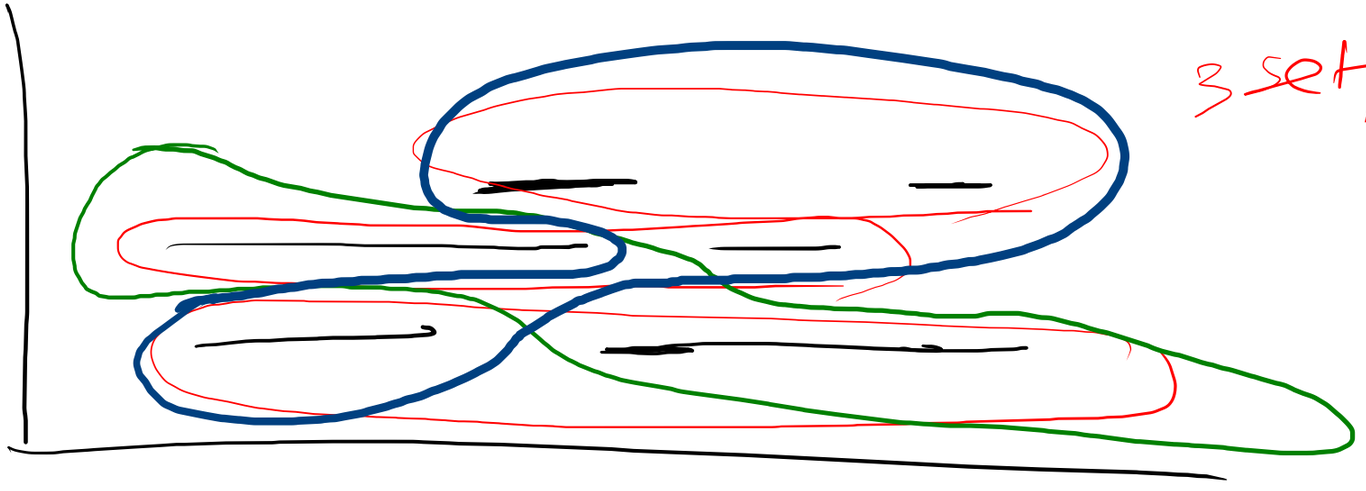
Interval partitioning problem

Input: A set of n jobs $J = \{J_1, J_2, \ldots, J_n\}$ where each job J_i has a start time s_i and a finish time f_i.

Output: A partition of the jobs in J into K sets such that each set of jobs are mutually compatible.

Objective: minimize K



3 sets $K=3$

green, blue } 2 sets $K=2$

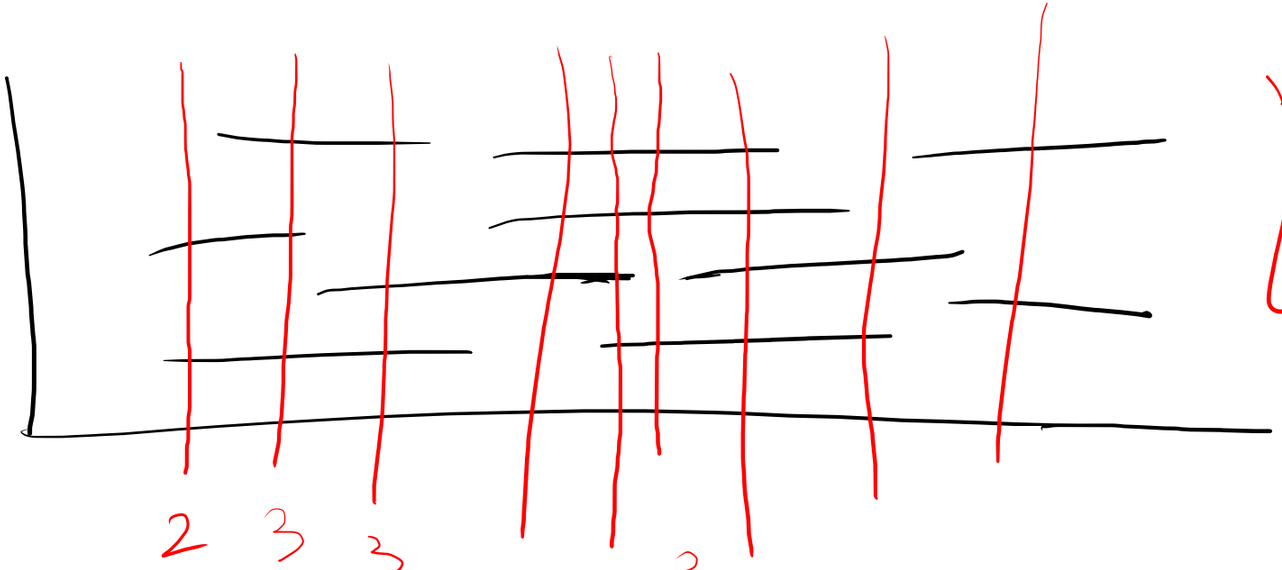
Situations where the problem appears

i) you have a set of fixed jobs and want to schedule them using minimum resources.

ii) There are a set of lectures that are schedule in some classrooms.
one wants to schedule the lectures using minimum classrooms.

Depth of a set of intervals

The maximum number of intervals that contain any given time



depth = 4

observation: Number of sets must be $K \geq$ depth

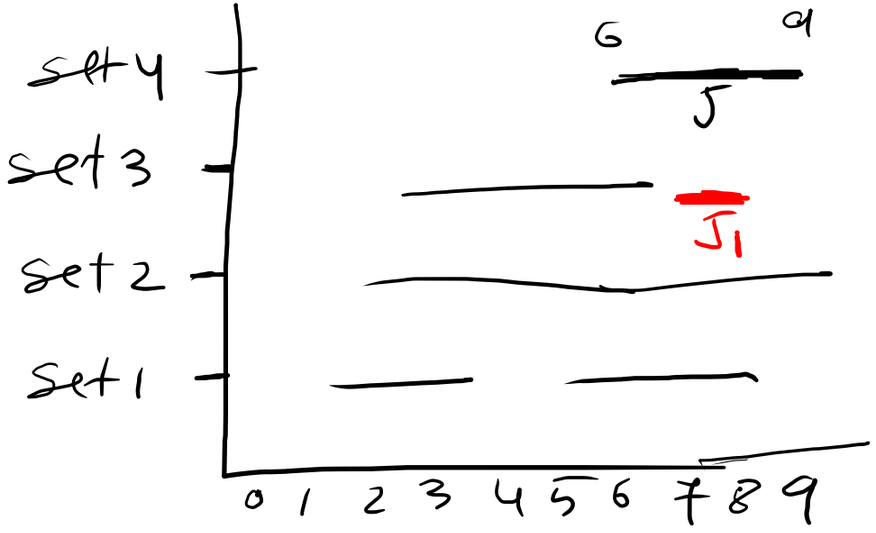
Question: What about K = depth ??

$\Rightarrow$ schedule is optimum.

Target: — Need a solution with depth number of sets.

We design greedy algorithm

- consider the jobs in some order — Which order?
- Assign each job to an available set — Which set?
  - If all the ~~sets~~ are not available then create a new set.



j → 6 — 9

J_1 → 7 - 8

rule 1 : Earliest finish time ✗

rule 2 : shortest interval first ✗

rule 3 : Fewest conflict ✗

rule 4 : Earliest start time. ✓

H.W. counter examples

Earliest start time

Input $(n, (s_1, f_1), (s_2, f_2), \ldots, (s_n, f_n))$

- Sort the jobs in increasing order of their start time. — $O(n \log n)$

$$s_1 \leq s_2 \leq s_3 \leq \cdots \leq s_n$$

- depth 0 — $\theta(1)$
- for $i = 1$ to n — $O(n)$ times
  - if J_i is compatible with some job in any set — $O(n)$
    - assign J_i in such set.

Priority queue $O(\log n)$

  - else
    - create a new set — $\theta(1)$
    - schedule J_i in depth+1 set
    - depth = depth + 1
- return the schedule.

Total time $O(n^2)$

improved $O(n \log n)$

correctness

claim: The algorithm never schedule two incompatible jobs in the same set.

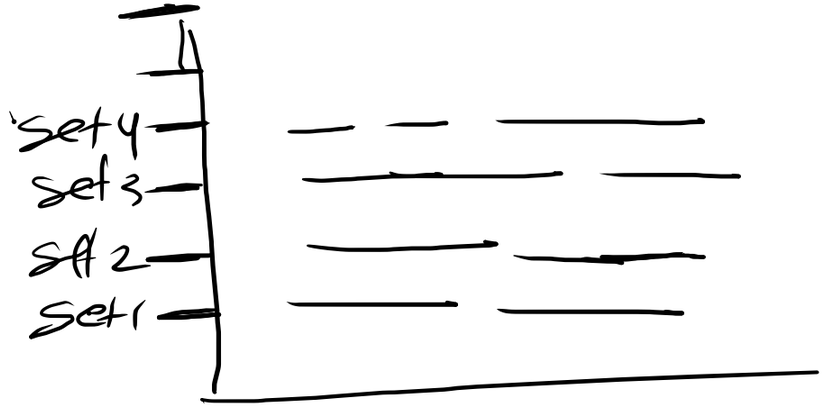
straightforward.

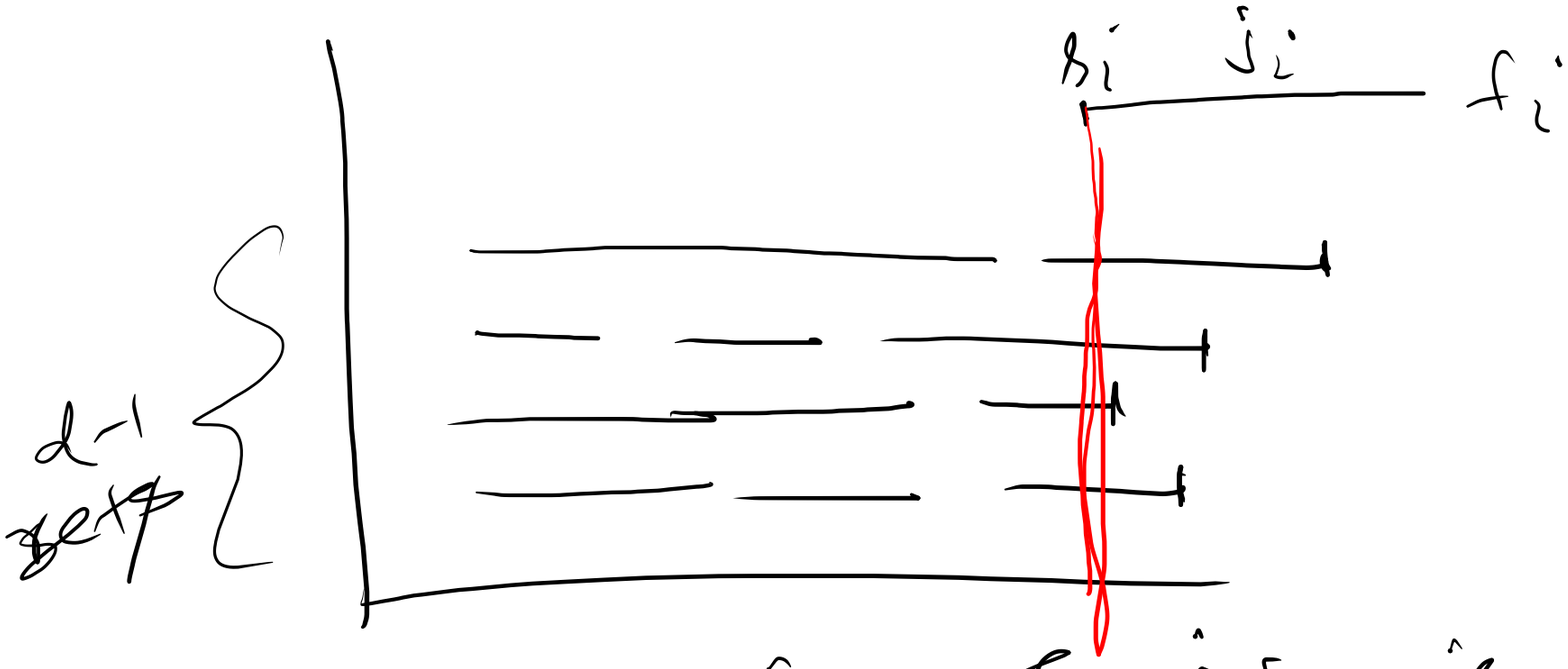
claim: The greedy algorithm is optimal.

Proof: Let d be the no. of sets the algorithm returns.

Question: When the last set ie, the d-th set first used?

The algorithm trying to add J_i^{th} job but it is incompatible with all other $d-1$ sets.





The start time of j_i is in between the start time and finish time of all the last jobs in d-1 sets.

Therefore the algorithm uses d sets only because of ther are d overlapping jobs ie, the depth of the problem is d.