

CS 331, Fall 2026

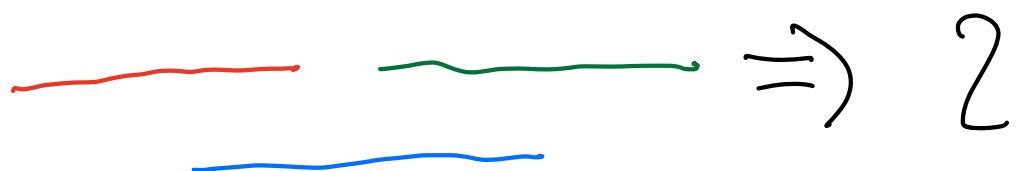
Lecture 9 (9/23)

Today: - Scheduling revisited
- Balpar revisited
- Trees
- MST

Scheduling revisited (Part IV, Section 4.1)

Input: n intervals $[l_i, r_i] \subset \mathbb{R}$
start end

Output: max # non-overlapping

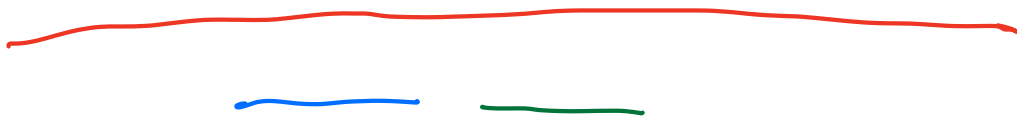
e.g.  $\Rightarrow 2$

Claim: greedy suffices. What rule?

Shortest?



First-come-first-serve?



Fewest overlaps?



Earliest end time? Works!!!

General argument: "greedy stays ahead"

Greedy Stays ahead

Think of **OPT** and **ALG** as making seq. of choices.

1) Declare invariant: after k choices by **ALG**, invariant holds compared to **OPT** (proof: by induction)

2) Prove invariant at the end contradicts

$$f(\text{OPT}) > f(\text{ALG})$$

Greedy Scheduling (L):

Sort L by right endpoint $// r_1 \leq \dots \leq r_n$

$(\text{count}, i) \leftarrow (1, 1)$

For $2 \leq j \leq n$:

If $l_j > r_i$:

$(\text{count}, i) \leftarrow (\text{count} + 1, j)$ $//$ Include j
as soon as
you can

Return count

1) Declare invariant

$$\text{let } \text{ALG} = \{a_1, a_2, \dots, a_k\}$$

$$\text{OPT} = \{o_1, o_2, \dots, o_k, \dots, o_l\}$$

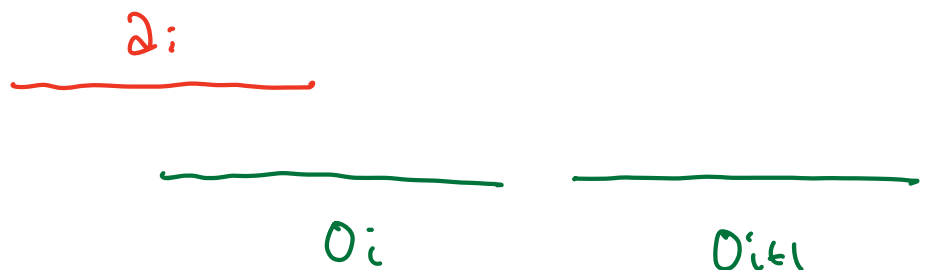
Greedy stays ahead: $a_i \leq o_i \quad \forall i \in [k]$

Proof:

↑
earlier interval

Base case: $a_1 = 1$

Induction: Assume $a_i \leq o_i$



Then $r_{a_i} \leq r_{o_i} < l_{o_{i+1}}$

So o_{i+1} available. We take first available, so $a_{i+1} \leq o_{i+1}$

2) Obtain Contradiction

Consider termination.

$$\{ \color{red}{a_1}, \color{red}{a_2}, \dots, \color{red}{a_k} \}$$

$\wedge \quad \wedge \quad \wedge$

$$\{ \color{green}{o_1}, \color{green}{o_2}, \dots, \color{green}{o_k}, \color{green}{o_{k+1}} \}$$

$$\frac{\color{red}{a_k}}{\color{green}{o_k} \quad \color{green}{o_{k+1}}}$$

We could include $\color{green}{o_{k+1}}$! $\Rightarrow \Leftarrow$

BalPar revisited (Part IV, Section 4.2)

Input: S , string of $(,)$

length n , even

Output: Minimum # of flips
to become balanced

generalizes prev.

flips = 0?

True/False

Exs.

$() () ((((,) () (() (($

Define $bal(S) = \# \left(\underset{\substack{\uparrow \\ \text{paren string}}}{\text{in } S} - \# \right) \text{ in } S$
 $\quad \quad \quad = +1 \quad \quad \quad = -1$

Claim: If S is balanced, then for all $i \in [n]$,
 $bal(S[:i]) \geq 0, \quad bal(S[i:]) \leq 0$
 $\quad \quad \quad \underbrace{\hspace{1cm}}_{\text{prefix}} \quad \quad \quad \underbrace{\hspace{1cm}}_{\text{suffix}} \quad \quad \quad \uparrow \text{ } |S|$

Proof (prefix): If $bal(S[:i]) < 0$,
 more $)$ than $($ in prefix, some $)$ unmatched $\Rightarrow \Leftarrow$

Proof (suffix): Similar, some unmatched $($

Say a string S is feasible if

$$bal(S[:i]) \geq 0 \text{ for all } i \in [n]$$

- Idea:
- Greedily flip to maintain feasibility.
 - Fix if $bal \neq 0$ at the end.

Bal For Min Flips (S):

$(\text{flips}, \text{bal}) \leftarrow (0, 0)$

For $i \in [n]$:

If $S[i] == ($: $\text{bal} \leftarrow \text{bal} + 1$ // Case 1

If $S[i] ==)$ AND $\text{bal} > 0$: // Case 2

$\text{bal} \leftarrow \text{bal} - 1$

If $S[i] ==)$ AND $\text{bal} = 0$: // Case 3

$(\text{flips}, \text{bal}) \leftarrow (\text{flips} + 1, \text{bal} + 1)$

// flip $S[i]$ to ensure feasibility

Return $\text{flips} + \frac{1}{2} \cdot \text{bal}$

// fix extraneous $($ left over

Example

) ())) (((

1 2 1 0 1 2 3 4
↑
flips
bal

flip 2 more at end

Invariant: let $ALG_i =$ flips used in $S[:i]$

let $OPT_i =$ fewest flips needed for $S[:i] \rightarrow$ feasible

Then, $ALG_i = OPT_i$ for all $i \in [n]$

Proof: Induction. $i=1$: Must flip if)

Assume $ALG_i = OPT_i$.

Case 1 or 2: $ALG_{i+1} = ALG_i$ (no flip)

$$OPT_{i+1} \geq OPT_i \quad \checkmark$$

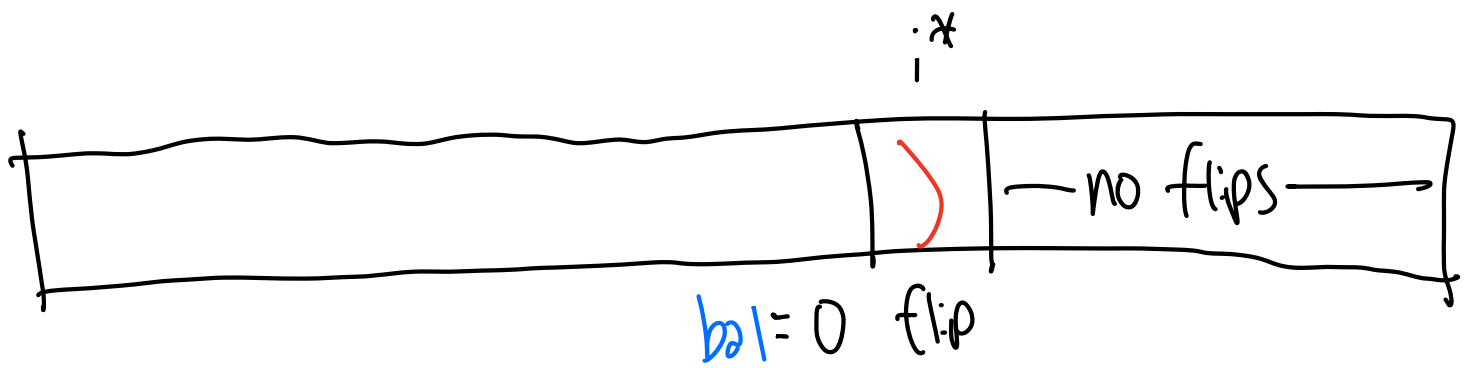
Case 3: Claim: Any algo using OPT_i flips
must flip $S(i+1)$

Proof: — $S[:i]$ —)

only depends on #flips $\rightarrow bal = 0 \quad bal = -1$

Hence, $OPT_{i+1} = OPT_i + 1 = ALG_{i+1} \quad \checkmark$

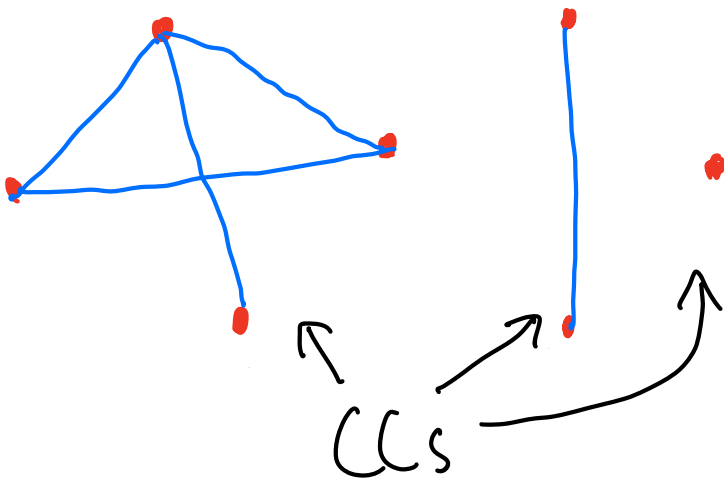
To conclude proof, let $i^* = \text{last flip}$



- Any algo must
- Use **AUG** i^* flips
 - fix $S[i^* :]$ to have **bal** ≤ 0

Trees (Part I, Section 4.2)

Undirected graphs only today.



Connected = $\exists$ path

connected components

(CCs) = partition vertices
into maximal connected pieces

Forest: graph w/ no cycles.

Tree: connected forest.

Key fact 1: Forests w/ k CCs
have $n - k$ edges

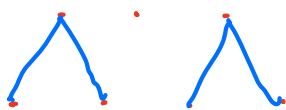


CCs:

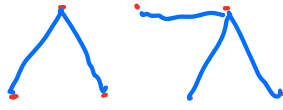
n

$n-1$

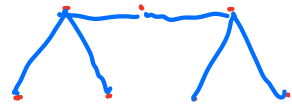
$n-2$



3

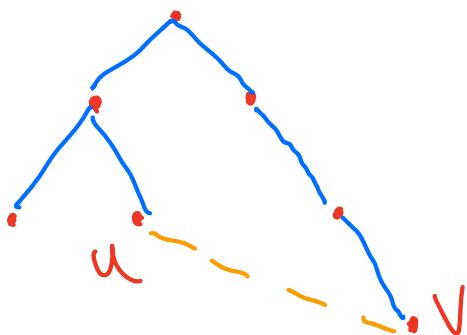


2



1

Key fact 2: off-tree edges make unique cycles



There is a unique existing
 $u \rightarrow v$ path in the tree.

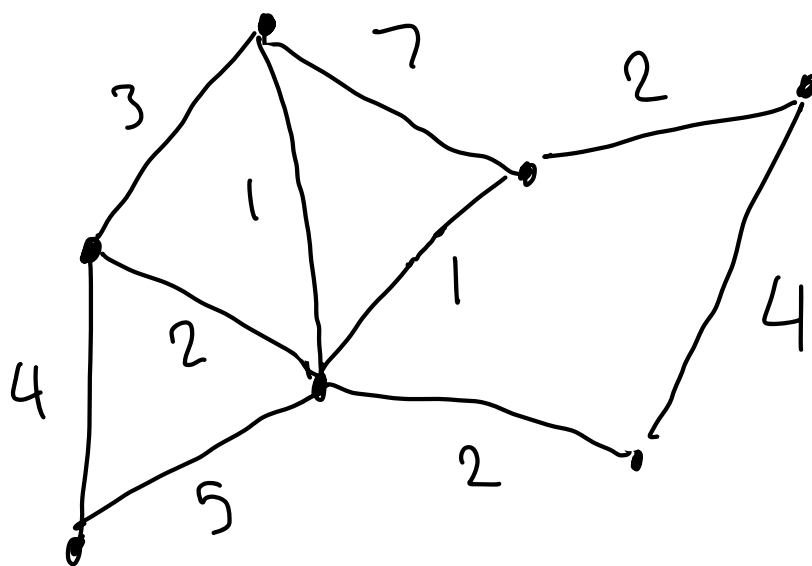
Minimum Spanning tree (Part IV, Section 4.3)

Input: (V, E, w) Connected

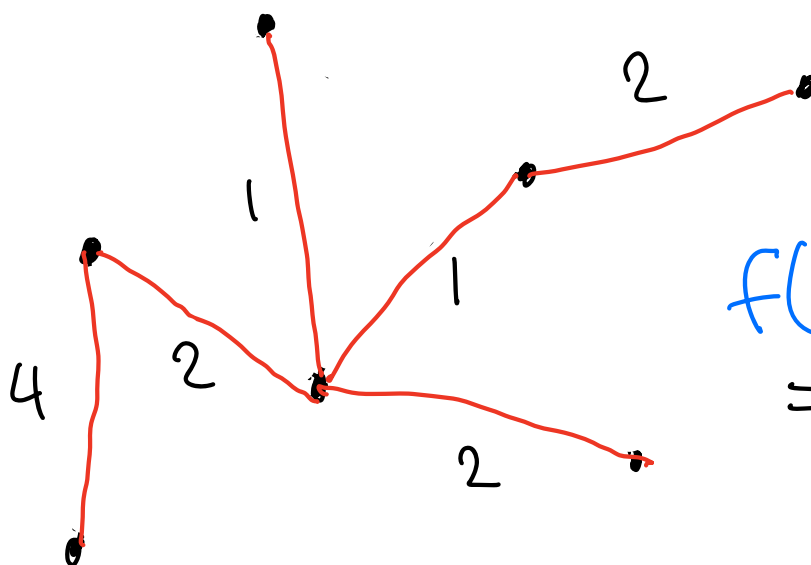
Output: $T \subseteq E$ a tree

Minimizing $f(T) := \sum_{e \in T} w_e$

Example



MST:



$$f(\text{MST}) = 12$$

How to greedy?

- Pref low weight
- Form no cycle

MST Conceptual (V, E, w) :

Sort E by weight $// w_1 \leq w_2 \leq \dots \leq w_{|E|}$

$T \leftarrow \emptyset$

For $e \in E$:

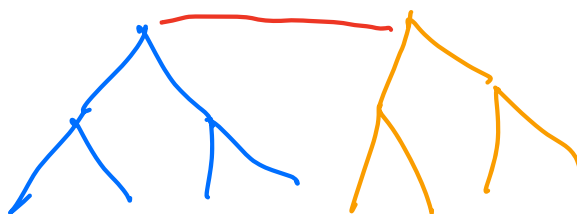
If $T \cup \{e\}$ contains no cycle:

$T \leftarrow T \cup \{e\}$

Return $\sum_{e \in T} w_e$

Does it return a tree?

Yes:



Suppose ≥ 2 CC's at termination.

There's Δ between edge (graph connected)

Doesn't form a cycle. Should have included!

Exchange Lemma

Suppose F, F' forests, $|F| < |F'|$

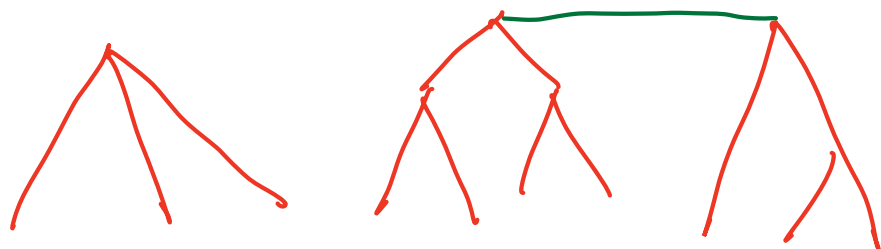
There's some $e \in F'$ s.t. $F \cup \{e\}$ is forest

Proof: F has k CCs

F' has k' CCs

Some edge of F' between CCs

or $k' \geq k \Rightarrow |F'| \leq |F|$



Greedy stays ahead + exchange

$$ALG = \{a_1, a_2, \dots, a_{|V|-1}\}$$

Stronger claim: after adding k edges,

$$ALG_k = \{a_1, a_2, \dots, a_k\}$$

doing better than any forest

$$OPT_k = \{o_1, o_2, \dots, o_k\}$$

$$\text{Greedy stays ahead: } \sum_{i \in (k)} w_{a_i} \leq \sum_{i \in (k)} w_{o_i}$$

Let first violation be after k edges.

Exchange: $ALG_{k-1} \cup \{o_k\}$ is forest

Case 1: $w_{o_k} \geq w_{a_k}$

$$\sum_{i \in [k]} w_{a_i} = \sum_{i \in [k-1]} w_{a_i} + w_{a_k}$$

$$\leq \sum_{i \in [k-1]} w_{o_i} + w_{o_k} = \sum_{i \in [k]} w_{o_i}$$

$\Rightarrow \Leftarrow$ (greedy still ahead!)

Case 2: $w_{o_k} < w_{a_k}$

Recall $ALG_{k-1} \cup \{o_k\}$ is forest

We should have taken o_k , it's earlier.

MST is optimal! Take $k = |V| - 1$

Implementation (Kruskal)

let $m := |E|$ $n := |V|$

MST(G):

Sort E by weight $O(m \log(n))$

For $i \in [n]$:

$C(i) \leftarrow i$

$S_i \leftarrow \{i\}$

} $O(n)$

For $(u,v) \in E$:

If $C(u) \neq C(v)$:

$T \leftarrow T \cup \{(u,v)\}$

Merge $S_{C(u)}, S_{C(v)}$

} $O(m)$

???

Return $\sum_{e \in T} w_e$

Cost of merging:

$$\text{let } |S_{\text{left}}| \leq |S_{\text{right}}|$$

For $w \in S_{\text{left}}$:

- Update $C(w) \leftarrow C(v)$
 - Add w to S_{left}
- $\left. \begin{array}{l} \text{Update } C(w) \leftarrow C(v) \\ \text{Add } w \text{ to } S_{\text{left}} \end{array} \right\} O(|S_{\text{left}}|)$

Every w on smaller side $O(\log(n)) \times$

$$\sum_{v \in V} \text{cost}(v) = O(n \log(n))$$

- Improvements:
- Boruvka: Parallel $O(\log(n))$
 - KKT: randomized $O(m)$
 - Pettie-Ramachandran: @UT!

Optimal in comparison model