4.1 Interval Scheduling
Interval scheduling.

- Job $j$ starts at $s_j$ and finishes at $f_j$.
- Two jobs compatible if they don't overlap.
- Goal: find maximum subset of mutually compatible jobs.

![Interval Scheduling Diagram](image-url)
Interval Scheduling: Greedy Algorithms

**Greedy template.** Consider jobs in some order. Take each job provided it's compatible with the ones already taken.

- [Earliest start time] Consider jobs in ascending order of start time $s_j$.
- [Earliest finish time] Consider jobs in ascending order of finish time $f_j$.
- [Shortest interval] Consider jobs in ascending order of interval length $f_j - s_j$.
- [Fewest conflicts] For each job, count the number of conflicting jobs $c_j$. Schedule in ascending order of conflicts $c_j$. 
Interval Scheduling: Greedy Algorithms

**Greedy template.** Consider jobs in some order. Take each job provided it's compatible with the ones already taken.

- Breaks earliest start time
- Breaks shortest interval
- Breaks fewest conflicts
**Interval Scheduling: Greedy Algorithm**

**Greedy algorithm.** Consider jobs in increasing order of finish time. Take each job provided it's compatible with the ones already taken.

```
Sort jobs by finish times so that \( f_1 \leq f_2 \leq \ldots \leq f_n \).

// jobs selected
A ← φ
for j = 1 to n {
    if (job j compatible with A)
        A ← A ∪ {j}
}
return A
```

**Implementation.** \( O(n \log n) \).

- Remember job \( j^* \) that was added last to \( A \).
- Job \( j \) is compatible with \( A \) if \( s_j \geq f_j^* \).
Theorem. Greedy algorithm is optimal.

Proof. (by contradiction)

Assume greedy is not optimal, and let's see what happens.

Let $i_1, i_2, \ldots, i_k$ denote set of jobs selected by greedy.

Let $j_1, j_2, \ldots, j_m$ denote set of jobs in the optimal solution with $i_1 = j_1$, $i_2 = j_2$, ..., $i_r = j_r$ for the largest possible value of $r$.

why not replace job $j_{r+1}$ with job $i_{r+1}$?
Theorem. Greedy algorithm is optimal.

Pf. (by contradiction)

- Assume greedy is not optimal, and let's see what happens.
- Let $i_1, i_2, \ldots, i_k$ denote set of jobs selected by greedy.
- Let $j_1, j_2, \ldots, j_m$ denote set of jobs in the optimal solution with $i_1 = j_1, i_2 = j_2, \ldots, i_r = j_r$ for the largest possible value of $r$.

Greedy: $i_1, i_1, i_r, i_{r+1}$

OPT: $j_1, j_2, j_r, i_{r+1}$

Job $i_{r+1}$ finishes before $j_{r+1}$

Solution still feasible and optimal, but contradicts maximality of $r$. 
4.1 Interval Partitioning
Interval Partitioning

Interval partitioning.

- Lecture $j$ starts at $s_j$ and finishes at $f_j$.
- Goal: find minimum number of classrooms to schedule all lectures so that no two occur at the same time in the same room.

Ex: This schedule uses 4 classrooms to schedule 10 lectures.
Interval Partitioning

Interval partitioning.

- Lecture $j$ starts at $s_j$ and finishes at $f_j$.
- Goal: find minimum number of classrooms to schedule all lectures so that no two occur at the same time in the same room.

Ex: This schedule uses only 3.
Interval Partitioning: Lower Bound on Optimal Solution

Def. The **depth** of a set of open intervals is the maximum number that contain any given time.

**Key observation.** Number of classrooms needed $\geq \text{depth}$.

**Ex:** Depth of schedule below = 3 $\Rightarrow$ schedule below is optimal.

a, b, c all contain 9:30

**Q.** Does there always exist a schedule equal to depth of intervals?
Interval Partitioning: Greedy Algorithm

Greedy algorithm. Consider lectures in increasing order of start time: assign lecture to any compatible classroom.

Sort intervals by starting time so that \( s_1 \leq s_2 \leq \ldots \leq s_n \).

d \leftarrow 0 \quad \text{--- number of allocated classrooms}

for \( j = 1 \) to \( n \) {
    if (lecture \( j \) is compatible with some classroom \( k \))
        schedule lecture \( j \) in classroom \( k \)
    else
        allocate a new classroom \( d + 1 \)
        schedule lecture \( j \) in classroom \( d + 1 \)
        \( d \leftarrow d + 1 \)
}

Implementation. \( O(n \log n) \).

- For each classroom \( k \), maintain the finish time of the last job added.
- Keep the classrooms in a priority queue.
Observation. Greedy algorithm never schedules two incompatible lectures in the same classroom.

Theorem. Greedy algorithm is optimal.

Pf.

- Let $d =$ number of classrooms that the greedy algorithm allocates.
- Classroom $d$ is opened because we needed to schedule a job, say $j$, that is incompatible with all $d-1$ other classrooms.
- Since we sorted by start time, all these incompatibilities are caused by lectures that start no later than $s_j$.
- Thus, we have $d$ lectures overlapping at time $s_j + \varepsilon$.
- Key observation $\Rightarrow$ all schedules use $\geq d$ classrooms. $\blacksquare$