CS 331, Fall 2024

Lecture 9 (9/25)

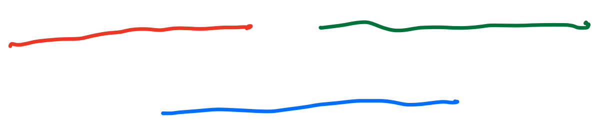
Today:
- Scheduling revisited
- Trees
- MST

Scheduling revisited (Part IV, Section 3.3)

Same problem as in DP.

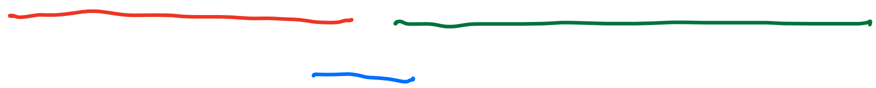
Input: n intervals $[\underset{\text{start}}{l_i}, \underset{\text{end}}{r_i}] \subset \mathbb{R}$

Output: max # non-overlapping

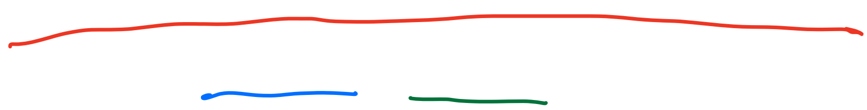
e.g.  $\Rightarrow 2$

Claim: greedy suffices. What rule?

Shortest?



First-come-first-serve?



Fewest overlaps?

Earliest end time? Works!!!

General argument: "greedy stays ahead"

Greedy stays ahead

1) Declare invariant: after k steps of ALG, some invariant holds compared to OPT

2) Prove invariant contradicts

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

```
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
    Return count
```

1) Declare invariant

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

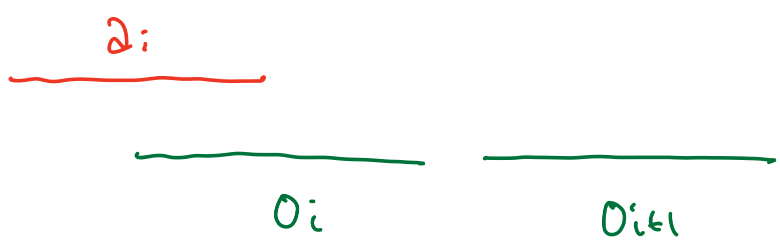
$OPT = \{o_1, o_2, \ldots, o_k, \ldots, o_\ell\}$

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

Proof:

Base case: $a_1 = 1$

Induction: Assume $a_i \leq o_i$

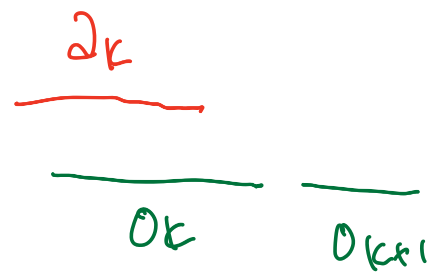


Then $r_{a_i} \leq r_{o_i} < l_{o_{i+1}}$

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

2) Obtain Contradiction

Consider termination.

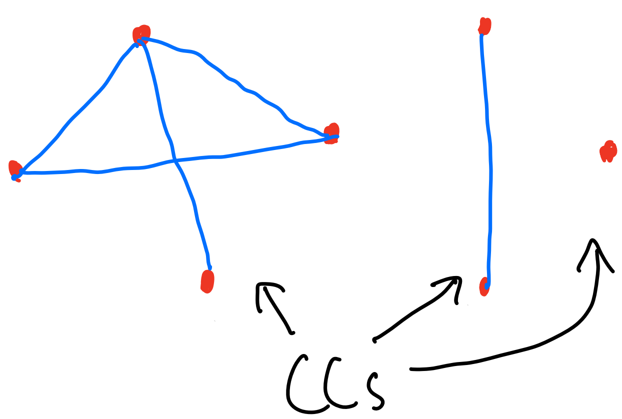


$$\begin{array}{ccccc} \{a_1, & a_2, & \ldots, & a_k\} & \\ \rotatebox{-90}{$\leq$} & \rotatebox{-90}{$\leq$} & & \rotatebox{-90}{$\leq$} & \\ \{O_1, & O_2, & \ldots, & O_k, & O_{k+1}\} \end{array}$$

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

Trees (Part I, Section 4.2)

Undirected graphs only today.



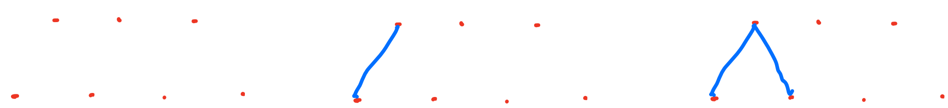
Connected = $\exists$ path

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

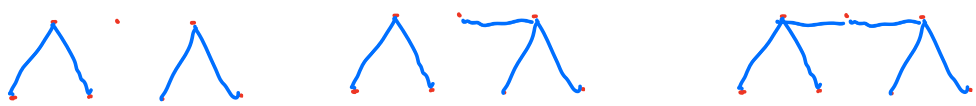
Forest: graph w/ no cycles.

Tree: connected forest.

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

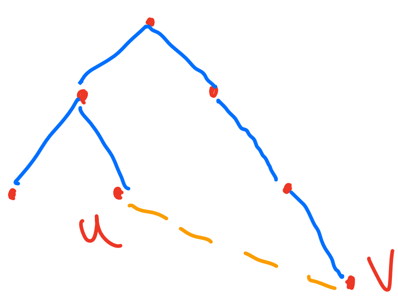


CCs: n $\quad$ $n-1$ $\quad$ $n-2$



3 $\quad$ 2 $\quad$ 1

Key fact 2: off-tree edges make unique cycles



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

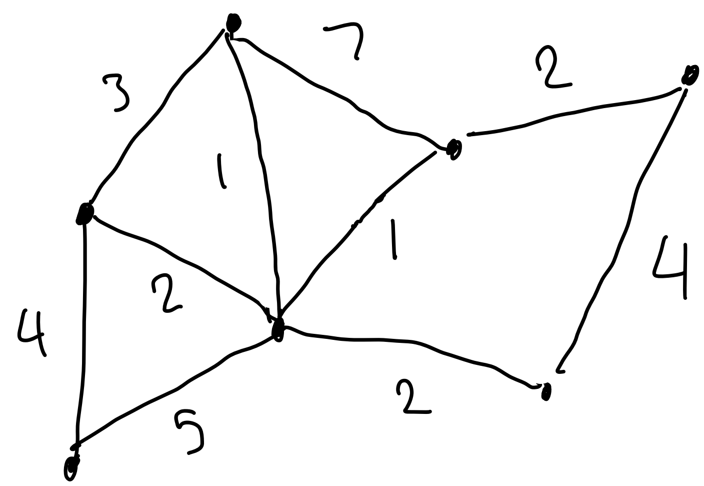
Minimum spanning tree (Part IV, Section 4.1)

Input: (V, E, w) Connected

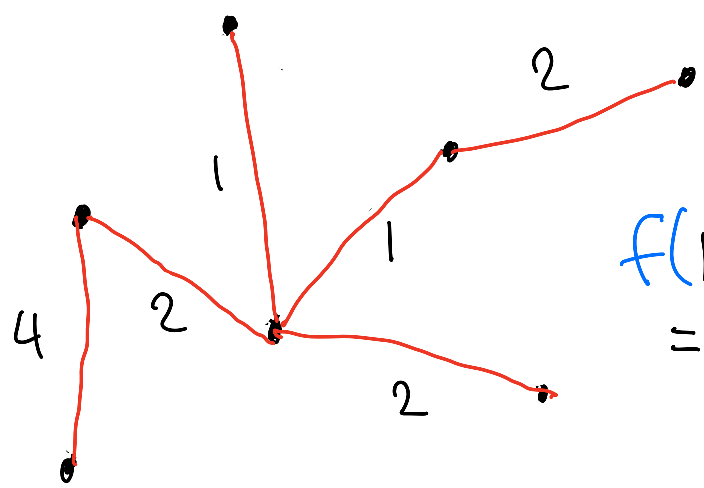
Output: $T \subseteq E$ a tree

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 $e \in E$:

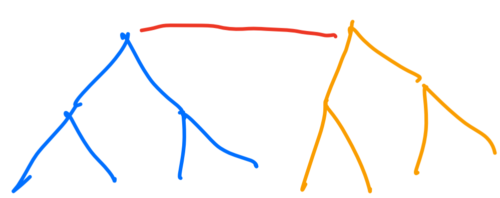
If $T \cup \{e\}$ contains no cycle:

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

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, $|F| < |F'|$

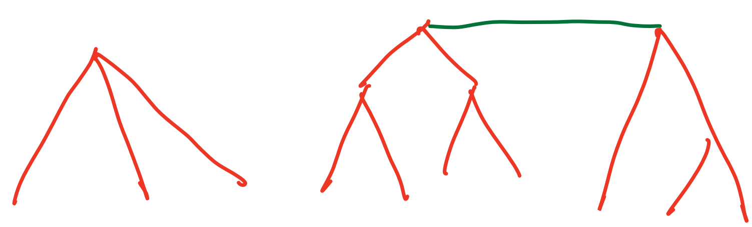
There's some $e \in F'$ s.t. $F \cup \{e\}$ is forest

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 $i \in [n]$:

    $C[i] \leftarrow i$

    $S_i \leftarrow \{i\}$

  $\Big\}\ O(n)$

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

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

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

  $\Big\}\ O(m)$

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

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

Cost of merging:

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

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

- Update $C[w] \leftarrow C[v]$
- Add w to $S_{C[v]}$

$\Big\} \; O(|S_{C[u]}|)$

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