14/02/2022 : Greedy Algorithms

- DP: Solve optimization problems
  - At each step you have a set of choices.
  - figure out what is the best choice by solving subproblems.

- Greedy algorithms:
  you make a locally optimal choices. in the hope that it leads to a global optimal solution.

Interval Scheduling:

you have a resource — a lecture room, a supercomputer, etc.

People who wants to use the resource. They make a request by saying I want to reserve it from time s to f.

Assumption: The resource can be used by at most one person at a time.

Two requests are compatible if the requested intervals do not overlap.

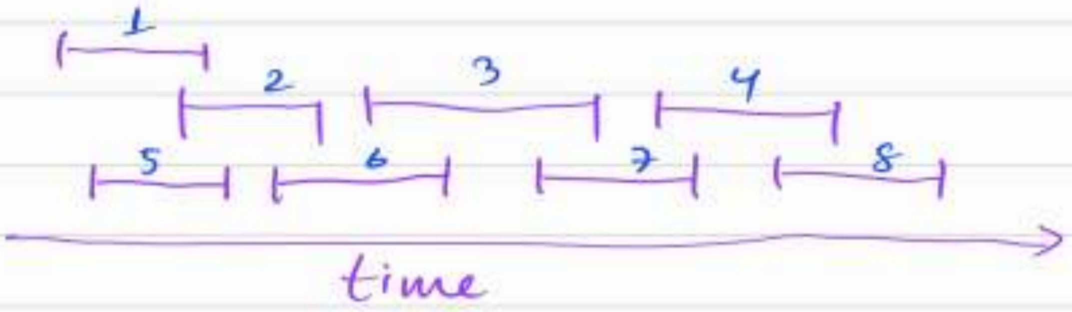
Goal :- is to find a maximum-sized set of compatible requests.

Formally, a set of n requests labeled say $1, \ldots, n$.

Each request i has a start time $s(i)$ and a finish time $f(i)$

Compatible requests: two requests i and j are compatible if $f(i) \le s(j)$ or $f(j) \le s(i)$

Goal: to find a maximum sized set of compatible requests.

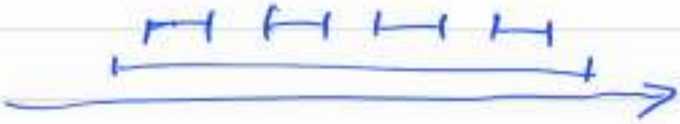


solutions :- $\{2, 3, 4\}$, $\{5, 6, 7, 8\}$

$\{1, 6, 7, 8\}$, $\{1, 3, 4\}$

↑

maximum sized-set.



DP-algo:- either the request 1 is in the optimal set or not.

Case 1:- Suppose 1 is in a optimal set

Before $:= \{ i \mid 2 \le i < n \text{ s.t. } f(i) \le s(1) \}$

After $:= \{ i \mid 2 \le i \le n \text{ s.t. } s(i) \ge f(1) \}$

Optimal Solution $:= \{1\} \cup$ Optimal-before $\cup$ optimal-after.

Case 2: 1 is not in any optimal solution

recursively solve it on $\{2, \dots, n\}$

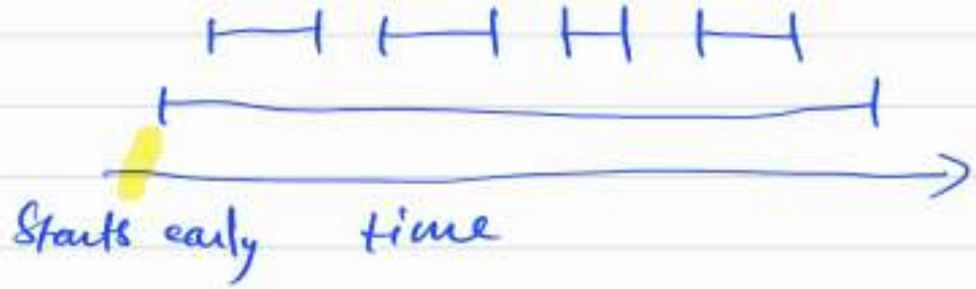
runtime:- $O(n^3)$.

Greedy algorithm:

- Choice to pick the first request i_1
- remove all request that are not compatible with i_1 from the set of all request.

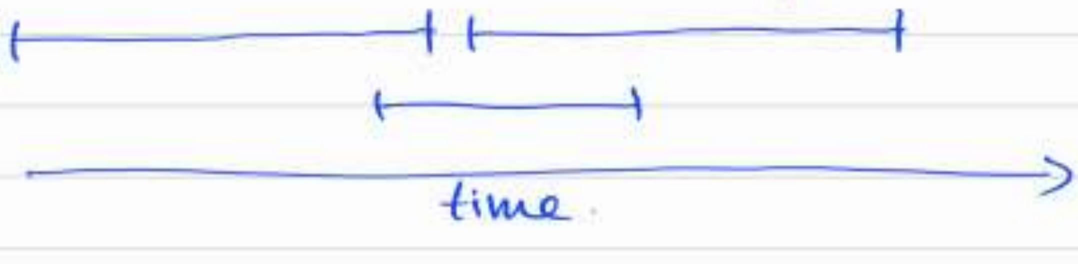
Challenge:- how to choose the first request i_1?

Choice 1: pick a request with earliest start time.



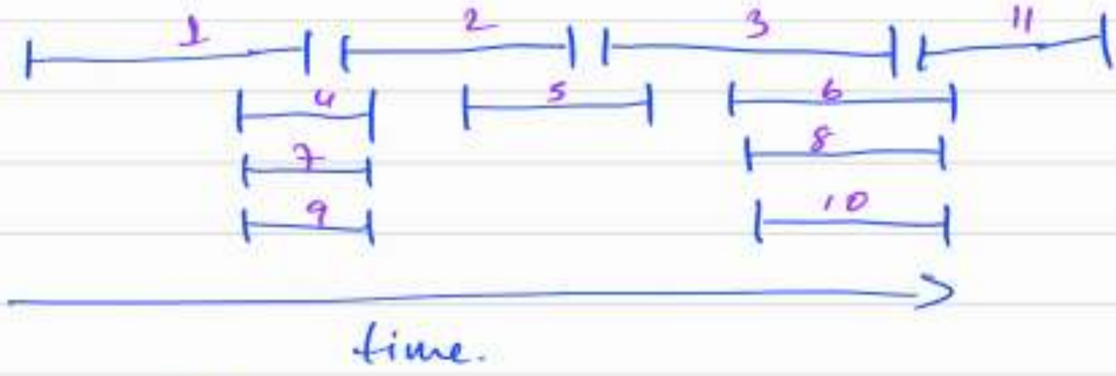
This doesn't work.

Choice 2:- pick a request that $f(i) - s(i)$ as small as possible.



Choice 3:- pick a request that has minimal no. of conflicts.

for every request count the no. of request that overlaps with it.
and then pick the one with smallest count.



$$\left.\begin{array}{l} 1 \to 3 \\ 2 \to 4 \\ 3 \to 4 \end{array}\right\} \begin{array}{l} 4, 7, 9, 6, 8, 10 \to 3 \\ 5 \to 2 \end{array}$$

If you choose 5 then maximal no is 3.
but the optimal is $4 = \{1, 2, 3, 11\}$

Choice 4: pick a request with earliest finish time.

" more free time to serve other requests"

define $R :=$ set of all requests that are neither accepted nor rejected
$A :=$ Set of all accepted requests.

Initialise R to be the set of all request and A to be the empty set.

While R is not empty.
- pick a request with earliest finish time
- add this request to A
- remove incompatible requests from R

End while.

Output A.

<u>Observation</u>: A is a set of compatible request.

Analysis of the Algorithm:

<u>First way</u>:- "Greedy stays ahead"

sorting your requests based on finish time

$$f(1) < f(2) < \cdots < f(n)$$

the Algorithm adds request 1 to A.

Our algorithm finishes the first request before the output from any other algorithm.

$$A = \{i_1, \ldots, i_k\}$$

$$opt = \{j_1, \ldots, j_m\}$$

<u>Goal</u>: to prove that $k = m$.

<u>Claim</u>:- for any $l \leq k$, $f(i_l) \leq f(j_l)$

<u>Proof</u>:- Based on induction on l.

<u>Base case $l = 1$</u>: $f(i_1) \leq f(j_1)$

by the choice.

Induction Step: arbitrary l.

$$f(i_1) \leq f(j_1),\ f(i_2) \leq f(j_2),\ \ldots,\ f(i_{l-1}) \leq f(j_{l-1}).$$

Can this happen?

Consider the l-th time

No. because, $f(i_{l-1}) \leq f(j_{l-1}) \leq s(j_l)$

Theorem:- A is a maximum sized-set.

Proof:- Suppose not: This implies there exist an optimal set $\{j_1, \ldots, j_m\}$ s.t. $m > K$.

Applying previous claim with $l = k$.

$$f(i_K) \leq f(j_K) \leq s(j_{K+1})$$

The above inequality implies R is not empty which is a contradiction

Runtime & Implementation:- Sort according to finish time and then scan according to start time. $O(n \log n)$.

$f(1) \le f(2) \le \cdots f(n)$

$A = A \cup \{i\}$, $s(1) \le s(2) \le \cdots \le s(n)$

Second way "Exchange argument".

- Greedy output
- Optimal solution

→ massage your optimal solution into the greedy output without losing on optimality.

Lemma:- $\exists$ an optimal solution that contains the request with earliest finish time.

Proof:- Pick an optimal solution B that doesn't contain the request with earliest finish time.

$$B = \{ j_1, \ldots, j_m \}$$

let i_1 be the request with earliest finish time.

$$f(i_1) \le f(j_1)$$

$B \setminus \{j_1\} \cup \{i_1\}$. This has the same size.