) 1 a	Jar	MM ON'	$\begin{array}{cccccccccccccccccccccccccccccccccccc$	More	Greed	y Algo	· · · · · ·	· · · · ·	· · · · ·	•
	2 lou Fi	n ish	 		starys 1	Anead	Avaly Sis	· · · · · ·	 · · · · · 	•
	An	NOUNC	cener	xts				 . .<	· · · · · ·	· · · · · ·	
								 	· · · · ·	 	•
· · · ·	· · ·	· · · · · ·	· · · · · · ·		· · · · · ·	· · · · · ·	· · · · · ·	
· · · ·	· ·	 	 	 	· · · · · · ·	 	 	 	 	
											•

Earliest Finish Time D Sort jobs by finish time 2) Schedule = 25 Iterate through jobs in sorted order j=1....n - if job j does not conflict w/ Schedule -> Schedule <- Schedule vZjJ Return Schedule.

Optimal schedule $O^{\ddagger} \left[\begin{bmatrix} s_1^{\ast}, f_1^{\ast} \end{bmatrix}, \begin{bmatrix} s_2^{\ast}, f_2^{\ast} \end{bmatrix}, \begin{bmatrix} s_2^{\ast}, f_2^{\ast} \end{bmatrix} \right)$ EFT schedule $S = \langle [S_1, f_1], [S_2, f_2], \dots, [S_k, f_k] \rangle$ 'IF EFT stays ahead, then EFT is optimal" Earliest Finish Time Lemma For all je Z1,..., KJ ⇒ S is also optimal.

Optimal schedule $O^{\ddagger} \left[\begin{bmatrix} s_1^{\ast}, f_1^{\ast} \end{bmatrix}, \begin{bmatrix} s_2^{\ast}, f_2^{\ast} \end{bmatrix}, \begin{bmatrix} s_2^{\ast}, f_2^{\ast} \end{bmatrix} \right)$ EFT schedule $S = \langle [s_1, f_1], [s_2, f_2], \dots, [s_k, f_k] \rangle$ 'IF EFT stays ahead, then EFT is optimal" Earliest Finish Time Lemma For all je Z1,..., KJ S is also optimal. "EFT stays ahead of O* Greedy Stays Ahead Lemma For all je ZI, ..., ISIG $f_1 \leq f_1 \leq f_1$

"EFT stays ahead of Ot Greedy Stays Ahead Lemma. For all $j \in \{1, \dots, |S|\} = f_j^*$ By induction on jobs added by EFT. EFT adds jobs by earliest finish time. Base Case.

"EFT stays ahead of Ot Greedy Stays Ahead Lemma. For all $j \in \mathbb{Z}_1, ..., |S| \mathcal{G} = f_j^*$ By induction on jobs added by EFT. EFT adds jobs by carliest finish time. Base Case. \implies [s_1, f_1] \in S chosen because $f_1 = \min_{i \in [n]} f_i$

"EFT stays ahead of Ot Greedy Stays Ahead Lemma. For all $j \in \{1, \dots, |S|\}$ $f_j \leq f_j^*$ By induction on jobs added by EFT. EFT adds jobs by contrest finish time. Base Case. \implies $[s_1, f_1] \in S$ chosen because $f_1 = \min_{i \in [n]} f_i$ $= = \sum_{n=1}^{\infty} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{i=1}^{n} \sum_{i=1}^{n}$

"EFT stays ahead of Ot Greedy Stays Ahead Lemma. For all $j \in [1, ..., |S|] f_1 \leq f_1^*$ Inductive Hypothesis. For all i < k, $f_i \leq f_i^*$ $B_{y} = I_{H}, \quad F_{k-r} \leq f_{k-1} \leq s_{k}$ By Iff tk-r · · · · · · · · ·

"EFT stays ahead of Ot Greedy Stays Ahead Lemma. For all je $[1, ..., |S|] f_1 \leq f_j^*$ Inductive Hypothesis. For all i < k, $f_i \leq f_i^*$ By IH, $f_{k-1} \leq f_{k-1} \leq s_k$ So $[s_{k}^{\star}, f_{k}^{\star}] \in O^{\star}$ does not conflict $\langle [s_{i}, f_{i}]^{--} - [s_{k-i}, f_{n-i}] \rangle$ $f_{k-1} = f_{k} = f_{k} = f_{k}$ <mark>`k-l</mark>

"EFT stays ahead of Ot Greedy Stays Ahead Lemma. For all $j \in [1, ..., |S|] f_j \leq f_j^{\dagger}$ Inductive Hypothesis. For all i < k, $f_i \leq f_i^*$ By IH, $f_{k-1} \leq f_{k-1}^* \leq s_k^*$ So [sk, fk] E O* does not conflict {[s, f,] --- [sk-1, fu-1]} But [skifk] is non-conflicting job of earliest finish fine. f_{k-1} f_{k-1} f_{k} K-(.

"EFT stays ahead of Ot Greedy Stays Ahead Lemma. For all $j \in [1, ..., |S|] f_j \leq f_j^{\dagger}$ Inductive Hypothesis. For all i < k, $f_i \leq f_i^*$ By IH, $f_{k-1} \leq f_{k-1}^* \leq s_k^*$ So [sk, fk] E O* does not conflict {[s, f,] --- [sk-1, fu-1]} But [skifie] is non-conflicting job of earliest finish fine.

"If EFT stays ahead, then EFT is optimal Earliest Finish Time Lemma For all je Z1,..., KJ s is also optimal. $\int_{-\infty}^{\infty} \frac{1}{2\pi} \int_{-\infty}^{\infty} \frac{1}{2\pi} \int_{$ Similar Argument. See KT Claim (4.3) S OR FIDRI

Announcemm	ts		· ·
* Enrollment	Cap lifteo		· ·
x + W + 1 due L > A + most	Thurs, 1 3 slip da	J: 59 PM J: 59 Sun,	$1 \cdot 1 \cdot$
		. .	
.
. 	

Greedy Graph Algorithms

Minimum	Spanning	Tree
Given a	connected, undirected, weighted	$G_{i} = \left(\begin{array}{c} V_{i} \\ V_{i} \\ \end{array} \right) \left(\begin{array}{c} V_{i} \\ \end{array} \right) \left(\left(\left(\begin{array}{c} V_{i} \\ \end{array} \right) \left(\left(\left($

Minimum Spanning Tree
Given a undirected, graph $G = (V, E, W)$ weighted
• A graph is connected if $Hu, v \in V$ J a path $p=(e_1, \dots, e_{ P })$ connecting $u \neq v$.
Not connected Connected

Minimum Spanning Tree
Given a undirected, graph $G = (V, E, W)$ weighted
· A graph is connected if tu, ve V
$F = (e_1, \dots, e_{ P })$ connecting $u \neq v$.
• A graph is undirected if $(u,v) \in E \iff (v,u) \in E$
That is, (u,v) and (v,u)
represent the same edge
· · · · · · · · · · · · · · · · · · ·

Minimum Spanning Tree
Given a undirected, graph G=(V, E, W) weighted
· A graph is connected if Hu, ve V
I a path $p=(e_1, \dots, e_{ P })$ connecting $u \neq v$.
• A graph is undirected if $(u,v) \in E \iff (v,u) \in E$
• Each edge $e \in E$ has an associated Nonnegative weight $w_e \ge 0$
$\mathcal{M}_{[\mathcal{M}_{l}\mathcal{N}_{l}]} = \mathcal{R}_{l} \qquad \qquad$
$\begin{array}{cccccccccccccccccccccccccccccccccccc$
· · · · · · · · · · · · · · · · · · ·

Minin	rum Spanning Tree
<u><u><u></u><u></u><u>u</u><u></u><u>u</u><u></u><u>u</u><u></u><u>u</u><u></u><u>u</u><u></u><u>u</u><u></u><u>u</u></u></u>	a undirected, graph G=(V, E, W) weighted
Find	a minimum spanning tree $T=(V, E' \in E)$
· · · · · · · · · ·	
· · · · · · · · · ·	· ·
· · · · · · · · · ·	
· · · · · · · · ·	· ·
· · · · · · · · · ·	
 	· ·

Minimum Spanning Tree
Given a undirected, graph G=(V, E, W) weighted
Find a minimum spanning tree $T=(V, E' \in E)$
Minimum
weight
$W_{T} = 2$ W_{e}

Minimum	Spanning	Tree	· · · · · · · · · · · · ·	· · · · ·
$\frac{1}{1}$	connected, undirected, weighted	$ \begin{array}{c} \begin{array}{c} \end{array} \\ \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} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \end{array} $	$ = \left(\begin{array}{c} 1 \\ 1 \\ 1 \\ 2 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\$	
Find a mi	nimum <u>Spann</u>	ing tree	$T = \left(V_{1} \in C \right)$	
Minimum total weight	connects U, v E V		. .	
$W_T = \sum_{e \in T} W_e$				· · · · ·
	· ·		. .	· · · · ·
				• • • •

Minimum	Spanning Tree	•
	connected, graph $G = (V, E, W)$ weighted	•
Find a mi	nimum spanning tree T= (V, E'EE)	•
		•
Minimum total weight	connects all graph with u,v EV no cycles	•
Minimum total weight $W_{T} = \sum_{i=1}^{N} W_{e}$	connects all graph with u,v EV no cycles	· · · ·
Minimum total weight $W_T = \sum W_e$ eet	connects all graph with u,veV no cycles	

 $\frac{2}{2}$. 7 · · · ·

. 3 · · · · · · · · · · $\mathcal{W}_{\mathsf{T}} \simeq \mathcal{W}_{\mathsf{T}} \simeq \mathcal{W}_{\mathsf{T}}$

 $\frac{1}{2}$.7 i · · 3 . .2. . .7. . /. /. **.** . . . · · · / · · · . . /. · [· · · · ·

. . . · · · · <u>3</u>· . . / . . . $W_{\mp} = 12$

. 	· ·	· ·	· ·	. .
Given	a tree	h h h h h h	do ve lenow	that
. 	$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	is not a	Min Spannily	$ \frac{1}{1} + 1$
· · · · · · ·	· · · · · · · · · · ·			
	
 	· · · · · · · · · · ·
· · · · · · ·	· · · · · · · · · · ·	· · · · · · · · · · · ·		· · · · · · · · · · ·

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

Cut Lemma.	Given a graph, $G = (V, E, W)$.
	Simplifying Assumption Today wights are distinct.
· · · · · · · · · · · · · · · · · · ·	$\frac{1}{4} = \frac{1}{4} = \frac{1}$
. .	
. .	. .
· ·	. .
· · · · · · · · · · · · · · · · · · ·	<pre></pre>

Cut Lemma. Given a graph, G=(V, E, W). Consider any non-empty cut (S, V1S). > partition of vertices into two sets $C \subseteq V \land V \land S \subseteq$

Given a graph, G = (V, E, W). Cut Lemma. Consider any non-empty cut (S, VIS). Denote by est the minimum-weight edge crossing the cut.

Given a graph, G = (V, E, W), Cut Lemma Consider any non-empty cut (S, VIS). Denote by est the minimum-weight edge crossing the cut. The MST of G contains es.

Significance	of Cut Len	mma	•
Given T= ($\bigvee_{i} \stackrel{i}{\in} \left(\stackrel{i}{\leftarrow} \right)^{i} \stackrel{i}{\leftarrow} \left(\stackrel{i}{\leftarrow} \left(\stackrel{i}{\leftarrow} \right)^{i} \stackrel{i}{\leftarrow} \left(\stackrel{i}{\leftarrow} \left(i$		•
if there	exists some	$cut (S, V \setminus S)$	•
and T	Joes not	include est, then	•
· · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · ·	. .	•
· · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·	•
· · · · · · · · · · · · · · · · · · ·			•
· · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·	
· · · · · · · · · · · · · · · · · ·		· ·	•
· · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · ·	. .	•
· · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·	

Significance of Cut Lemma? Given T = (V, E')if there exists some cut (S,VIS) and T does not include es, then T is not an MST

Significance of Cut Lemma? Given T = (V, E')if there exists some cut (S, VIS) and T does not include es, then T is not an MST So, any algorithm for solving MST unst "collect" es for every non-empty (S,V\S)

Proof of Cut Lemma. By exchange avgument. Start with a spanning tree 10 that does not contain some es.

Proof of Cut Lemma. By exchange argument. Start with a spanning tree To that does not contain some es. Show how to exchange est for an edge e in To s.t. the tree weight drops.

Proof of Cut Lemma. By exchange argument. Start with a spanning tree To that does not contain some est. Show how to exchange est for an edge e in To s.t. the tree weight drops. To cannot be MST.

. 0 $\left(\begin{array}{c} & & \\ &$ (by assumption By defn. es connects some UES, VE To is a tree \implies connected. Idea O. ⇒ must be some other edge from S→VIS

e e to crosses (S, VS) $e_{s} \neq t_{o}$ (by assumption By defn. est connects some UES, VEVIS Idea 0. To is a tree => connected. \implies must be some other edge from $S \longrightarrow V \setminus S$

Idea O.	Exchange	est for	
	$ \begin{array}{cccccccccccccccccccccccccccccccccccc$		$ \begin{array}{cccccccccccccccccccccccccccccccccccc$
	$\left(\begin{array}{cccccccccccccccccccccccccccccccccccc$		· ·
$\begin{array}{cccccccccccccccccccccccccccccccccccc$. .
· · · · · · · · · · · · · · · · · · ·	. .	· · · · · · · · · · · ·	. .
.

Idea O. Exchange est for	
$\begin{array}{c} \begin{array}{c} \end{array}$	
$ \begin{array}{c} \begin{array}{c} \end{array} \\ \end{array} $. .
. .	. .
· · · · · · · · · · · · · · · · · · ·	

Idea O. Exchange est for eo la De la serve Total weight: We* - Weo + WT. By defin. Wets < We. > Total weight decreases.

Are	we done?	· · · · · · · ·	· · · · · · · · · · ·	· · · · · · · · · · · · · · · ·
	Exchanging preserve	est fo	r eo migh Structure	f = n + i + i + i + i + i + i + i + i + i +
		O_{1}^{1}		
				. .
	$e_{s} = e_{s} + e_{s$
· · · · · · · ·		· · · · · · · ·		

Are we done? No Exchanging es for eo might not preserve tree structure. est ∉ To disconnected & contains cycle

Idea 1	Find a path through To connecting ends of est.
	· · · · · · · · · · · · · · · · · · ·
· · · · · · · · · · · · ·	
· · · · · · · · · · · · ·	
· · · · · · · · · · · · ·	
· · · · · · · · · · · · · ·	
· · · · · · · · · · · · ·	$ \begin{array}{c} & & & \\ & $
· · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·
	· · · · · · · · · · · · · · · · · · ·
	· · · · · · · · · · · · · · · · · · ·
· · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·

Idea 1	Find a path through To connecting ends of est.
	S S S S S S S S S S S S S S S S S S S
. .	Normal Arrived Arri
· · · · · · · · · · · · ·	

$\frac{1}{2} \frac{1}{2} \frac{1}$	Find a path through To connecting ends of est.
	add es to form a cycle L) New graph still connected.

Idea		Conne	a pat cting	h thr ends	orgh	est.	· · · ·	· · · · ·
. .	· · · · · · · ·				· · · · · · · · · · · · · · · · · · ·		· · · ·	· · · · ·
	· · · · · · · ·						 	
· · · · · · · · ·					· · · · · · ·			
Remove	edge		n Fhe	cycle,	where	$W_{e_{s}}^{e_{s}}$		We

Idea 1 Find a path through To connecting ends of est. Remove edge e in the cycle, where West < We Ly Weight goes down > Removing edge from cycle cannot disconnect graph. ⇒ free