

Greedy Algorithms.

Interval Scheduling:

Input: n intervals (s_i, f_i)

Story: only 1 classroom.

Schedule as many classes as possible w/o conflicts.

If (s_i, f_i) and (s_j, f_j) do not overlap,
there is no conflict between them.

Output: the maximum number of intervals that we can schedule without conflict.

Example 1: $(1, 8) \quad (2, 3) \quad (4, 5) \quad (6, 7)$.



Example 2: $(1, 5) \quad (6, 10), \quad (4, 7)$.



Example 3.



Greedy: decide on a simple rule that selects the first interval and include it in the final solution.



Remove all intervals that conflict with the chosen interval.

① Choose an interval that starts the earliest.
This is not optimal (see Example 1).

② Choose a shortest interval.

This is not optimal (see Example 2).

③ Choose an interval with the fewest number of conflicts.
This is not optimal (see Example 3)

④ Choose an interval that finishes the earliest.
(break ties arbitrarily).

Correctness:

- We output a compatible set of intervals. ✓
- We output an optimal solution.

Idea: compare the algorithm's solution with some optimal solution.

Proof: Fix an input.

Let $|A|=k$ be the set of interval chosen by GREEDY.

Let $|O|=m>k$ be an optimal solution.

$A = (i_1, i_2, \dots, i_k)$ → Ordered from left to right.

$O = (j_1, j_2, \dots, j_m)$.

Claim: $\forall 1 \leq r \leq k. f(i_r) \leq f(j_r)$

Base case: $f(i_1) \leq f(j_1)$

Inductive step: Suppose $f(i_{r-1}) \leq f(j_{r-1})$.

Then $f(i_r) \leq f(j_r)$.

$A: \text{---} \dots \text{---} i_{r-1} \text{---} i_r \text{---}$

$O: \text{---} \dots \text{---} j_{r-1} \text{---} j_r \text{---}$

This impossible because
GREEDY can pick the
interval j_r .

Consequently, $f(i_k) \leq f(j_k)$. \Rightarrow Contradict that $|A|=k$.
Because GREEDY can
pick j_{k+1} .

Implementation and Runtime: (n intervals).

$\Theta(n \log n)$ Sort all intervals by finishing time (early to late).
(In the sorted list):

$$\text{current-finish} = -\infty, A = \emptyset.$$

For $i = 1$ to n

if $(\text{current-finish} \leq s_i)$

Add i to A :

$\text{current-finish} = f_i$;

End if

End for.

Return A .

Overall runtime
 $= \Theta(n \log n)$.