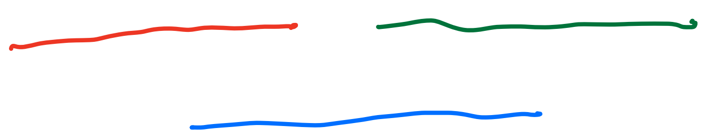
CS 331, Fall 2024
Lecture 9 (9/25)

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

Scheduling revisited (Part IV, Section 4.1)

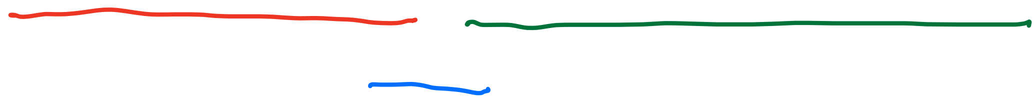
Input: n intervals $[l_i, r_i] \subset \mathbb{R}$
start end

Output: max # non-overlapping

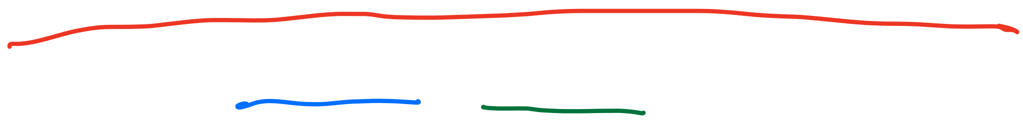
e.g.

$\Rightarrow 2$

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

$$f(OPT) > f(ALG)$$

```
GreedyScheduling(L):
    Sort L by right endpoint    // r_1 ≤ ... ≤ r_n
    (count, i) ⇐ (1, 1)
    For 2 ≤ j ≤ n:
        If l_j > r_i:
            (count, i) ⇐ (count + 1, j)    // Include j as soon as you can
    Return count
```

1) Declare invariant

Let $ALG = \{a_1, a_2, \ldots, a_k\}$

$OPT = \{o_1, o_2, \ldots, o_k, \ldots, o_\ell\}$

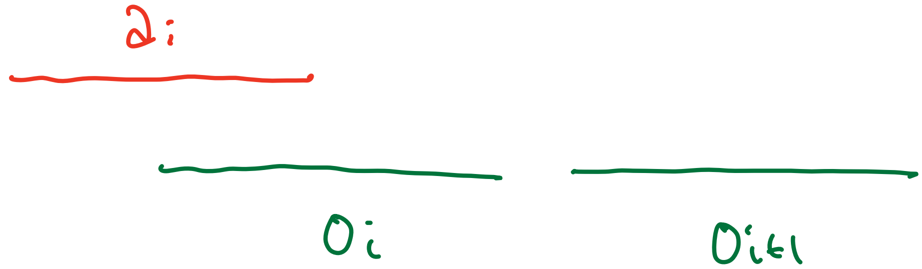
Greedy stays ahead: $a_i \leq o_i \quad \forall i \in [k]$

earlier interval

Proof:

Base case: $a_1 = 1$

Induction: Assume $a_i \leq o_i$

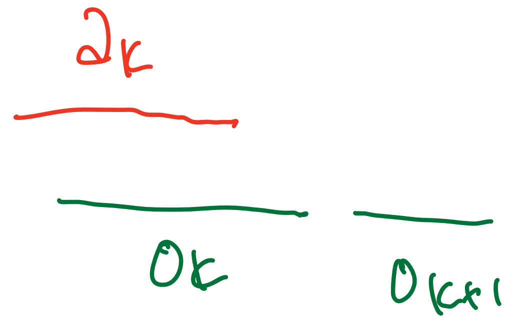


Then $r_{a_i} \leq r_{o_i} < l_{o_{i+1}}$

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

2) Obtain Contradiction

Consider termination.



$$\{a_1, a_2, \ldots, a_k\}$$
$$\wedge\!\!\backslash \quad \wedge\!\!\backslash \qquad \wedge\!\!\backslash$$
$$\{o_1, o_2, \ldots, o_k, o_{k+1}\}$$

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

BalPar revisited (Part IV, Section 4.2)

Input: $\underbrace{S}_{\text{length } n,\ n \text{ even}}$, string of (,)

Output: Minimum # of flips to become balanced

generalizes prev. flips = 0? True/False

$\boxed{\text{Exs.}}$ ()()((((,)()(()((

Define bal(S) = #(in S − #) in S

$\text{bal}(\underset{\text{paren string}}{S}) = \underset{=+1}{\#(} \text{ in } S - \underset{=-1}{\#)} \text{ in } S$

Claim: If S is balanced, then for all $i \in [\underset{|S|}{n}]$,

$\text{bal}(\underbrace{S[:i]}_{\text{prefix}}) \geq 0$, $\text{bal}(\underbrace{S[i:]}_{\text{suffix}}) \leq 0$

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

Proof (suffix): Similar, some unmatched (

Say a string S is <u>feasible</u> if

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

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

BalParMinFlips(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

$\boxed{\text{Example}}$

)	(	)	)	)	(	(	(	
1				2				flips
1	2	1	0	1	2	3	4	bal

flip 2 more at end

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

Let OPT_i = fewest flips needed for $S[:i] \to$ 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)

$OPT_{i+1} \geq OPT_i$ ✓

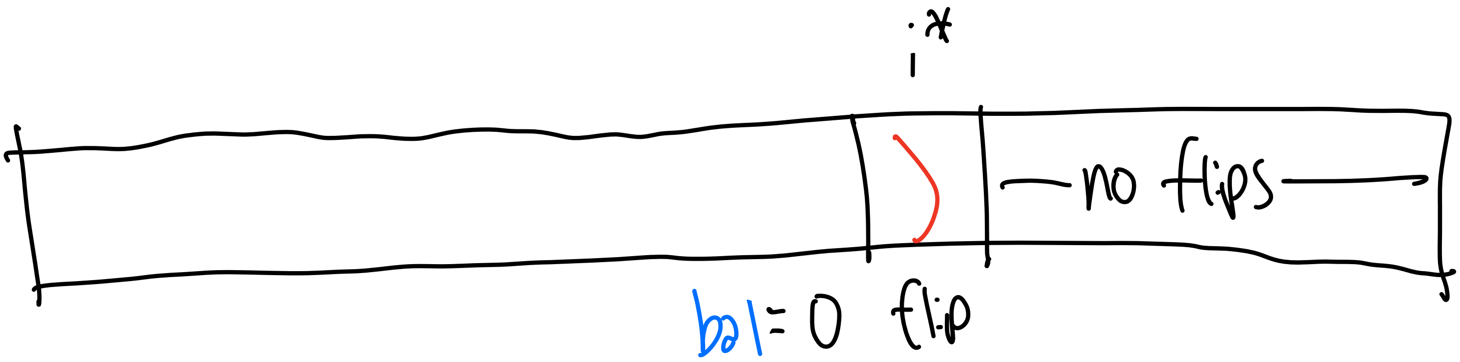
Case 3: Claim: Any algo using OPT_i flips must flip $S[i+1]$

Proof: —$S[:i]$——)

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

Hence, $OPT_{i+1} = OPT_i + 1 = ALG_{i+1}$ ✓

To conclude proof, let i^* = last flip

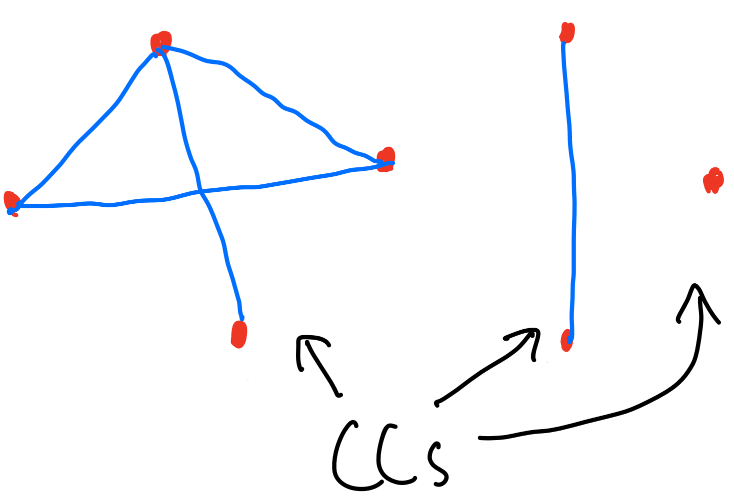


Any algo must

- use ALG_{i^*} flips
- fix $S[i^*:]$ to have bal ≤ 0

Trees (Part I, Section 4.2)

Undirected graphs only today.



Connected = $\exists$ path

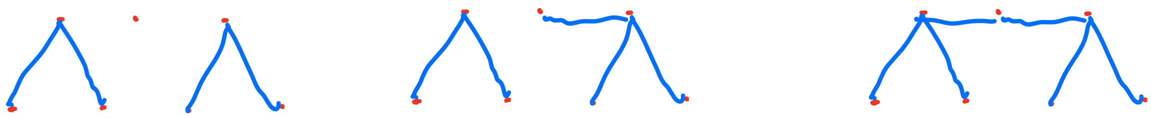
connected components (CCs) = partition vertices into maximal connected pieces

Forest: graph w/ no cycles.

Tree: connected forest.

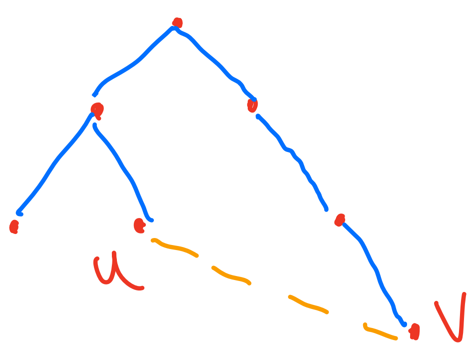
Key fact 1: Forests w/ k CCs have $n-k$ edges

CCs: n $n-1$ $n-2$



3 2 1

Key fact 2: off-tree edges make unique cycles



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

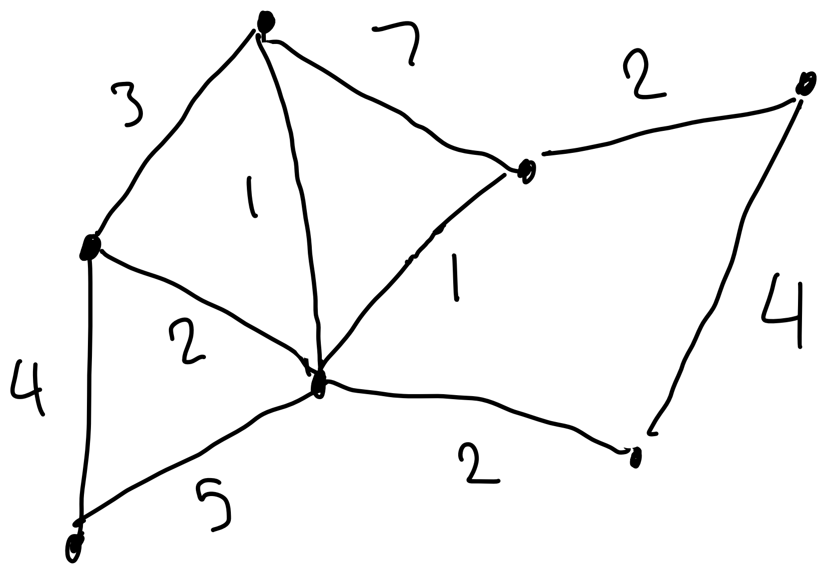
Minimum Spanning tree (Part IV, Section 4.3)

Input: (V, E, w) Connected

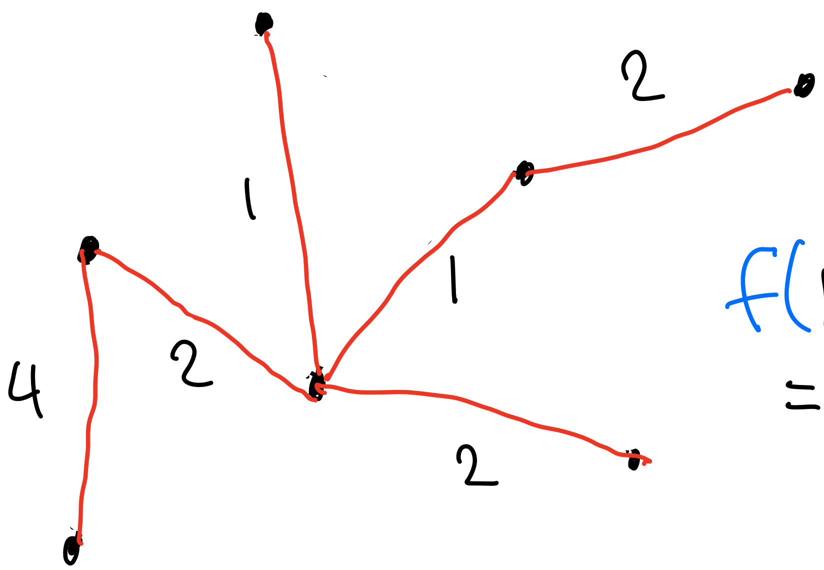
Output: $T \subseteq E$ a tree

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

Example



MST:



$f(MST) = 12$

How to greedy?

- Pref low weight
- Form no cycle

MST Conceptual(V, E, w):

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

$T \leftarrow \emptyset$

For $e \in E$:

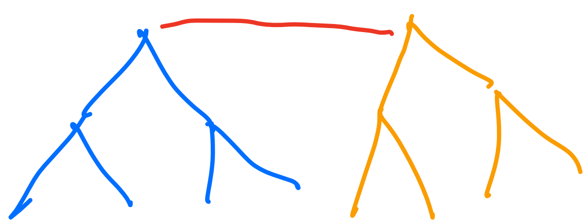
If $T \cup \{e\}$ contains no cycle:

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

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

Does it return a tree?

Yes:



Suppose ≥ 2 CC's at termination.

There's a between edge (graph connected)

Doesn't form a cycle. Should have included!

$\boxed{\text{Exchange lemma}}$

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

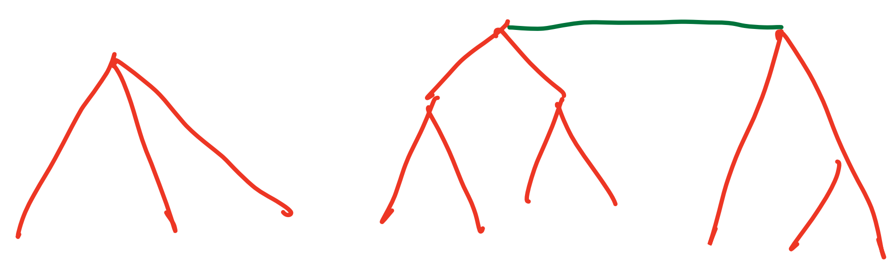
There's some $e \in F'$ s.t. $F \cup \{e\}$ is forest

Proof: F has k CCs

F' has k' CCs

Some edge of F' between CCs

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



$\boxed{\text{Greedy stays ahead + exchange}}$

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

Stronger claim: after adding k edges,

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

doing better than any forest

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

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 $m := |E|$ $\quad n := |V|$

MST(G):

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

    For $i \in [n]$:

        $C[i] \leftarrow i$

        $S_i \leftarrow \{i\}$ $\quad O(n)$

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

        If $C[u] \neq C[v]$:

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

            Merge $S_{C[u]}, S_{C[v]}$ $\quad$???

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

Cost of merging:

Let $|S_{C[u]}| \leq |S_{C[v]}|$

For $w \in S_{C[u]}$:

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

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

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

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

  Optimal in comparison model