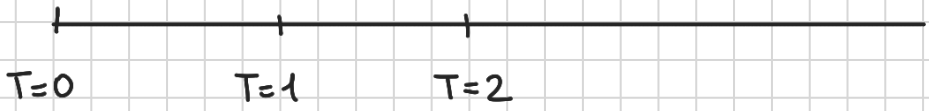
Algorithms

DATE: 10/03/2022

GREEDY ALGORITHMS

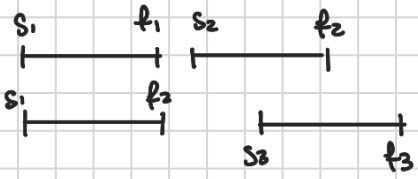
INTERVAL SCHEDULING PROBLEM

1 RESOURCE (CPU, CLASSROOM, CAR)



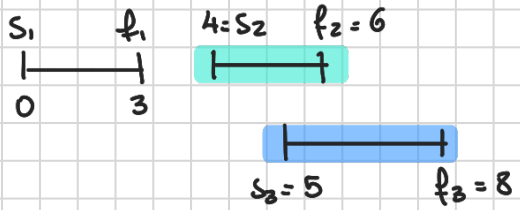
Each job i has a starting time s_i, and an ending time f_i

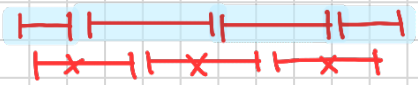
$$I = \{(s_1, f_1), (s_2, f_2), \ldots, (s_n, f_n)\}$$



s = 2

s = 2

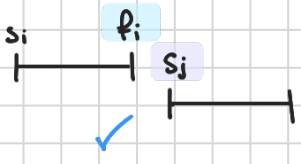


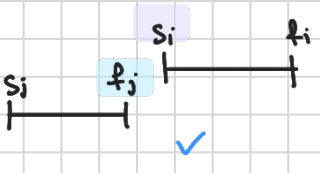


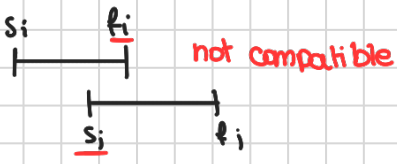
if you choose one, you have to cancel out another

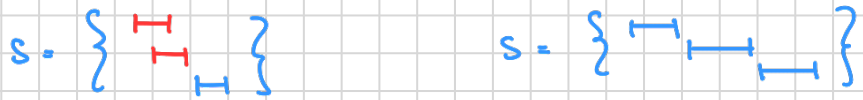
Q: How to find a longest set of "compatible" jobs to be scheduled?

DEF: Let us a consider a set of jobs S. Now, $S \subseteq I$ and it's compatible if $\forall (s_i, f_i), (s_j, f_j) \in S$ and $(s_i, f_i) \neq (s_j, f_j)$, it holds that $\min(f_i, f_j) < \max(s_i, s_j)$









S is incompatible

S is COMPATIBLE

PRECISE PROBLEM DEFINITION

INPUT, INSTANCES

P: Given an input I (a set of slots I), what is a largest subset $S \subseteq I$ of pairwise compatible jobs?

INPUT, INSTANCES

SELECT$_M$ (I):

$S = [\,]$

WHILE $|I| \geq 1$:

PICK $(s_j, f_j) \in I$ according to rule M

$N = \{(s_i, f_i) \mid (s_i, f_i) \in I$ AND S.T. (s_i, f_i) and (s_j, f_j) ARE INCOMPATIBLE$\}$

IT CONTAINS ALL THE REMAINING INTERVALS INCOMPATIBLE WITH (s_j, f_j)

$I = N$ // I and N are sets

S.APPEND$((s_j, f_j))$ // appending compatible intervals

RETURN S

L: $\forall$ rule M, SELECT$_M$ returns a set of compatible jobs/intervals, hence, it returns a FEASIBLE (ammissibile) solution

P: Let (s_j^*, f_j^*) be the jth interval selected by the algorithm. (That is, the interval time the algorithm selects in the jth iteration of its loop).

Likewise, (Allo stesso modo)

- Let $I_0 \triangleq I$ be the set of input intervals, ($\triangleq$ another way to say "uguale =")
- Let $S_0 = [\,]$ be the value of S, before the loop begins;
- let I_j be the value of the variable I at the end of the jth iteration of the loop;
- let S_j be the value of S at the end of jth iteration.

Usiamo l'indice generico j perché vogliamo sapere i valori di I ed S in diversi momenti del ciclo

STRICT SUPERSET

Then, $I_0 \supsetneq I_1, \supsetneq I_2 \supsetneq \dots \supsetneq I_A = \emptyset$, if the loop runs for A iterations.

This is because the set N has to contain at least the interval that the rule M has just selected. In particular, then, that interval will be removed from I.

$A \supsetneq B \iff A \supset B$
$\Updownarrow$
$(B \subseteq A \text{ AND } B \neq A)$

The set I will be decreased of one or more intervals at every iteration.

N will never be empty because there will always be an interval (s_j, f_j) that is incompatible with itself

$S_{j+1} = S_j \cup \{(s^*_{j+1}, f^*_{j+1})\}, \forall j = 0, \dots, t-1$

$\subsetneq$ = STRICT SUBSET

$\emptyset = S_0 \subsetneq S_1 \subsetneq S_2 \subsetneq \dots \subsetneq S_t$

claim ← C: Each interval in I_j is compatible with each interval in S_j, $\forall j \geq 0$

P: IF $j=0$, then $S_j = S_0 = \emptyset$. Thus, $\forall A \in I_j$ "$\forall B \in S_j$" A and B are compatible. This claim is true but "empty". (There is nothing in S_j)

Suppose that the claim holds for j. Let's prove it for $j+1$. That is, we assume that $\forall A \in I_j$ $\forall B \in S_j$ A and B are compatible.
We would then just like to prove that $\forall A \in I_{j+1}$, $\forall B \in S_{j+1}$, A AND B are compatible. Recall that $S_{j+1} = S_j \cup \{(s^*_{j+1}, f^*_{j+1})\}$.
Thus, if we prove that (s^*_{j+1}, f^*_{j+1}) is compatible with each $A \in I_{j+1}$, we are done.

Now, $I_{j+1} \subseteq I_j$ and the generic interval of I_j is in I_{j+1} IFF (IF and only if) it is compatible with (s^*_{j+1}, f^*_{j+1}).

Thus, $\forall A \in I_{j+1}$ is compatible with (s^*_{j+1}, f^*_{j+1}), the inductive step is then proved.

We now show that each of $S_0, S_1, S_2, \dots$ is a set of compatible jobs. Since the last of the S_j's is going to be returned, the algorithm returns a set of compatible jobs.

By induction,

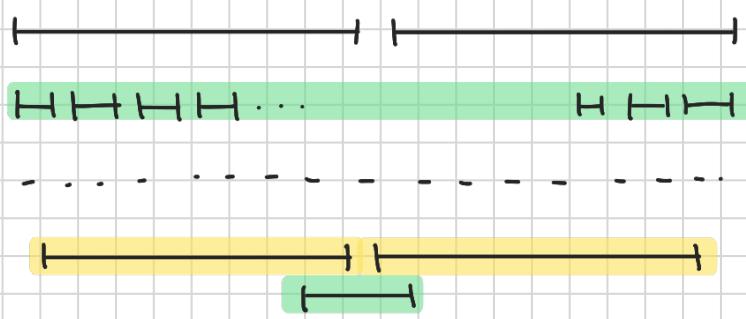
- S_0 is vacuously compatible;
- Assume that the claim holds for S_j. We prove it for $S_{j+1} = S_j \cup \{(s^*_{j+1}, f^*_{j+1})\}$, with $(s^*_{j+1}, f^*_{j+1}) \in I_j$. By the previous claim, each interval in S_j is compatible with each interval in I_j. Thus, $(s^*_{j+1}, f^*_{j+1}) \in I_j$ is compatible with each $B \in S_j$. Moreover, by induction, any two intervals $A, B, \in S_j$. $A \neq B$ are compatible. Thus, S_{j+1} is compatible.

SELECT$_M$ (I): → INPUT, INSTANCES

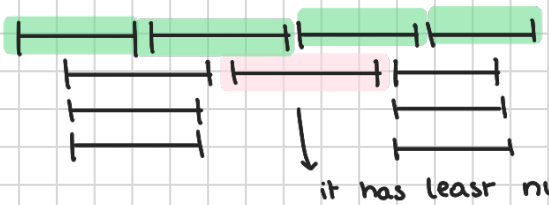
```
S = [ ]
WHILE |I| ≥ 1 :
    PICK (s_j, f_j) ∈ I according to rule M
    N = {(s_i, f_i) | (s_i, f_i) ∈ I AND S.T. (s_i, f_i)
        and (s_j, f_j) ARE INCOMPATIBLE }
    I -= N        // I and N are sets
    S.APPEND((s_j, f_j))
RETURN S
```

HOW TO CHOOSE A RULE M?

- M = "PICK ONE OF THE SHORTEST INTERVALS" ✗

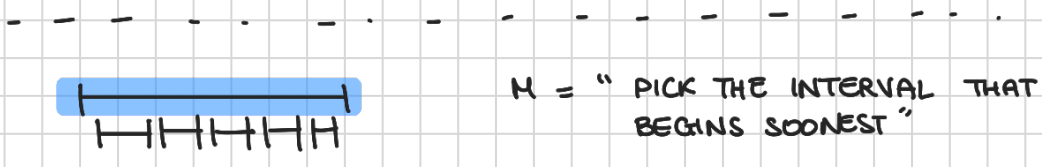


- M = "PICK ONE OF THE INTERVALS HAVING THE SMALLEST NUMBER OF INCOMPATIBILITIES" ✗



→ THIS WOULD BE THE OPTIMAL SOLUTION BUT THE RULE CHOSEN WON'T PICKS THIS

it has leasr number of incompatibilities



M = "PICK THE INTERVAL THAT BEGINS SOONEST"

∎

- M^* = "PICK AN INTERVAL THAT ENDS SOONEST" ✓

L: SELECT_{M^*} returns an optimal solution.

P: Let $O \subseteq I$ be an optimal solution (O is compatible, $|O|$ is as large as possible).

CARDINALITY NOT LOL

W.L.O.G. (WITHOUT LOSS OF GENERALITY), $O = \{J_1, \dots, J_n\}$. Let us denote the ending time of interval A with $f(A)$, and its starting time with $S(A)$.

REPHRASE:

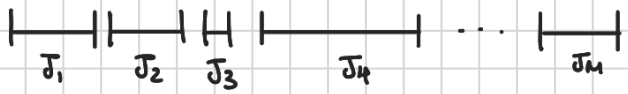
AGAIN, W.L.O.G, we can assume that $f(J_i) < s(J_{i+1})$ $\forall i = 1..m-1$

SUPPOSE we sort the J_i's by their ending time.

$f(J_1) < f(J_2) < f(J_3) < \dots < f(J_n)$

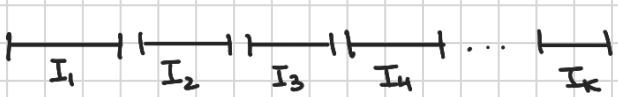
Then it must be that $s(J_i) < f(J_1)$. Moreover, $f(J_i) < s(J_2)$ - if this does not hold J_1 and J_2 are incomp., and thus O is not a solution. Thus, by induction,

$$s(J_1) < f(J_1) < s(J_2) < f(J_2) < s(J_3) < f(J_3) < \dots < s(J_m) < f(J_m).$$



Let S be the solution by SELECT_M. By the previous lemma, S is compatible. Then, W.L.O.G., $S = \{I_1, I_2, \dots, I_k\}$ such that

$$s(I_1) < f(I_1) < s(I_2) < f(I_2) < \dots < s(I_k) < f(I_k)$$

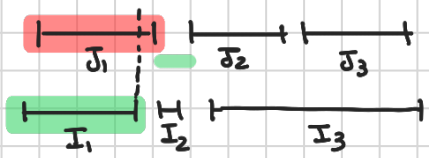


We would like to prove that $k \geq m$. (our algorithm is no worse than the optimum), $|S| \geq |O|$.

CLAIM

C: $f(I_1) \leq f(J_1)$

P: By our rule M^*, I_1 is going to be one of the intervals that end soonest. Thus, $f(I_1) \leq f(J_1)$.



Thus, if $O = \{J_1, J_2, \dots, J_m\}$ was an optimal solution, then $O_1 = \{I_1, J_2, J_3, \dots, J_m$ is also an opt. sol.

Recall that $S = \{I_1, I_2, \dots, I_k\}$

The solution O_1 is optimal and shares its first interval with our greedy solution.

C: For each $1 \le i \le k-1$, $\exists$ Optimal solution O_{i+1} S.T. $O_{i+1} = \{I_1, I_2, \dots, I_{i+1}, J_{i+2}, J_{i+3}, \dots, J_m\}$ and $f(I_{i+1}) \le f(J_{i+1})$

P: We already proved the claim for $i=0$.
Let us assume to have proved the claim for i. We prove it for $i+1$

Thus, $f(I_{i+1}) \le f(J_{i+1})$. Our algorithm picks the next interval among those that are compatible with $\{I_1, \dots I_{i+1}\}$ and that ends soonest. - Let us call this interval I_{i+2} whichever Interval J_{i+2} is, it must be compatible with $I_1, I_2, \dots, I_{i+1}$ then $f(I_{i+2}) \le f(J_{i+2})$.

Thus defining $O_{i+2} = \{I, I_2, \dots, I_{i+1}, I_{i+2}, J_{i+3}, \dots, J_m\}$ results in an optimal, feasible, solution (I_{i+2} is compatible with each of $I, \dots, I_{i+1}$ and since $f(I_{i+2}) \le f(J_{i+2})$, it is also compatible with $J_{i+3}, J_{i+4}, \dots, J_n$. ■

Thus, we know that the solution $O_K = \{I_1, I_2, \dots, I_k, J_{k+1}, \dots, J_k\}$ is optimal. (DOESN'T EXIST)

Since the greedy algorithms stops exactly when there are no more compatible jobs with the ones it selected, it must be that $J_{k+1}, J_{k+2}, \dots, J_m$ DON'T EXIST

Thus, $O_k = \{I_1, \dots, I_k\}$, so that $m = k$, and the greedy solution is optimal ■

GREEDY ALGOS proofs hypothesize the existence of some optimal solution O
As the greedy algo progresses one shows that the existing hypothetical optimal solution can be turned into an optimal solution that matches all the choices made by the greedy algo so far.

Thus, in the end, greedy can be shown to return an optimal solution.

In other words, you should always be able to prove that the current (partial) solution of the greedy algo can be extended to a full optimal solution.

NOTE: Remember that whenever we want to study an algorithm, there are essentially three things we want to do:

1. We want to prove that whatever it returns is a valid solution
2. We want to prove ideally that the solution is optimal
3. We want to prove that the runtime of the algorithm is small