24 January 2025	Greedy Algorithms:
· · · · · · · · · · · · · · · · · · ·	Minimum Spanning Tree
· · · · · · · · · · · · · · · · · · ·	. .
Plan	. .
* Minium Spanning Tr	er (MST) Problem
* Announcements	<pre></pre>
* Greedy Algorithms	
· · · · · · · · · · · · · · · · · · ·	· ·
· · · · · · · · · · · · · · · · · · ·	. .
· · · · · · · · · · · · · · · · · · ·	. .
· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·

Miniu	mum Spannin	g Tree	ventices we	ights
	a undirected weighted	J graph	$ \begin{array}{c} \begin{array}{c} \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} $
Find a	a minimum sp	anning tre	$\mathcal{F} = \left(\mathcal{V}_{f} \in \mathcal{F} \right)$	
	· · · · ·
. 	· · · · ·
. 	· · · · ·
. 	
. 	· · · · · · · · · · · ·	. .	· · · · ·

Minimum Spann	ning Tree
Given a undirec- weight	ted, graph G= (V, E, W) ted
Find a minimum s	spanning tree $T=(V, E' \in E)$
. .	
. .	graph with no cycles
. .	· · · · · J. · · · · · · · · · · · · · ·
. .	. .
 	

Minimum Spanning Tree	· · · · · · · · · · · · · · · · · · ·
Given a undirected, graph (weighted	$ \mathcal{A} = \left(\begin{array}{cccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 $
Find a minimum spanning tree	$T = \left(V_{1} \in E \right)$
$= \forall \forall u_1 \forall e_1 \forall e_2 \forall u_1 \forall e_2 \forall e_1 \forall e_1 \forall e_2 \forall e_1 \forall e_2 \forall e_1 \forall e_2 \forall e_2 \forall e_1 \forall e_2 \forall e_$	graph with
Connected	no cycles a
$\tilde{\boldsymbol{\boldsymbol{\omega}}} = \tilde{\boldsymbol{\omega}} + \tilde$	· · · · · · · · · · · · · · · · · ·
	· · · · · · · · · · · · · · · · · ·

Minimum	Spanning Tree	· · · · · · · · · · · · · · · · · · ·
Given a a a a a a a a a a a a a a a a a a a	connected, graph ndirected, graph weighted	$ \begin{array}{c} \begin{array}{c} \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} $
Find a mini	mum spanning tre	$e T = (V, E' \in E)$
	$\begin{array}{cccccccccccccccccccccccccccccccccccc$. .
· · · · · · · · · · · · · · ·		
Mimmize	$\forall u, v \in \vee$,	graph with
Minimize Sum of edge weights.	YuveV, u and v connected	graph with no cycles
Minimize Sum of edge weights. $W_{T} = \sum W_{e}$	YuveV, u and v connected in T	graph with no cycles
Minimize Sum of edge weights. $W_T = \sum W_e$ eet	YuveV, u and v connected in T	graph with no cycles
Minimize Sum of edge weights. $W_T = \sum W_e$ $e \in T$	YuveV, u and v connected in T	Joraph with no cycles

 $\frac{1}{7}$ 773 | 1 | 1 | 1 | 1 31 7 $\frac{7}{2}$ 5

 $\frac{2}{2}$. 7 · · · ·

of HWQ Q1(d) Announcements * Canvas published (SORRY!) * KT added to CAMP. * OH start this afternoon L, Prof. Kim -> Gates 203, 1-3p L' See full schedule on course page. dedicated 4820 Space 24/7. Locations Rhodes 590 & Rhodes 655 lange room for peak OH

MST Algorithuns.	. .
Given a weighted	graph G = (V, E, W)
Refurn a minimum	spanning free.
Approach. Greedy	AlzoriHuns
. .	. .
	. .
. .	. .

	Greedy Algorithms																															
· ·	Design Pavadigm Choose solution "greedily"																															
· · ·	• •	•	· ·			• •		· · ·	· · ·	ly	op	ĩC		· ·	· · ·	10	c`¢	L	· · ·	de	с 1	si	O'N'	د د	-11	na	f.	· · · · · · · · · · · · · · · · · · ·	්ංර		7000	
· ·	· ·		· ·		•	Ţ	V./ (e V	10 0	ia.	6(L.	•		· · ·	Ö	n c	e M	 	Ŵ	e . 1 e .	N	12	ke	 	С. С. С. С.	d	ec	is Je		 ۲. <i>9</i> .	
· ·	• •	•	· ·	•	•	• •	•	• •	• •	•		· •	•	•	• •	•	•			•			· ·			•	•	· ·	• • •		• •	
· ·	• •	•	• •	•	•	• •	•	· ·	• •		• •	• •	•	•	• •	•	•	•	• •	•	• •	•	••••	•	• •	•	•	• •	•	• •	• •	· ·
• •	• •	•	· ·	•	•	• •	•	· ·	• • • •	•	• •	• •	•		••••	•	•	•	••••	•	••••	•	••••	•	• •	•	•	• •		• •	• •	• •
· ·	• •	•	• •	•		• •	•	• •	• •	•	• •	• •	•		••••	•	•		• •	•	· ·	•	• •	•	• •	•	•	• •		• •	• •	• •
· ·	• •	•	· ·	•		• •	•	••••	• •	•	• •	• •	•		• •	•	•		• •	•	· ·		••••	•	• •	•	•	• •	•	• •	· ·	· ·
• •	• •	•	•••	•	•	• •	•	• •	• •	•	• •	• •	•	•	••••	•	•	•	• •	•	•••	•	•••	•	• •	•	•	••••	•	• •	• •	• •
	• •		· ·	•		- • • •		· · ·	- • • •	•	• •	· ·	•		· · ·	•		•		•	· ·	•	••••	•		•	•	- · ·	•	- • • •	· ·	· · ·
• •	• •			•	•	• •	•	• •	° °	•	• •		•	•	••••	•	•	•	••••	•	• •		•••	•				• •	•			

Greedy Algorithms Choose solution greedily Design Powadigm. Myopic -> local decisions that "look good" Irrevocable -> once we make a decision, we never reconsider. Forst & Local Advantage. Runnice Time? Greedy Prototype (1) Sort by "priority" $O(n \log n)$ (2) Iterate through elems in priority order 4 + Make decision about elem."constant" O(1) time

Greedy Algorithms Choose solution greetily Design Pavadigm. Myopic - local decisions that "look good" Irrevocable -> once we make a decision, we never reconsider. Fast & Local Advantage. Greedy Prototype Design of Greedy Algorithm (2) Iterate through elems in priority order Ly Make decision about elem.

Greedy Algorithms
Design Pavadigm Choose solution "greetily"
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 ave incorrect.
1 1

Greedy Algorithms for MST Given graph G = (V, E, W) weights Greedy Prototype D Sort by "priority" e dojes (2) Iterate through elems in priority order Ly Make decision about elem.

Two MST Algs. On input G = (V, E, W)Step Ø Sout edges by weight $W_{e_1} < W_{e_2} < \cdots < W_{e_m}$

Two MST Algs. On input G = (V, E, W)Step Ø. Sout 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 Ø. Sort edges by weight $W_{e_1} < W_{e_2} < \cdots < W_{e_m}$ Delete Max Add Min $T = \left(\bigvee_{i} \sum_{j} \sum_{i} \sum_{j} \right)^{i}$ For $i = 1 \longrightarrow M$. For $i = M \longrightarrow 1$ if TVZeig does not form a cycle if T \ Zeig is still connected Add ez to T. Remove ez from T Return 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 emax be the edge in C of Maximum weight. Then emax is NOT in the MST of G. emox O

The Cycle Lemma, G=(V,E,W) w/ distinct edge weights. Suppose C is a cycle within G and let emax be the edge in C of Maximum weight. Then emax is NOT in the MST of G. emox of Corollary. Delete Max returns the MST of G. C7 KT Reverse Delete

Proof of Correctness of Delete Max (assuming Cycle Lemma) Delete Max $\widehat{\nabla (x)} = \widehat{\nabla (x)} + \widehat{\nabla (x)}$ For i = M -> 1 if T \ Zeig is still connected Remove ez from T Return T.

Proof of Correctness of Delete Max (assuming Cycle Leurna) By contradiction, Suppose the MST is T*, but Delete Max refurns T = T* \Rightarrow Delete Max deletes some edge $(u,v) \in T^*$. Delete Max $\overline{(1, 1)} = (1, 1)$ For $i = M \longrightarrow 1$ if T \ Zeig is still connected Remove ez from T Return T.

Proof of Correctness of Delete Max (assuming Cycle Leurie) By contradiction, Suppose the MST is T*, but Delete Max returns T = T* ⇒ Delete Max deletes some edge (u, v) ∈ T*. > When (n,v) is removed from T, Fa path P from U->V in T 7[0,0] Delete Max $\overline{\mathbf{r}} = \mathbf{r} = \mathbf{r} = (\mathbf{r} + \mathbf{r} + \mathbf{r}) \mathbf{r} = \mathbf{r}$ $For i = M \longrightarrow 1$. if T \ Zeig is still connected $T \setminus \{(u,v)\}$ Remove ez from T Return T.

(assumine Cycle Leurna) Proof of Correctness of Delete Max By contradiction, Suppose the MST is T*, but Delete Max returns T = T* \Rightarrow Delete Max deletes some edge $(u,v) \in T^{*}$. > When (u,v) is removed from T, Fa path p from U->V in T Delete Max $\overline{(1 + 1)} = 1 \quad (1 + 1) \quad (1 + 1)$ $For i = M \longrightarrow 1$. if T \ Zeig is still connected $T \setminus \{(u,v)\}$ Remove ez from T $\Rightarrow p \cup \{(u,v)\}$ is a cycle in G Return T.

Proof of Correctness of Delete Max (assumily Cycle Leuma) By contradiction, Suppose the MST is T*, but Delete Max returns T = 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 By removal order (greatest -> (east) $\overline{r} (r) = r (r) (r) (r) (r)$ (u,v) is the max wt. edge For $i = M \longrightarrow 1$. in cycle! if T \ Zeig is still connected Remove ez from T Return T.

Proof of Correctness of Delete Max (assuming Cycle Leurice) By contradiction, Suppose the MST is T*, but Delete Max returns T = T* ⇒ Delete Max deletes some edge (u,v) ∈ T*. $\Rightarrow P \cup \{(u,v)\}$ is a cycle in G But why did we remove (u,v)? Delete Max By removal order (greatest -> (east) $\frac{1}{2} \left(\frac{1}{2} \right) = \frac{1}{2} \left(\frac{1}{2} \sqrt{1} + \frac{1}{2} \right) \left(\frac{1}{2} \sqrt{1} + \frac{1}{2} \right) \left(\frac{1}{2} \sqrt{1} + \frac{1}{2} \sqrt{1} \right)$ (u,v) is the max wt. edge For $i = M \longrightarrow 1$. in cycle! if T \ Zeig is still connected ⇒ By the Cycle Lemma Remove ez from T $(u,v) \not\in \mathcal{T}^{\mathcal{K}}$ Return T. (A contradiction)

$S_0, +o$	complete proof of cor	vectness for Delete Max
	need to prove the	Cycle Lemma.
· · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·
· · · · · · · · ·		
· · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·
· · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·
		· · · · · · · · · · · · · · · · · · ·
	· · · · · · · · · · · · · · · · · · ·	
· · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·
· · · · · · · · ·	· ·	· · · · · · · · · · · · · · · · · · ·
· · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·
· · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·

So, to complete proof of correctness for Delete Max
we need to prove the Cycle Lemma.
· · · · · · · · · · · · · · · · · · ·
Assumptions
DG=(V,E,W) w/ district edge wts,
2) C is a cycle in G contained emax
3) emax is the max we edge on C
Conclusion
* emax is Not in MST.
Next. Exchange Argument

Pf. By exchange avgument. Suppose T is a spanning tree s.t. $e_{max} = (u,v) \in T$ We show that T is NOT the minimum spanning tree. Of the Conax of Conax . / . . . / . . .

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 emax from T.
· · · · · · · · · · · · · · · · · · ·
V O O
· ·

Pf By exchange argument. Suppose T is a spanning tree s.t. emax = (u,v e We show that T is NOT the minimum spanning tree. two pieces

Pf By exchange avgument. 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, emax is in some cycle C within G. > there exists some edge e'EC (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 emax = T \Zemax Z U Ze'S

Claims	. .	· · · · · · · · · ·	· · · · · · · · · · ·	. .
	has smaller	weight	then T	· · · · · · · · · ·
· · · · · · · · · · · ·	· · · · · · · · · · · · · · ·	· · · · · · · · · ·	· · · · · · · · · · ·	· · · · · · · · · ·
	· · · · · · · · · · · · · · ·			
$\begin{array}{cccc} & & & & \\ & & & & \\ & & & \\ & & & \\ & & & \\ & & & \\ & & & \\ & & & \\ & & & \\ $	is a tree	· · · · · · · · ·	· · · · · · · · · ·	· · · · · · · · ·
· · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · ·	· · · · · · · · · · ·	· · · · · · · · · ·
· · · · · · · · · · · ·	· · · · · · · · · · · · · · ·	· · · · · · · · · ·	· · · · · · · · · ·	· · · · · · · · · ·
$\begin{array}{c} \begin{array}{c} & & \\ & & \\ & & \\ \end{array} \end{array} \begin{array}{c} \\ \end{array} \end{array} \begin{array}{c} \\ \end{array} \begin{array}{c} \\ \end{array} \end{array} \begin{array}{c} \\ \end{array} \begin{array}{c} \\ \end{array} \end{array} \begin{array}{c} \\ \end{array} \end{array} \begin{array}{c} \\ \end{array} \end{array} \begin{array}{c} \\ \end{array} \begin{array}{c} \\ \end{array} \end{array} \end{array} \begin{array}{c} \\ \end{array} \end{array} \begin{array}{c} \\ \end{array} \end{array} \end{array} \end{array} \begin{array}{c} \\ \end{array} \end{array} \end{array} \end{array} \end{array} \begin{array}{c} \\ \end{array} $	is connected		spanning	tree)
· · · · · · · · · · · ·		· · · · · · · · ·	· · · · · · · · · · ·	· · · · · · · · · ·
· · · · · · · · · · ·		· · · · · · · · ·	· · · · · · · · · · ·	· · · · · · · · ·
· · · · · · · · · · ·	· · · · · · · · · · · · · ·	· · · · · · · · ·	· · · · · · · · · ·	· · · · · · · · ·

Claims
() T' has smaller weight than T
- Rmax is the max while edge on C G WT K WT
-exchanged for e'EC. by D.43.
(2) T is α $tree$
(3) T' is connected (i.e. a spanning tree)

Claims,
() T' has smaller weight than T
$\cdots = 1 \cdot 1$
2) T'is a tree T we put dependence 1 coulo
- Removing Rmax breaks this cycle.
3) T' is connected (i.e. a spanning tree)

Claims
() T' has smaller weight than T

(2) T^{1} is a tree
3) T' is connected (i.e. a spanning tree)
Driginal Twas spanning.
- Any path using Rmax Can use detour along C to go from u <> V.

Pf. By exchange argument. Suppose T is a spanning tree st. $e_{max} = (u, v) \in T$ We show that T is NOT the minimum spanning tree. Thus, T is a spanning tree w/ $W_T' < W_T$ T is NOT the MST. Contraction of the second seco