9 Scheduling, Interval Partitioning

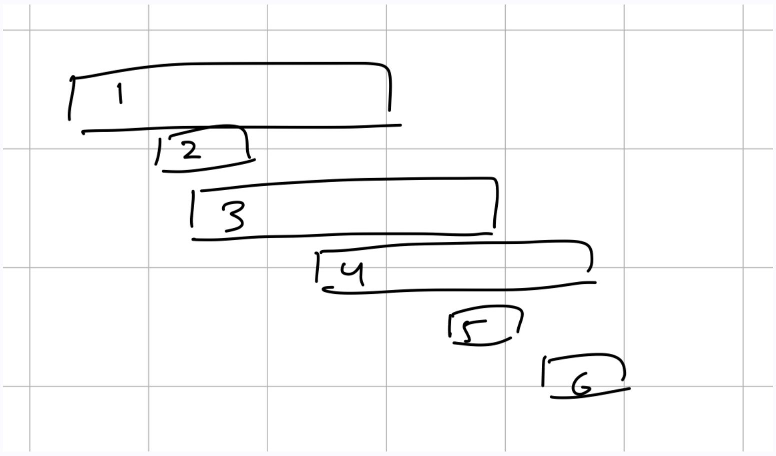
- Scheduling
- Divide and Conquer

9.1 Scheduling

Interval scheduling: Given n requests with starting times s_i and finishing times f_i for $i \in \{1, 2, \cdots, n\}$.
A subset of requests is compatible if no two intervals $[s_i, f_i]$ overlap.

Our goal is to find a largest possible compatible subset (with respect to the number of requests).

Example 9.1



Here, $\{1, 5, 6\}$ and $\{2, 5, 6\}$ are optimal solutions.

How can we be greedy in this problem? We could:

- Choose shortest activity first
- Take earliest finishing activity first (optimal)
- Take earliest activity first

```
GreedyIntervalSchedule(s, f):
    sort tasks by increasing order of finish times
    let A be an empty set
    let f_prev = -infty
    for i=1 to n
        if s_i > f_prev then
            add task i to A
            set f_prev = f_i
        endif
    endfor
```

This algorithm takes time $O(n \log n)$.
But why is the algorithm correct?

Firstly, there are no conflicts since we only schedule a task that starts after the previous task finished.

We will show that the number of tasks is maximal by an exchange argument. (Can also show this algorithm "stays ahead" - in book)

Theorem 9.2
GreedyIntervalSchedule outputs an optimal schedule.

Proof. Let $G = (g_1, \cdots, g_k)$ be the greedy schedule.
Let $B = (b_1, \cdots, b_l)$ be an optimal schedule ($l \geq k$).

Let j be the first index where the schedules differ:
$G = (g_1, \cdots, g_{j-1}, g_j, \cdots, g_k)$
$B = (g_1, \cdots, g_{j-1}, b_j, \cdots, b_l)$

Switch B to $B' = (g_1, \cdots, g_{j-1}, g_j, b_{j+1}, \cdots, b_l)$

By the greedy choice, $f_{g_j} \leq f_{b_j}$, so this schedule still has no conflicts, and it is just as long (still has l intervals).

Repeating this process, we get a schedule $(g_1, \cdots, g_k, b_{k+1}, \cdots, b_l)$.

If $l > k$, then b_{k+1} is an index of an interval that the greedy algorithm could have scheduled. But it didn't, so $l = k$. □

9.2 Interval Partitioning

Suppose we must schedule all intervals (given by starting and finishing times) while minimizing the number of resources used (number of classrooms used).

Definition 9.3
The **depth** of a set of intervals is $\max_t |\{i \in \{i, \cdots, n\} : t \in [s_i, f_i]\}|$ i.e., we wish find the largest number of overlapping subsets at any time t.

Our greedy strategy is that we scan through the intervals in increasing order of start time. Assign each to any available resource from $\{1, \cdots, \text{depth}\}$.

Theorem 9.4
This algorithm assigns a color from $\{1, \cdots, \text{depth}\}$ to every interval and no two overlapping inttervals have the same color.

Proof. When we assign a color, it is different from the colors of all overlapping intervals scheduled so far, so we never assign the same color to overlapping intervals.

Suppose when we are considering interval i, it overlaps t previous intervals.
So there are $t + 1$ overlapping intervals. This means the depth is $\geq t + 1$.
Then $t \leq \text{depth} - 1$, so there is always a color available. □