemma. By exchange argument.

Start with a spanning tree T_0 that does not contain some e_S^*.

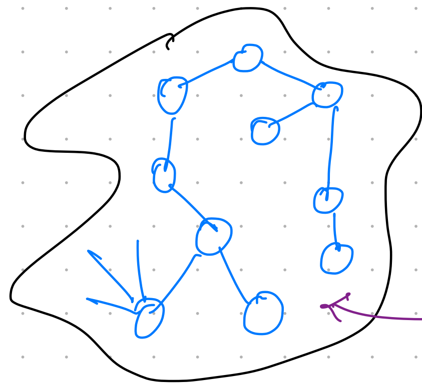
Show how to exchange e_S^* for an edge e in T_0 s.t. the tree weight drops.

$\Longrightarrow$ T_0 cannot be MST.

$G =$



T_0



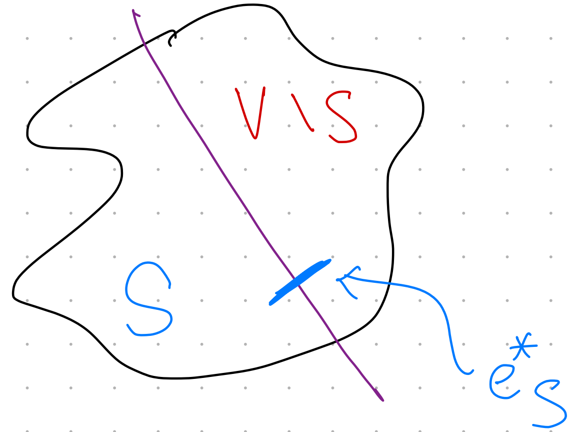
$e_S^* \notin T_0$ (by assumption)

By defn. e_S^* connects some $u \in S$, $v \in V \setminus S$

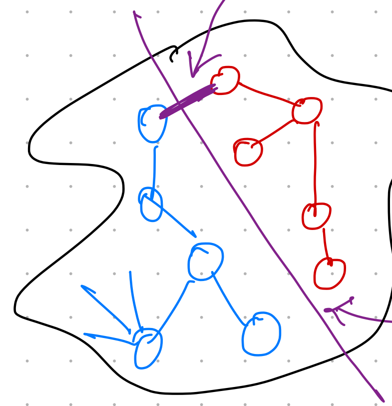
<u>Idea 0</u>. T_0 is a spanning tree $\Longrightarrow$ connected.

$\Longrightarrow$ must be some other edge from $S \longrightarrow V \setminus S$

$G =$



T_0



$e_0 \in T_0$ crosses $(S, V \setminus S)$

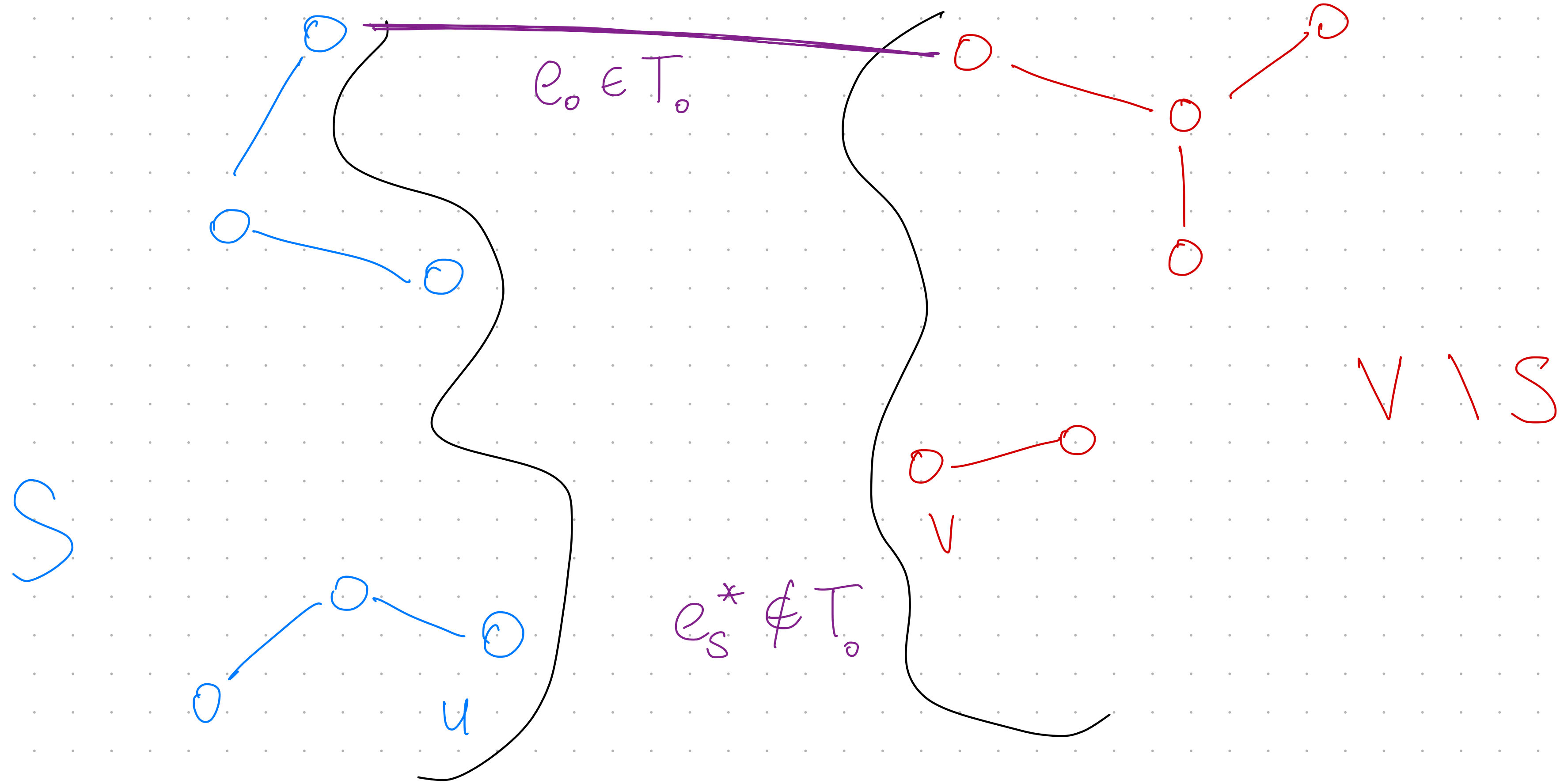
$e_S^* \notin T_0$ (by assumption)

By defn. e_S^* connects some $u \in S$, $v \in V \setminus S$

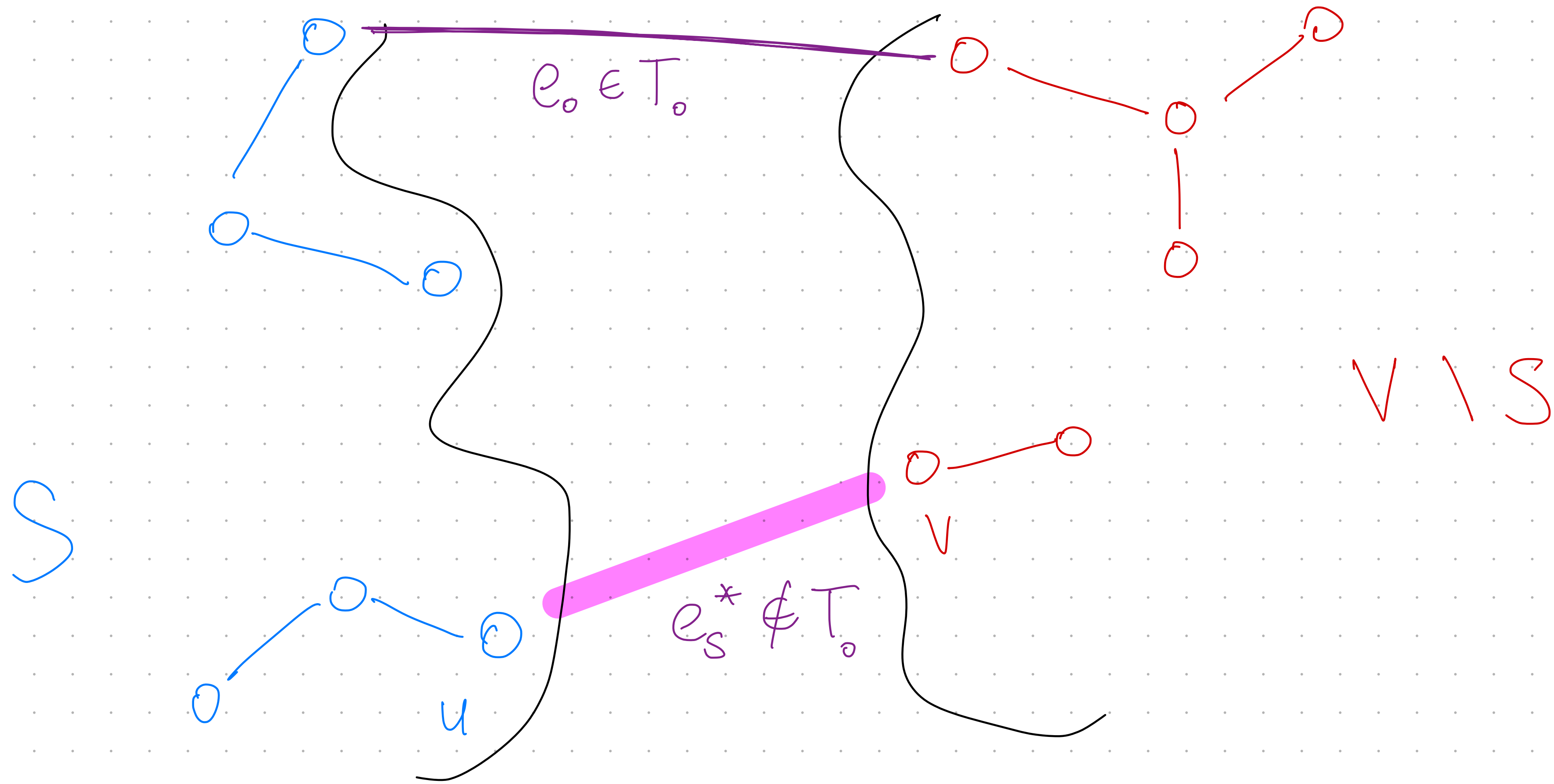
<u>Idea 0.</u> T_0 is a tree $\Longrightarrow$ connected.

$\Longrightarrow$ must be some other edge from $S \longrightarrow V \setminus S$

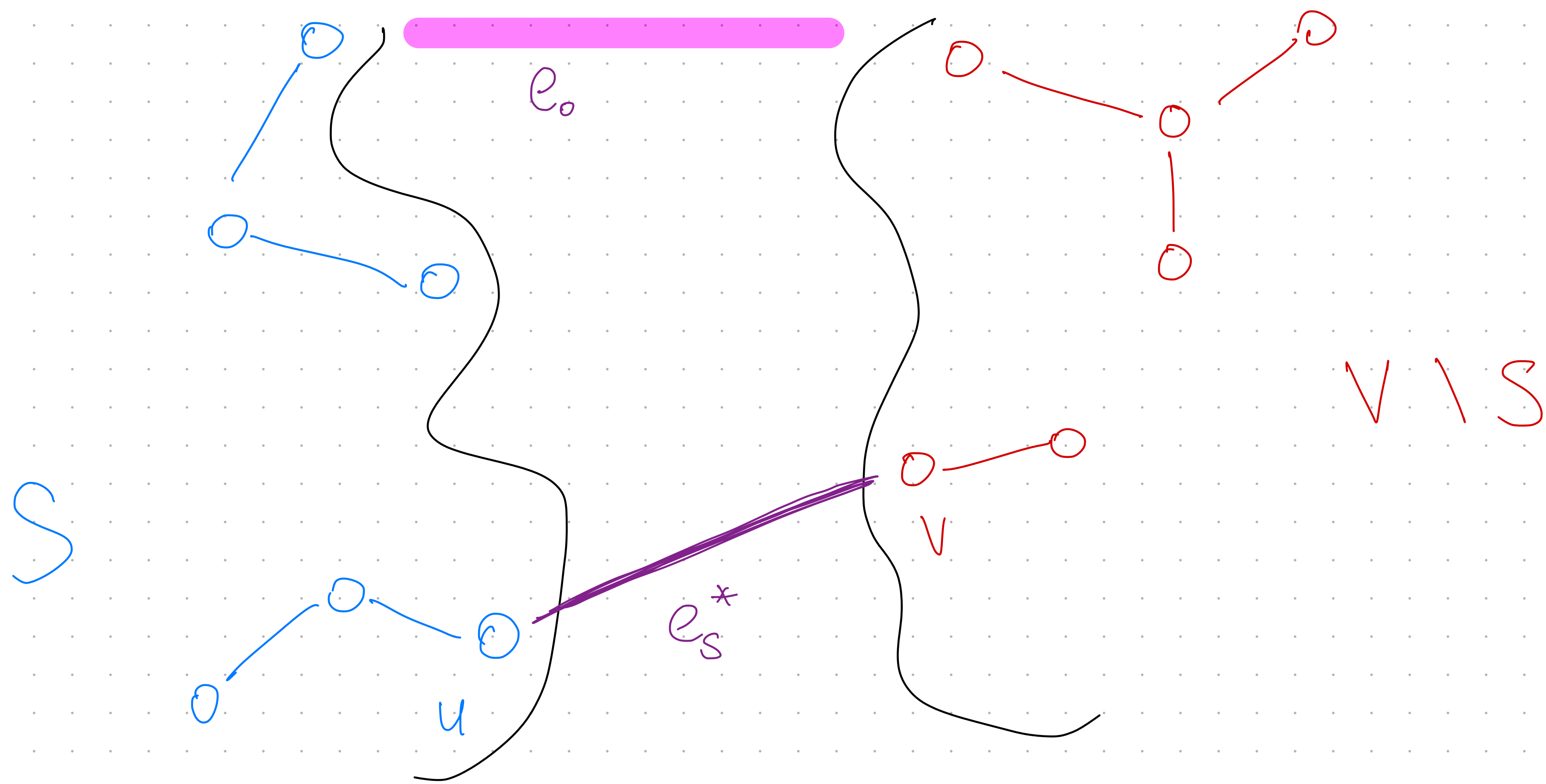
Idea 0. Exchange e_s^* for e_0.



Idea 0. Exchange e_s^* for e_0.



Idea 0. Exchange e_s^* for e_0.

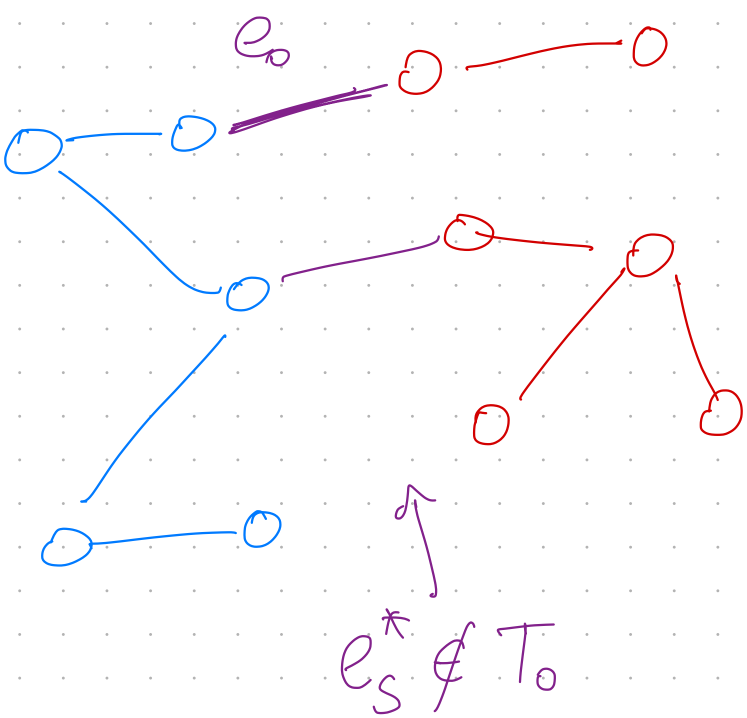


Total weight: $W_{e_s^*} - W_{e_0} + W_{T_0}$

By defn. $W_{e_s^*} < W_{e_0}$ $\Longrightarrow$ Total weight decreases!

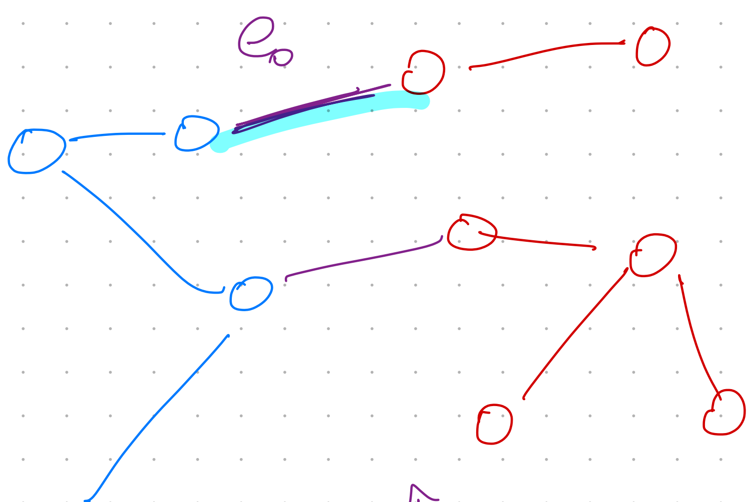
Are we done?

No. Exchanging e_s^* for e_0 might not preserve tree structure.

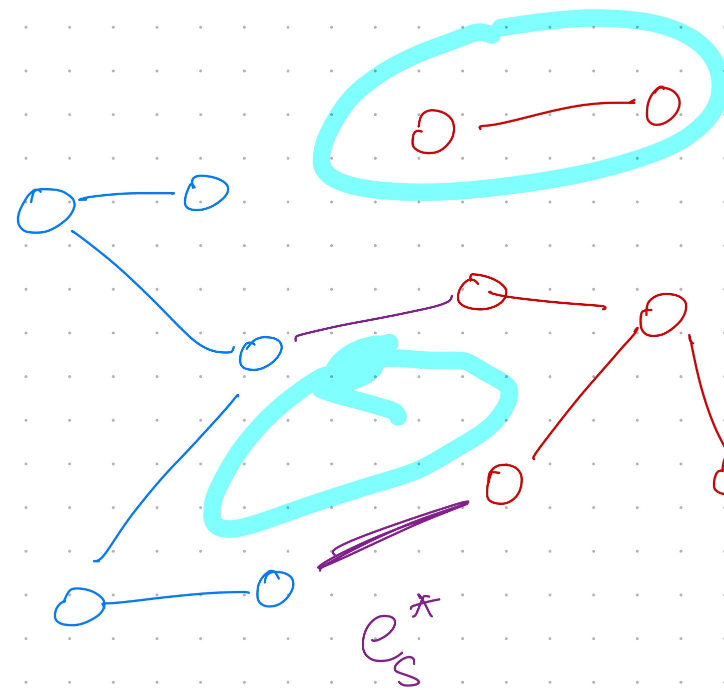


Are we done?

<u>No</u>. Exchanging e_s^* for e_0 might not preserve tree structure.

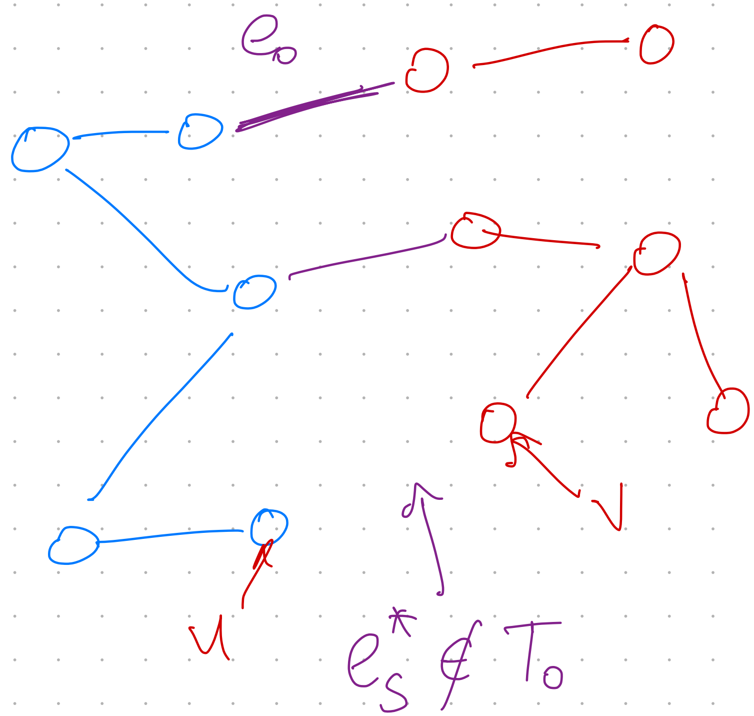


$e_s^* \notin T_0$

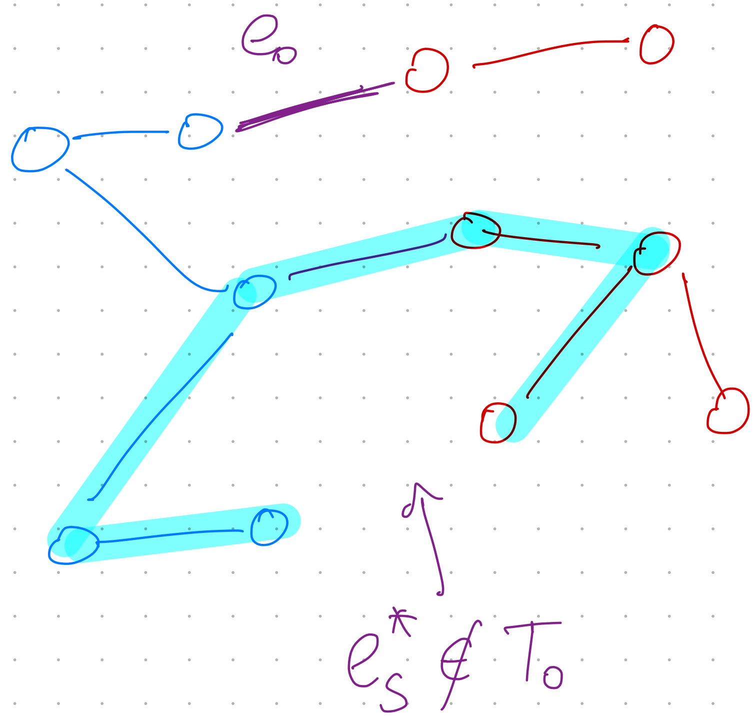


disconnected & contains cycle.

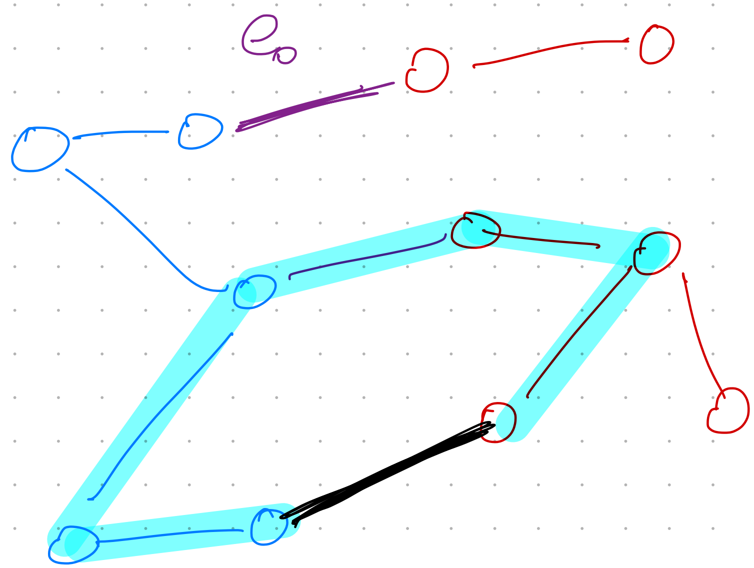
Idea 1. Find a path through T_0 connecting ends of e_s^*.



Idea 1. Find a path through T_0 connecting ends of e_s^*.



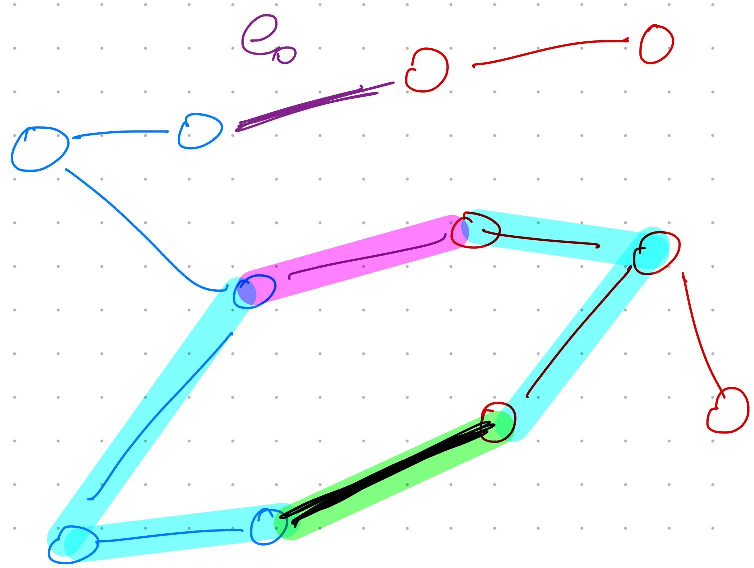
Idea 1. Find a path through T_0 connecting ends of e_s^*.



add e_s^* to form a cycle

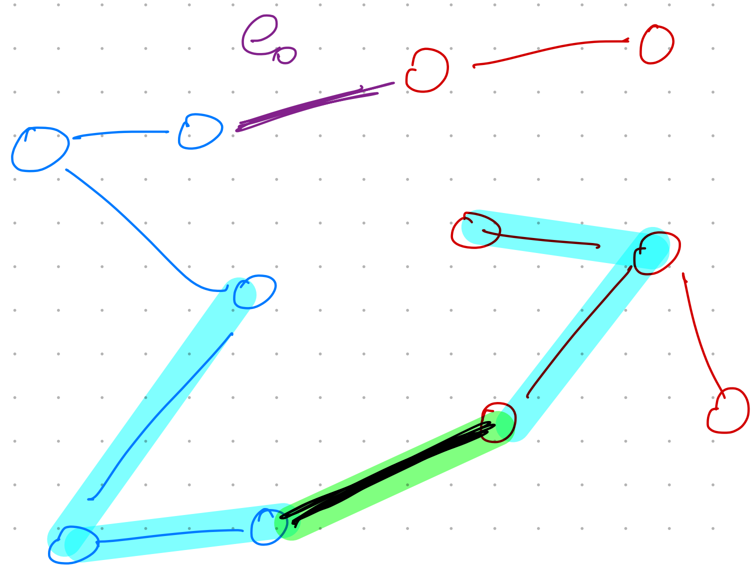
↳ New graph still connected.

Idea 1. Find a path through T_0 connecting ends of e_s^*.



Remove edge e in the cycle, where $w_{e_s^*} < w_e$

Idea 1. Find a path through T_0 connecting ends of e_s^*.



Remove edge e in the cycle, where $w_{e_s^*} < w_e$

↳ Weight goes down

↳ Removing edge from cycle cannot disconnect graph.

$\Longrightarrow$ tree $\boxtimes$