Greedy algorithms (contd.)

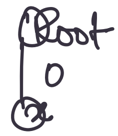
$$\text{Average bit length}(T) = \sum_{x \in S} f_x \cdot \text{depth}_T(x)$$

prefix tree

S = Set of letters / Alphabet.

We want to show that our algo gives an "optimal" prefix tree.

Lemma: Our algorithm gives optimal prefix tree.



Proof: By induction on $|S|$.

Base case: Trivial case.
$|S| = 1$

Ind. hypothesis: $\forall S$, s.t $|S| \le k-1$, algo gives optimal prefix trees.

Ind. step: $|S| = k$.

↳ Algorithm generates a tree T.

Suppose T is not optimal. $\exists Z$ s.t $ABL(Z) < ABL(T)$.

→ Picks two least freq. letters y and z and replaced them w/ a letter w s.t $f_w = f_y + f_z$.

$S \longrightarrow S'$ s.t $|S'| = k-1$.

$T \longleftarrow T'$

$$ABL(T) = \sum_{x \in S} f_x \cdot \text{depth}_T(x)$$

$$= \sum_{x \in S \setminus \{y,z\}} f_x \cdot \text{depth}_T(x)$$

$$+ \; f_y \cdot \text{depth}_T(y) + f_z \cdot \text{depth}_T(z)$$

$$= \sum_{x \in S \setminus \{y,z\}} f_x \cdot depth_T(x) + depth_T(y) \cdot (f_y + f_z).$$

Note that $depth_T(x) = depth_{T'}(x) \quad \forall\, x \in S \setminus \{y,z\}$.

$$depth_T(z) = depth_T(y) = depth_{T'}(w) + 1.$$

$$f_w = f_y + f_z$$

$$= \left(\sum_{x \in S' \setminus \{w\}} f_x \cdot depth_{T'}(x) \right) + f_w \cdot \left(depth_{T'}(w) + 1 \right)$$

$$= \left(\sum_{x \in S'} f_x \cdot depth_{T'}(x) \right) + f_w = ABL(T') + f_w.$$

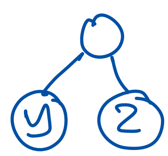
$$ABL(T) = ABL(T') + f_w.$$

Qn: Are y and z siblings in Z ?

↳ Suppose not. Since Z is an optimal tree, y and z occur at same depth.

↳ We can make y and z siblings without changing ABL.

↳ We can assume WLOG that y and z are siblings in Z.

From Z, obtain Z' s.t  is replaced by w

$$\Rightarrow \quad ABL(Z) = ABL(Z') + f_w.$$

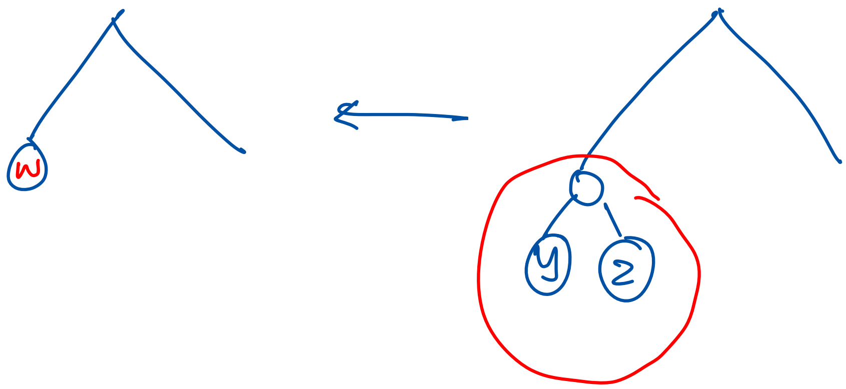
$$f_w + ABL(Z') = ABL(Z) < ABL(T) = ABL(T') + f_w$$

$\Rightarrow \quad ABL(Z') < ABL(T')$.

But T' was optimal (given by the induction hypothesis).

This cannot happen. That is, $ABL(Z)$ can't be less than $ABL(T)$.

$\Rightarrow$ T is optimal



Running time:

$k-1$ iterations

↳ $O(1)$ ExtractMin

$\Big\} \le O(k \log k)$

with.

ExtractMin $\le O(\log k)$

$O(k)$

Interval scheduling.

Premise: Processor / Resource and a set of requests/jobs.

We are given a list of intervals.

Req := $\{1, 2, \ldots, n\}$

We say a subset of req are "compatible" if no two requests have their intervals overlapping.

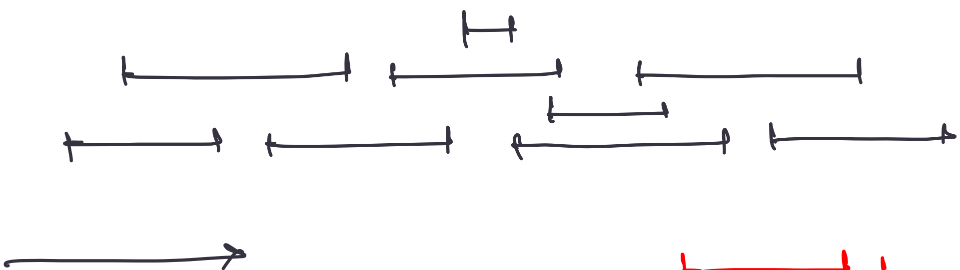
$s(i)$ $f(i)$

time interval.

<u>Goal</u>: Find a largest set of compatible intervals in the set of reqs given.

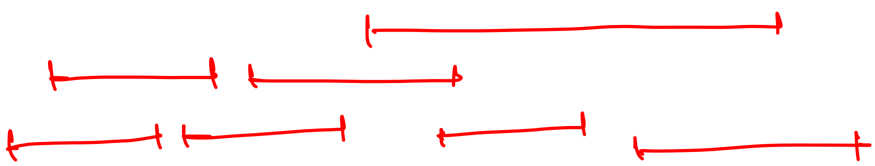
$$R = \{I_1, \dots, I_n\} \; ; \; A \subseteq R$$
$$= \{I_{a_1}, \dots, I_{a_k}\} \text{ s.t } I_{a_i} \cap I_{a_j} = \phi$$
$$\forall\, i \neq j \in \{1, \dots, k\}.$$

<u>Qu</u>: What is the maximum value of k?

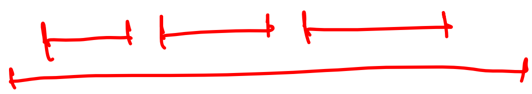


time.

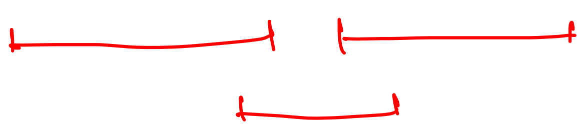
max $k = 5$.



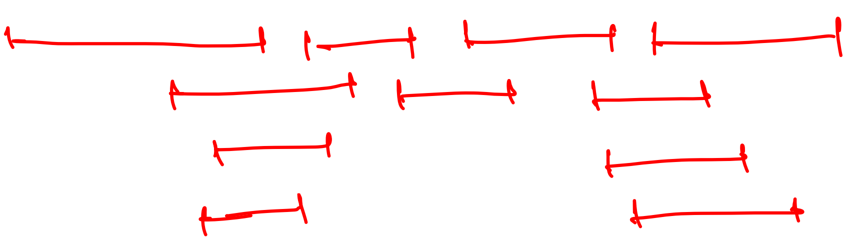
1. Finish times - Early are preferred.
2. Pick the next closest disjoint interval. X
3. Pick the one w/ fewer incompatibilities X
4. Early start time X
5. Shortest time intervals. X



4



5



3

Strategy: Pick req w/ early finish times.

Algo:
Input: R, a set of requests

$A \leftarrow \phi$.

While R is not empty:

Choose a req i w/ lowest finish time.

$A \leftarrow A \cup \{i\}$.

Remove all reqs incompatible w/ i in R, along w/ i.

Return A.

Correctness: A is optimal.

{ maximize the no. of jobs/reqs that are compatible w/ each other.

Say there exists a subset $O \subseteq R$ s.t O is optimal.

$s(u) - f(u)$

$O = \{J_1, \dots, J_m\}$ $\quad$ $A = \{I_1, \dots, I_k\}$.

in sorted order of their times.

If O is optimal, $m \geq k$.

If A is not optimal, $m > k$.

We want to argue that m cannot be strictly larger than k if A was built using our algorithm.

Obs: $f(I_1) \leq f(J_1)$.

Lemma: $\forall\, r \leq k$, $f(I_r) \leq f(J_r)$.

$O = \{J_1, \dots, J_k, \dots J_m\}$

$A = \{I_1, \dots, I_k\}$

$\Rightarrow J_{k+1}, \dots J_m$ are compatible w/ A.

Proof by induction: On $r \in [1, \dots, k]$

Base case: $r = 1$.

I. step: We can assume that Ind. hyp. holds for all $r' \leq r-1$.

$$f(I_{r-1}) \leq f(J_{r-1}). \qquad s(J_r) \geq f(J_{r-1}) \geq f(I_{r-1}).$$

Pick the job w/ least finish time.

I_r $\qquad\qquad\qquad$ J_r

Suppose $f(J_r) < f(I_r)$.

Then exchange J_r and I_r as algorithm would have actually picked J_r.

X.

$\Rightarrow \quad f(I_r) \leq f(J_r)$