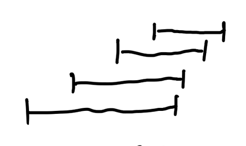
GREEDY ALGORITHMS

TYPICAL GREEDY ALGORITHMS PROOFS HYPOTHESIZE THE EXISTENCE OF AN OPTIMAL SOLUTION O.

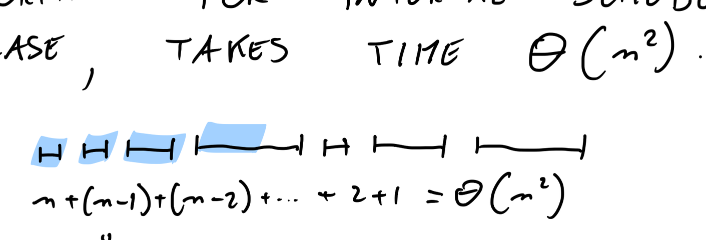
AS THE GREEDY ALGO PROGRESSES, ONE SHOWS THAT ITS SOLUTION (I) MATCHES THE OPTIMAL SOLUTION O, OR (II) MATCHES ANOTHER OPTIMAL SOLUTION (WHICH IS USUALLY OBTAINED BY MODIFYING O).

IN THE END, THIS PROVES THAT GREEDY RETURNS AN OPTIMAL SOLUTION.

THE GREEDY ALGORITHM FOR INTERVAL SCHEDULING, IN THE WORST CASE, TAKES TIME $\Theta(n^2)$.



$O(n)$



$$n+(n-1)+(n-2)+\dots+2+1=\Theta(n^2)$$

$$\frac{n+1}{2}n=\frac{(n+1)n}{2}=\Theta(n^2)$$

FAST ALG (I):

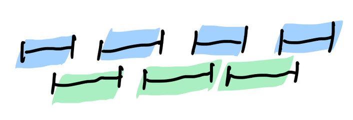
- SORT THE INTERVALS OF I INCREASINGLY BY FINISHING TIME $O(n\log n)$
- LET $I=\{I_1, I_2, \dots, I_n\}$ WITH $f(I_1)\le f(I_2)\le\dots\le f(I_n)$
- SET $T\leftarrow-\infty$, SET $S\leftarrow\emptyset$ $O(1)$
- FOR $i=1,\dots,n$ // n ITERATIONS
  - IF $s(I_i)\ge T$:
    $S=S\cup\{I_i\}$
    $T=f(I_i)$ $\Big\}$ $O(1)$
- RETURN S.

$$O(n\log n)+O(1)+n\cdot O(1)=O(n\log n)$$

EX: PROVE THAT FAST ALG RETURNS AN OPTIMAL SOLUTION.

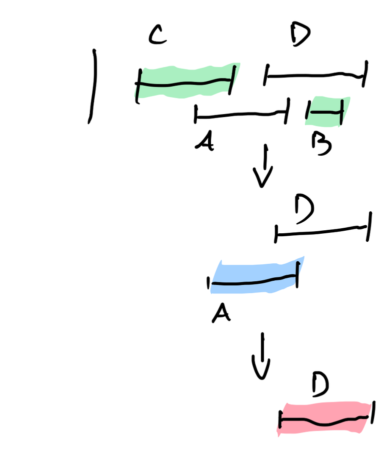
INTERVAL PARTITIONING

WE ARE GIVEN A SET OF INTERVALS I.
WE AIM TO SCHEDULE EACH INTERVAL ON THE MINIMUM POSSIBLE NUMBER OF RESOURCES.



TO SCHEDULE EACH INTERVAL I NEED 2 RESOURCES.

HERE, ONE RESOURCE IS SUFFICIENT



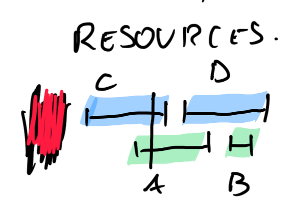
SELECT$_{M^*}$

M^*="EARLIEST TO FINISH"

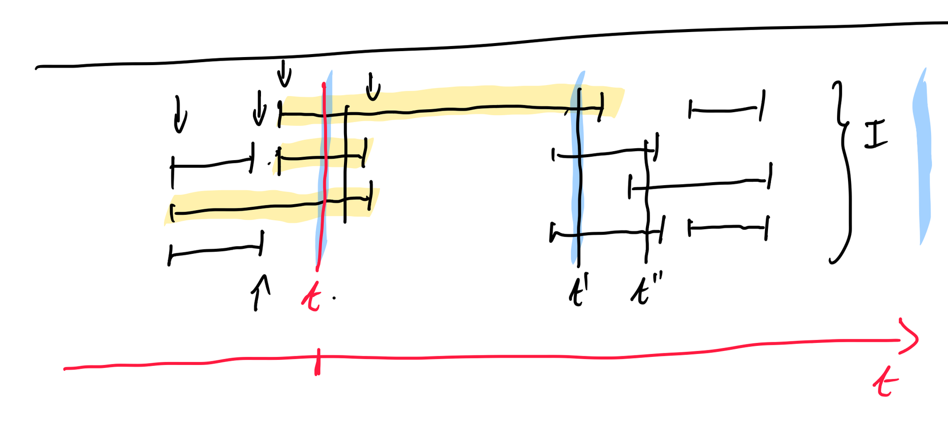
$\{C, B\}$

$\{A\}$ $\rightarrow$ SELECT$_{M^*}$ APPLIED GREEDILY USES 3 RESOURCES.

$\{D\}$

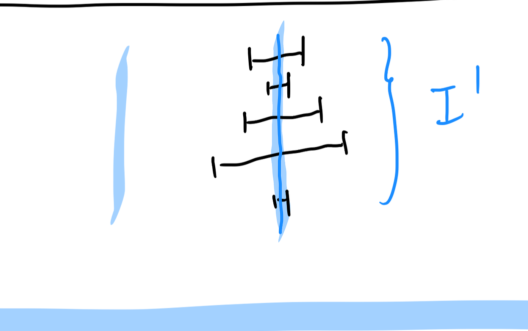


BUT 2 RESOURCES ARE OPTIMAL!



DEPTH $(I)=3$

DEF: DEPTH(I) IS THE MINIMUM INTEGER d S.T. $\forall t\in\mathbb{R}$
$$|\{I_i \mid I_i\in I \wedge t\in I_i\}|\le d.$$

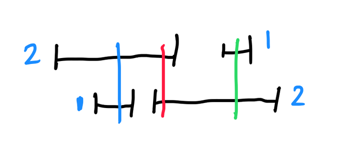


DEPTH $(I')=5$

DEF: LET OPT(I) BE THE MINIMUM NUMBER OF RESOURCES TO SCHEDULE EACH INTERVAL IN I.

L1: OPT$(I)\ge$ DEPTH(I).

P: THERE MUST EXIST A TIME t WHEN EXACTLY DEPTH(I) INTERVALS ARE RUNNING AT THE SAME TIME. AT TIME t, WE THEN NEED DEPTH(I) RESOURCES TO SCHEDULE ALL THE INTERVALS: OPT$(I)\ge$ DEPTH(I) $\square$



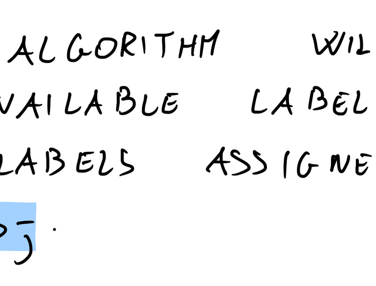
ALG(I):

- LET d BE d=DEPTH(I)
- SORT THE INTERVALS BY THEIR STARTING TIME, INCREASINGLY
- LET $I=\{(s_1,f_1),(s_2,f_2),\dots,(s_n,f_n)\}$ WITH $s_1\le s_2\le s_3\le\dots\le s_n$.
- FOR $j=1\dots n$
  - $L\leftarrow\{1,2,\dots,d\}$
  - FOR $i=1\dots j-1$
    IF (s_i,f_i) IS INCOMPATIBLE WITH (s_j,f_j):
    $L\leftarrow L-\{\ell(i)\}$ // $\ell(i)$
  - IF $|L|\ge1$:
    LET $e\in L$
    SET $\ell(j)=e$
  - ELSE:
    FAIL
- RETURN THE LABELLING $\ell(1),\ell(2),\dots,\ell(n)$.

L2: THE ALGORITHM NEVER FAILS.

P: CONSIDER A GENERIC ITERATION j OF THE LOOP.
LET S_j BE THE SET OF INTERVALS THAT (I) THE ALGORITHM CONSIDERED BEFORE (s_j,f_j) AND THAT (II) END AFTER s_j.

$$S_j=\{(s_i,f_i) \mid i\le j-1 \text{ AND } f_i\ge s_j\}.$$



THE ALGORITHM WILL REMOVE FROM THE SET OF AVAILABLE LABELS FOR (s_j,f_j) ALL AND ONLY THE LABELS ASSIGNED TO THE INTERVALS IN S_j.

IN PARTICULAR, AFTER THE INNER LOOP ENDS,
$|L|\ge d-|S_j|$. (EACH INTERVAL IN S_j REMOVES AT MOST ONE LABEL).

WE WILL PROVE $|L|\ge1$, BY PROVING $d-|S_j|\ge1$. THE LATTER IS EQUIVALENT TO $|S_j|\le d-1$.

CLAIM: $|S_j|\le d-1$

P: EACH INTERVAL IN S_j PASSES THROUGH s_j, AND COMES BEFORE (s_j,f_j) IN THE ORDERING. THEN, $(s_j,f_j)\notin S_j$.

SUPPOSE, BY CONTRADICTION, THAT $|S_j|\ge d$.

THEN, THE SET $S_j\cup\{(s_j,f_j)\}$ HAS A CARDINALITY OF AT LEAST $d+1$.

NOW, EACH INTERVAL IN $S_j\cup\{(s_j,f_j)\}$ PASSES THROUGH TIME s_j.

BUT, THEN DEPTH$(I)\ge|S_j\cup\{(s_j,f_j)\}|\ge d+1$. CONTRADICTION. $\square$

THUS, THE ALGORITHM NEVER FAILS. $\square$

L3: THE ALGORITHMS RETURNS A VALID LABELLING.

P: EXERCISE!

T: THE ALGORITHM RETURNS AN OPTIMAL SOLUTION TO INTERVAL PARTITIONING.

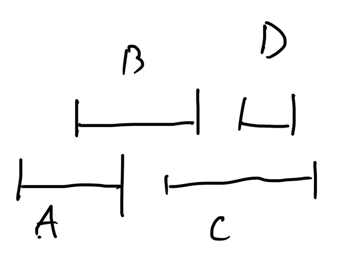
P: APPLY L1, L2, L3. $\square$

EX 2: PROVE THAT, IF INTERVALS ARE SORTED INCREASINGLY BY FINISHING TIME, TODAY'S ALGORITHM FAILS TO FIND AN OPTIMAL SOLUTION IN GENERAL.

E T FINISH

2 1

1 3



E T START

2 2

1 1