24 January 2025

Greedy Algorithms: Minimum Spanning Tree

Plan

* Minimum Spanning Tree (MST) Problem
* Announcements
* Greedy Algorithms

Minimum Spanning Tree

vertices edges weights

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 $T = (V, E' \subseteq E)$

tree → graph with no cycles

Minimum Spanning Tree

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

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

spanning: $\forall u, v \in V$, u and v connected in T

tree: graph with no cycles

Minimum Spanning Tree

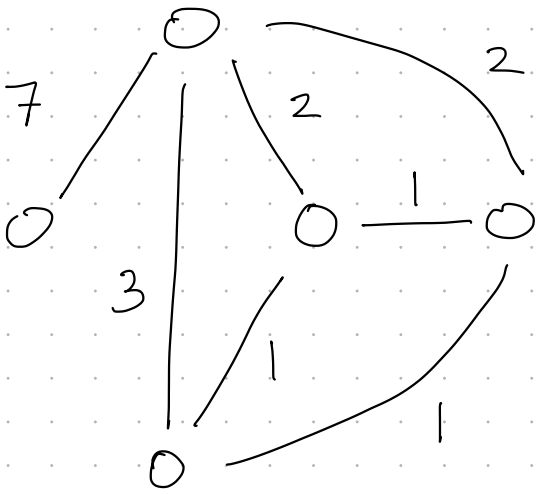
Given a connected, undirected, weighted graph $G = (V, E, w)$

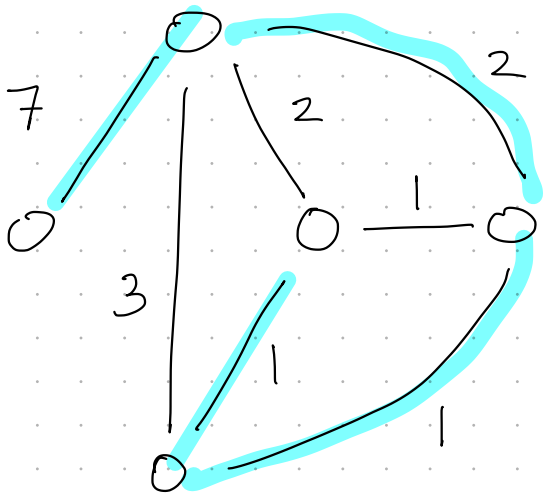
Find a minimum spanning tree $T = (V, E' \subseteq E)$

- minimum: minimize sum of edge weights.

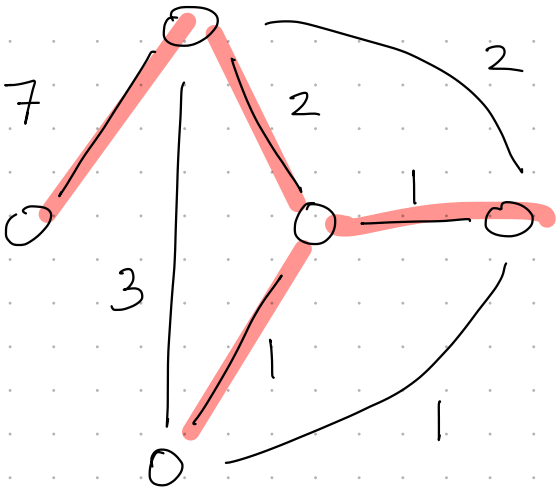
$$w_T = \sum_{e \in T} w_e$$

- spanning: $\forall u, v \in V$, u and v connected in T
- tree: graph with no cycles

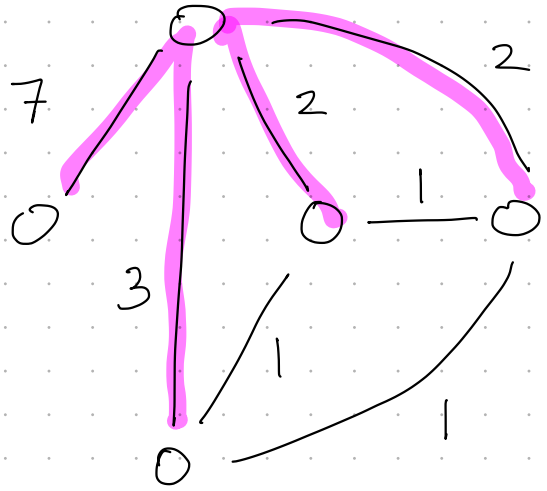




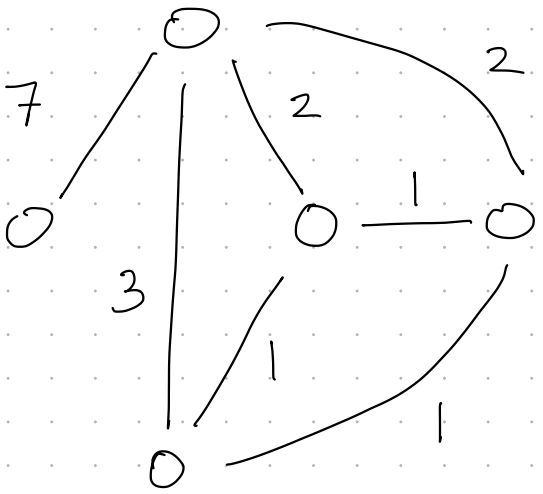
$W_T = 11$



$W_T = 11$



$W_T = 14$


7
2
2
1
3
1
1

Announcements.

Update to wording of HW0 Q1(d)

* Canvas published (SORRY!)
* KT added to CAMP.
* OH start this afternoon
  - ↳ Prof. Kim → Gates 203, 1-3p
  - ↳ See full schedule on course page.

Locations: Rhodes 590 ← dedicated space 4820 24/7.

Rhodes 655 ↑ large room for "peak" OH

MST Algorithms.

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

Return a minimum spanning tree.

Approach. Greedy Algorithms

Greedy Algorithms

Design Paradigm. Choose solution "greedily"

Myopic → local decisions that "look good"

Irrevocable → once we make a decision, we never reconsider.

Greedy Algorithms

Design Paradigm. Choose solution "greedily"

Myopic → local decisions that "look good"

Irrevocable → once we make a decision, we never reconsider.

Advantage. Fast & Local

Running Time?

Greedy Prototype.

① Sort by "priority" $O(n \log n)$

② Iterate through elems in priority order
↳ Make decision about elem.
"constant" $O(1)$ time

$O(n)$

Greedy Algorithms

Design Paradigm. Choose solution "greedily"

Myopic → local decisions that "look good"

Irrevocable → once we make a decision, we never reconsider.

Advantage. Fast & Local

Greedy Prototype

1. Sort by "priority" ← Design of Greedy Algorithm
2. Iterate through elems in priority order
   ↳ Make decision about elem.

Greedy Algorithms

Design Paradigm. Choose solution "greedily"

Myopic → local decisions that "look good"

Irrevocable → once we make a decision, we never reconsider.

Warning. Challenging to analyze correctness

More often than not,

Greedy approaches are incorrect.

Greedy Algorithms for MST

Given graph $G = (V, E, W)$

Greedy Prototype

1. Sort by "priority"

weights

2. Iterate through elems in priority order

edges

   ↳ Make decision about elem.

Two MST Algs.

On input $G = (V, E, W)$

Step 0. Sort edges by weight

$$w_{e_1} < w_{e_2} < \cdots < w_{e_m}$$

Two MST Algs.

On input $G = (V, E, W)$

Step 0. Sort edges by weight

$$w_{e_1} < w_{e_2} < \cdots < w_{e_m}$$

Add Min

Delete Max

Two MST Algs.

On input $G = (V, E, W)$

Step 0. Sort edges by weight

$$w_{e_1} < w_{e_2} < \cdots < w_{e_m}$$

Add Min

$T = (V, \{\})$

For $i = 1 \longrightarrow m$.

  if $T \cup \{e_i\}$ does not form a cycle

    Add e_i to T.

Return T.

Delete Max

$T = (V, E)$

For $i = m \longrightarrow 1$.

  if $T \setminus \{e_i\}$ is still connected

    Remove e_i from T

Return T.

Which algorithm (if any) is correct?

Given a tree T, how do we know that

T is / is not a min spanning tree?

Which algorithm (if any) is correct?

Given a tree T, how do we know that

T is / is not a min spanning tree?

Simplifying Assumption.

Assume edge weights are distinct.

Given G, are there any edges that MUST NOT be in the MST?

(Consider cycles w/in G.)

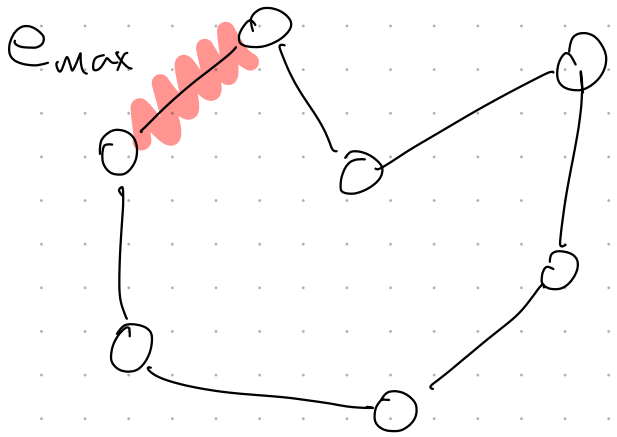
The Cycle Lemma.

$G = (V, E, W)$ w/ distinct edge weights.

Suppose C is a cycle within G and let e_{max} be the edge in C of maximum weight.

Then,

e_{max} is NOT in the MST of G.



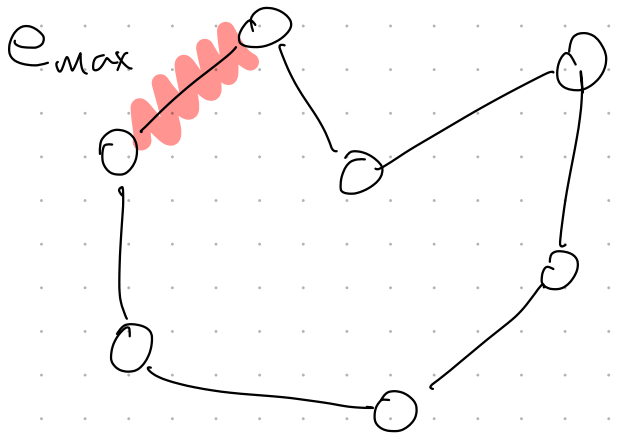
The Cycle Lemma.

$G = (V, E, W)$ w/ distinct edge weights.

Suppose C is a cycle within G and let e_{max} be the edge in C of maximum weight.

Then,

e_{max} is NOT in the MST of G.



Corollary. DeleteMax returns the MST of G.

↳ KT "Reverse Delete"

Proof of Correctness of Delete Max (assuming Cycle Lemma)

Delete Max

$T = (V, E)$

For $i = m \longrightarrow 1$.

  if $T \setminus \{e_i\}$ is still connected

    Remove e_i from T

Return T.

Proof of Correctness of Delete Max (assuming Cycle Lemma)

By contradiction,

Suppose the MST is T^*, but Delete Max returns $T \neq T^*$

$\Rightarrow$ Delete Max deletes some edge $(u, v) \in T^*$.

Delete Max

$T = (V, E)$

For $i = m \longrightarrow 1$.

  if $T \setminus \{e_i\}$ is still connected

    Remove e_i from T

Return T.

Proof of Correctness of Delete Max (assuming Cycle Lemma)

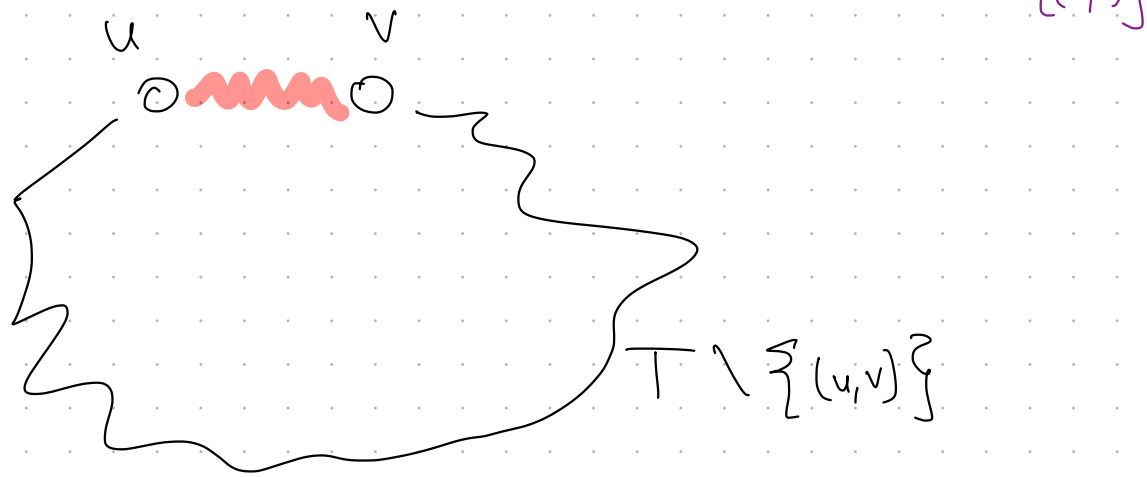
By contradiction,

Suppose the MST is T^*, but Delete Max returns $T \neq T^*$

$\Rightarrow$ Delete Max deletes some edge $(u,v) \in T^*$.

When (u,v) is removed from T,

$\exists$ a path p from $u \rightarrow v$ in $T \setminus \{(u,v)\}$



Delete Max

$T = (V, E)$

For $i = m \rightarrow 1$.

  if $T \setminus \{e_i\}$ is still connected

    Remove e_i from T

Return T.

Proof of Correctness of Delete Max (assuming Cycle Lemma)

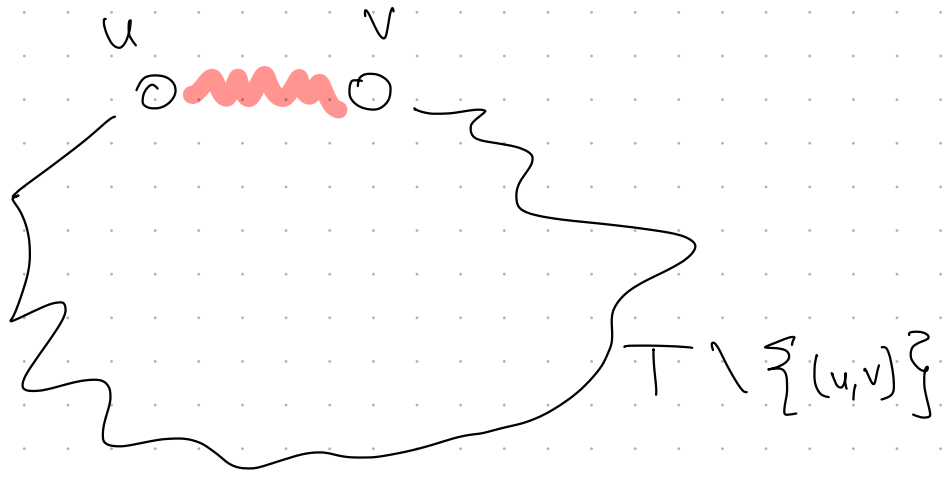
By contradiction,

Suppose the MST is T^*, but Delete Max returns $T \neq T^*$

$\Rightarrow$ Delete Max deletes some edge $(u,v) \in T^*$.

When (u,v) is removed from T,

$\exists$ a path p from $u \rightarrow v$ in T



$\Rightarrow p \cup \{(u,v)\}$ is a cycle in G

Delete Max

$T = (V, E)$

For $i = m \rightarrow 1$.

  if $T \setminus \{e_i\}$ is still connected

    Remove e_i from T

Return T.

Proof of Correctness of Delete Max (assuming Cycle Lemma)

By contradiction,

Suppose the MST is T^*, but Delete Max returns $T \neq T^*$

$\Rightarrow$ Delete Max deletes some edge $(u,v) \in T^*$.

$\Rightarrow$ $P \cup \{(u,v)\}$ is a cycle in G

But why did we remove (u,v)?

Delete Max

$T = (V, E)$

For $i = m \longrightarrow 1$.

  if $T \setminus \{e_i\}$ is still connected

    Remove e_i from T

Return T.

By removal order (greatest → least)

(u,v) is the max wt. edge in cycle!

Proof of Correctness of Delete Max (assuming Cycle Lemma)

By contradiction,

Suppose the MST is T^*, but Delete Max returns $T \neq T^*$

$\Rightarrow$ Delete Max deletes some edge $(u,v) \in T^*$.

$\Rightarrow$ $P \cup \{(u,v)\}$ is a cycle in G

But why did we remove (u,v)?

Delete Max

$T = (V, E)$

For $i = m \longrightarrow 1$.

  if $T \setminus \{e_i\}$ is still connected

    Remove e_i from T

Return T.

By removal order (greatest → least)

(u,v) is the max wt. edge in cycle!

$\Rightarrow$ By the Cycle Lemma

$(u,v) \notin T^*$

(A contradiction)

So, to complete proof of correctness for DeleteMax we need to prove the Cycle Lemma.

So, to complete proof of correctness for Delete Max we need to prove the Cycle Lemma.

Assumptions

1. $G = (V, E, W)$ w/ distinct edge wts.
2. C is a cycle in G containg e_{max}
3. e_{max} is the max wt edge on C

Conclusion

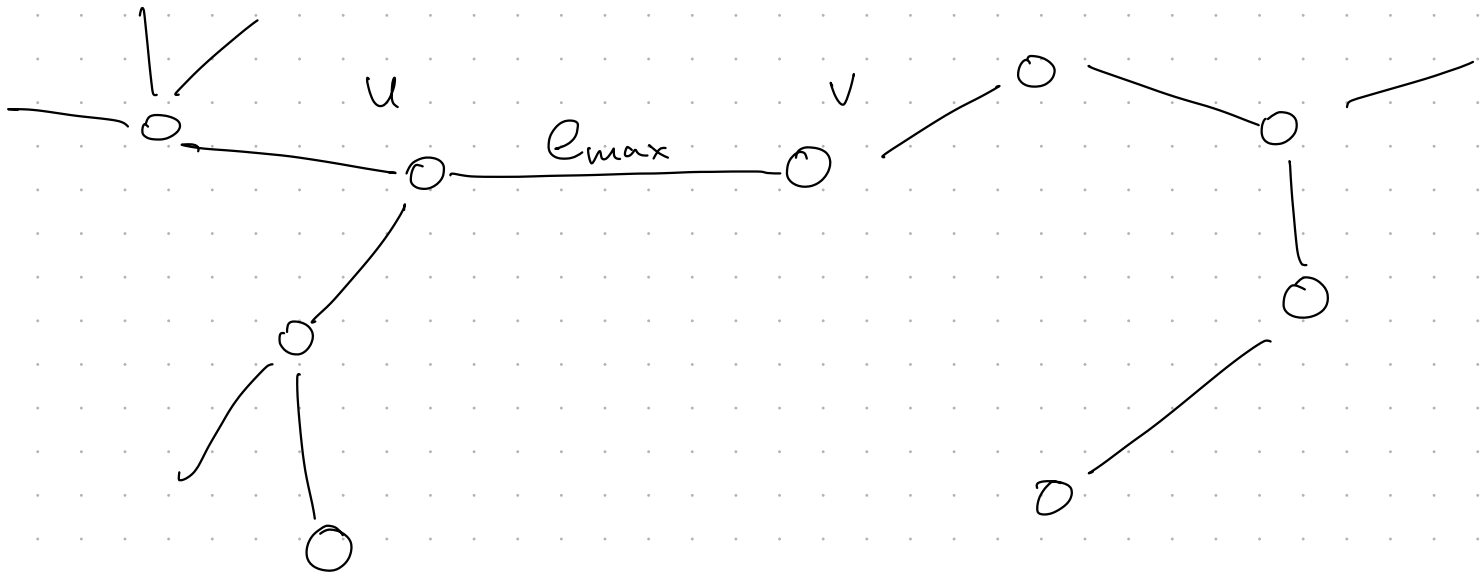
* e_{max} is NOT in MST.

Next. Exchange Argument

Pf. By exchange argument.

Suppose T is a spanning tree s.t. $e_{max} = (u, v) \in T$

We show that T is NOT the minimum spanning tree.

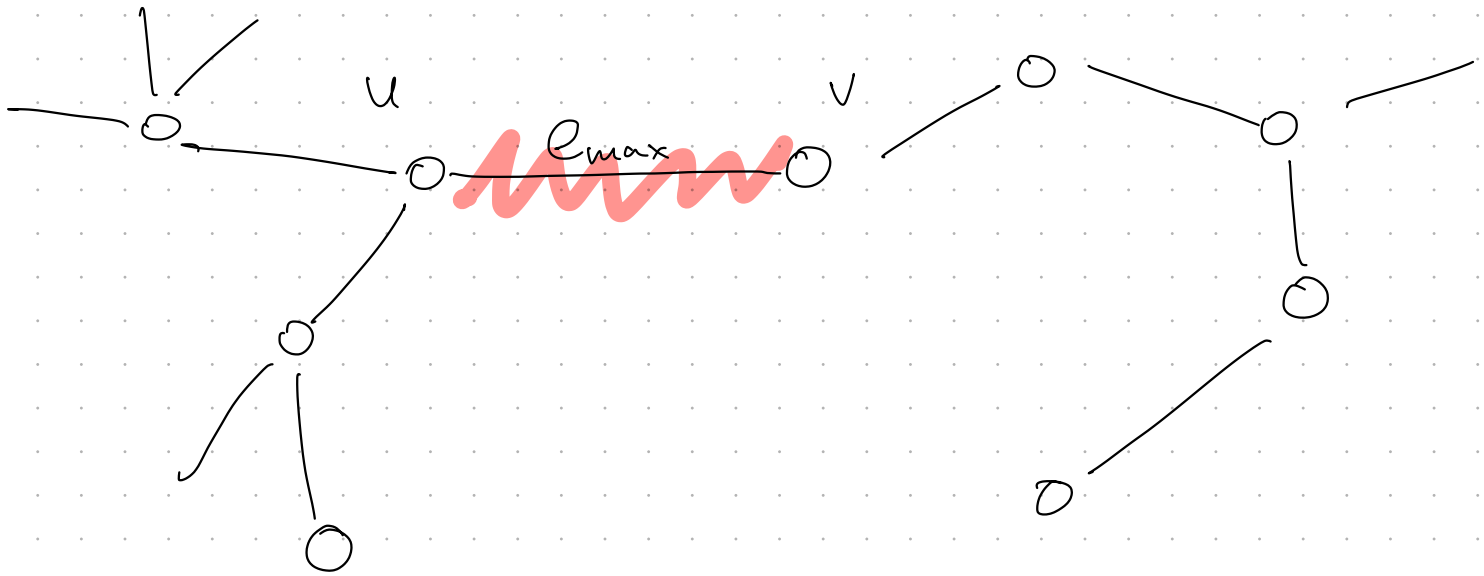


Pf. By exchange argument.

Suppose T is a spanning tree s.t. $e_{max} = (u, v) \in T$

We show that T is NOT the minimum spanning tree.

Consider removing e_{max} from T.

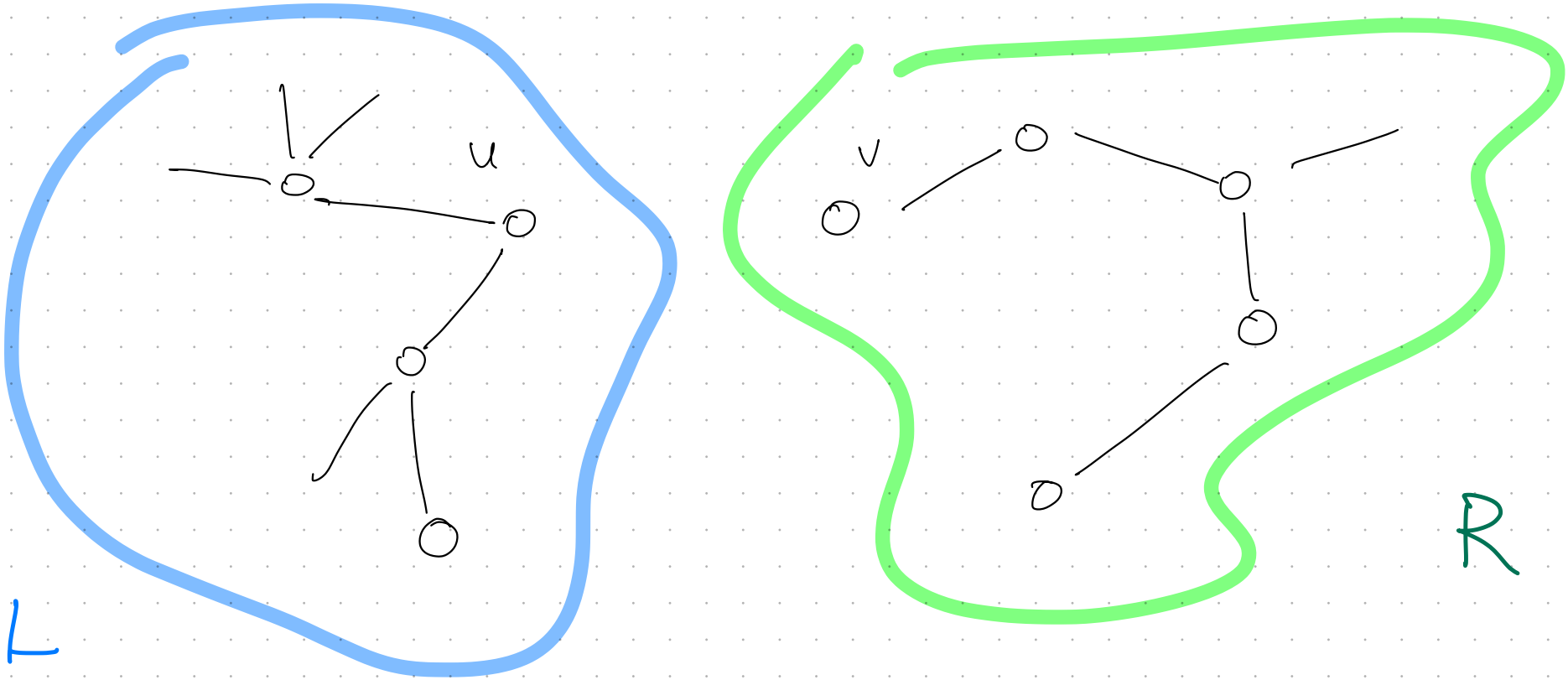


Pf. By exchange argument.

Suppose T is a spanning tree s.t. $e_{max} = (u, v) \in T$

We show that T is NOT the minimum spanning tree.

Disconnects T into two pieces.



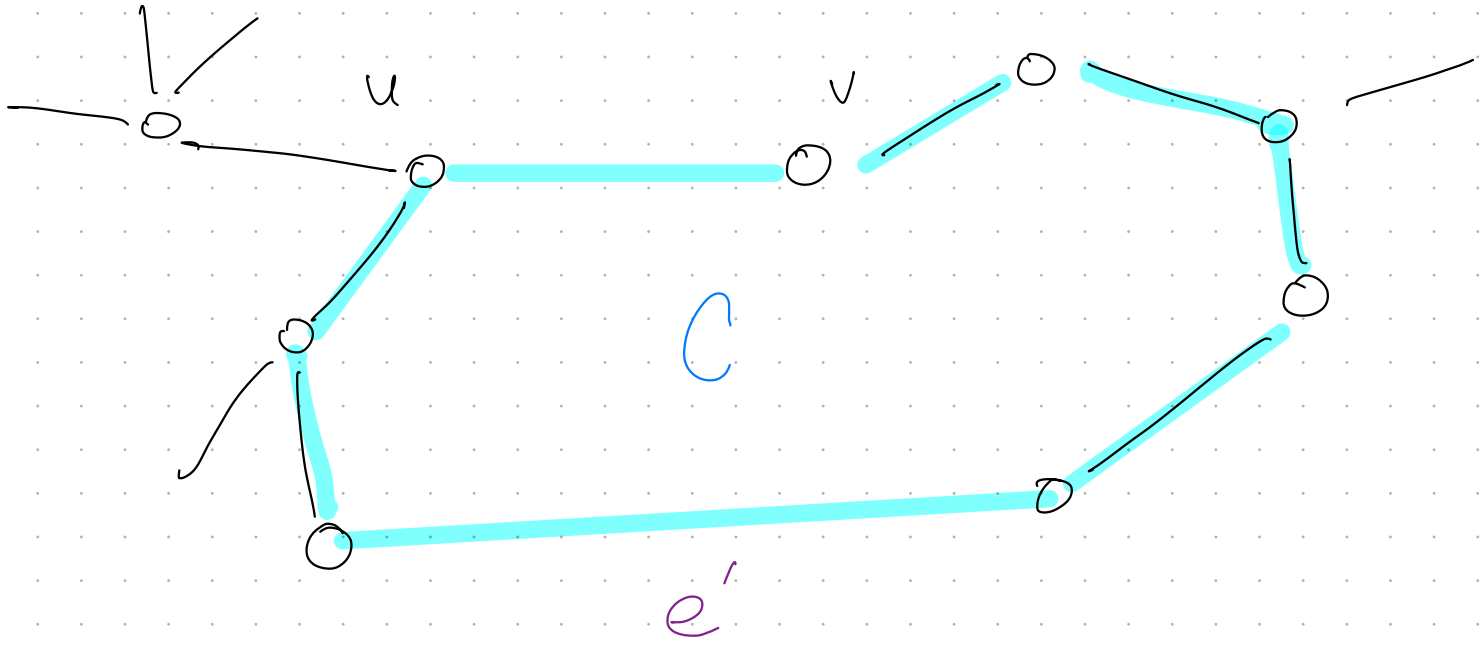
Pf. By exchange argument.

Suppose T is a spanning tree s.t. $e_{max} = (u, v) \in T$

We show that T is NOT the minimum spanning tree.

By (2), e_{max} is in some cycle C within G.

$\Rightarrow$ there exists some edge $e' \in C$ (but not in T) from L to R



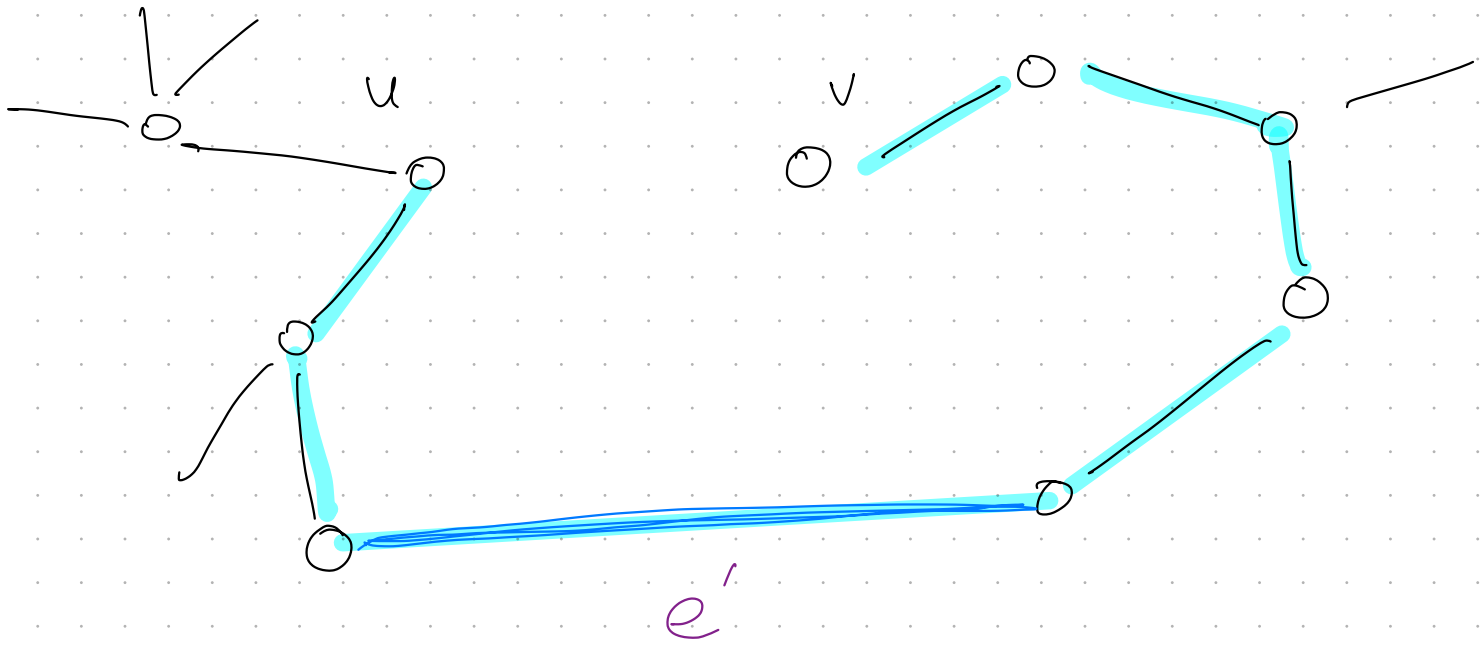
Pf. By exchange argument.

Suppose T is a spanning tree s.t. $e_{max} = (u, v) \in T$

We show that T is NOT the minimum spanning tree.

Consider exchanging e' for e_{max}.

$$T' = T \setminus \{e_{max}\} \cup \{e'\}$$



Claims.

① T' has smaller weight than T

② T' is a tree

③ T' is connected (i.e. a spanning tree)

Claims.

① T' has smaller weight than T

- e_{max} is the max wt. edge on C
- exchanged for $e' \in C$.

$\Rightarrow w_{T'} < w_T$ by ① & ③

② T' is a tree

③ T' is connected (i.e. a spanning tree)

Claims.

① T' has smaller weight than T

② T' is a tree

- if we add e' to T, we only introduce 1 cycle
- Removing e_{max} breaks this cycle.

③ T' is connected (i.e. a spanning tree)

Claims.

① T' has smaller weight than T

② T' is a tree

③ T' is connected (i.e. a spanning tree)

- Original T was spanning.
- Any path using e_{max} can use detour along C to go from $u \leftrightarrow v$.

Pf. By exchange argument.

Suppose T is a spanning tree s.t. $e_{max} = (u, v) \in T$

We show that T is NOT the minimum spanning tree.

Thus, T' is a spanning tree w/ $w_{T'} < w_T$

So T is NOT the MST. ▨