
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 = \emptyset

Iterate through jobs in sorted order \( j=1 \ldots n \)
   - if job \( j \) does not conflict w/ Schedule
     \( \rightarrow \) Schedule \( \leftarrow \) Schedule \( \cup \{ j \} \)

Return Schedule.
Optimal schedule \( O^* = \langle [s_1^*, f_1^*], [s_2^*, f_2^*], \ldots, [s_k^*, f_k^*] \rangle \)

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 \( j \in 1, \ldots, k \) \( f_j \leq f_j^* \) \( \Rightarrow \) \( S \) is also optimal.
Optimal schedule $O^* = \langle [s_1^*, f_1^*], [s_2^*, f_2^*], \ldots, [s_k^*, f_k^*] \rangle$

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

\[ f_j \leq f_j^* \Rightarrow S \text{ is also optimal.} \]

"EFT stays ahead of $O^*$"

**Greedy Stays Ahead Lemma**

For all $j \in \{1, \ldots, |S|\}$

\[ f_j \leq f_j^* \]
"EFT stays ahead of O*"

Greedy Stays Ahead Lemma.

For all $j \in \{1, \ldots, l\} \in G$, $f_j \leq f_j^*$

By induction on jobs added by EFT.

Base Case. EFT adds jobs by earliest finish time.
"EFT stays ahead of O*"

Greedy Stays Ahead Lemma.

For all $j \in \mathbb{E}, \ldots, |S| \subseteq \mathbb{E}$

\[ f_j \leq f_j^* \]

By induction on jobs added by EFT.

Base Case. EFT adds jobs by earliest finish time.

\[ [s_1, f_1] \in S \text{ chosen because } f_1 = \min_{i \in [n]} f_i \]
"EFT stays ahead of $O^*$".

**Greedy Stays Ahead Lemma.**

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

By induction on jobs added by EFT.

**Base Case.** EFT adds jobs by earliest finish time.

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

$\Rightarrow f_1 \leq f_1^*$
"EFT stays ahead of $O^*$".

Greedy Stays Ahead Lemma.

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

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

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

By IH

$\implies f_{k-1} < S_k^*$
"EFT stays ahead of $O^*$"

**Greedy Stays Ahead Lemma.**

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

---

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

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

So $[s_k^*, f_k^*] \in O^*$ does not conflict $\langle [s_i^*, f_i] : \ldots : [s_{k-1}^*, f_{k-1}] \rangle$
"EFT stays ahead of $O^*$.

**Greedy Stays Ahead Lemma.**

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

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

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

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

But $[s_k, f_k]$ is non-conflicting job of earliest finish time.
"EFT stays ahead of O*."

Greedy Stays Ahead Lemma.

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

Inductive Hypothesis. For all \( i < k \), \( f_i \leq f_i^* \)

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

So \( [s_k^*, f_k^*] \in O^* \) does not conflict \( \langle E[1, f_1], \ldots, E[s_{k-1}, f_{k-1}] \rangle \).

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

\[ \implies f_k \leq f_k^* \]
“If EFT stays ahead, then EFT is optimal”

Earliest Finish Time Lemma

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

$\text{f}_j \leq \text{f}^*_j$ \implies $S$ is also 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 for every $u, v \in V$, there exists a path $p = (e_1, \ldots, e_{|p|})$ connecting $u$ and $v$.

Not connected

Connected
Minimum Spanning Tree

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

- A graph is connected if $\forall u, v \in V$ there exists a path $p = (e_1, \ldots, e_{|p|})$ connecting $u$ and $v$.
- 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 connected, undirected, weighted graph $G = (V, E, W)$

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

\[ \text{O} \quad w_{(u,v)} = 8 \quad \text{O} \]

\[ u \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad v \]
Minimum Spanning Tree

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

Find a minimum spanning tree $T = (V, E', \leq \epsilon)$
Minimum Spanning Tree

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

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

Minimum total weight

\[ w_T = \sum_{ee \in T} w_e \]
Minimum Spanning Tree

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

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

- minimum total weight
  
  \[ w_T = \sum_{e \in T} w_e \]

- connects all $u, v \in V$
Minimum Spanning Tree

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

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

- minimum total weight
- connects all \( u, v \in V \)
- graph with no cycles

\[ w_T = \sum_{e \in T} w_e \]
$W_T > 11$
Given a tree $T$, how do we know that $T$ is/is not a min spanning tree?
Cut Lemma. Given a graph, $G = (V, E, W)$. 

[Diagram of a graph $G$]
Cut Lemma. Given a graph, $G = (V, E, W)$.

Simplifying Assumption

Edge weights are distinct.

$\forall e, e' \in E, \; w_e \neq w_{e'}$
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 \( e^* \) the minimum-weight edge crossing the cut.

The MST of \( G \) contains \( e^* \).
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 $T_o$ that does not contain some $e^*_o$. Show how to exchange $e^*_o$ for an edge $e$ in $T_o$ 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 $T_0$ such that the tree weight drops.

$\Rightarrow T_0$ cannot be MST.
\[ G = \begin{array}{c}
\vspace{0.5cm}
\end{array} \]

\[ T_0 \]

By defn. \( e_0^* \) connects some \( u \in S \), \( v \in V \setminus S \).

Idea 0. \( T_0 \) is a tree \( \Rightarrow \) spanning connected. \( \Rightarrow \) must be some other edge from \( S \rightarrow V \setminus S \).
\[ G = \]

\[ T_0 \]

By defn. \( e_s^* \) connects some \( u \in S, v \in V \setminus S \)

Idea 0. \( T_0 \) is a tree \( \Rightarrow \) connected.

\[ \Rightarrow \] must be some other edge from \( S \to V \setminus S \)

\[ e_0 \in T_0 \text{ crosses } (S, V \setminus S) \]

\[ e_s^* \notin T_0 \text{ (by assumption)} \]
Idea 0. Exchange $e_s^*$ for $e_0$. 
Idea 0. Exchange $e_s^*$ for $e_0$. 

$e_0 \in T_0$

$e_s^* \notin T_0$
Idea 0. Exchange $e_s^*$ for $e_0$.

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

By defn. $W_{e_s^*} < W_{e_0} \implies$ Total weight decreases!
Are we done?

No. Exchanging $e_s^*$ for $e_0$ might not preserve tree structure.
Are we done?

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

$e_0$

$e_s^* \notin T_0$

Disconnected & contains cycle.
Idea 1. Find a path through $T_0$ connecting ends of $e_0^*$. 

$e_0$, $o$, $o$, $e_0^* \in T_0$. 

$u$, $e_0^* \in T_0$. 

Diagram with nodes and edges connecting them.
Idea 1: Find a path through $T_0$ connecting ends of $e_s^*$. $e_0$ $e_s^* \notin T_0$
Idea 1. Find a path through $T_C$ connecting ends of $e_s^\ast$.

add $e_s^\ast$ to form a cycle

$\Rightarrow$ 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 $W_{e_s^*} < W_e$. 

$\Rightarrow$ Weight goes down.

$\Rightarrow$ Removing edge from cycle cannot disconnect graph. 

$\Rightarrow$ tree.