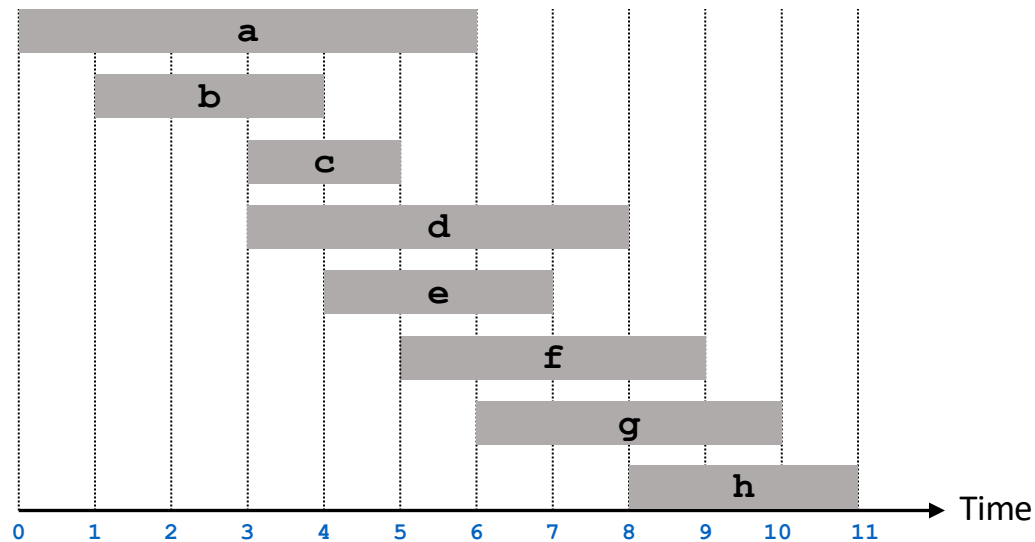


Greedy Algorithms

a. Unweighted Interval Scheduling

Interval Scheduling: Problem Definition

- 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: Greedy Attempts

- **Greedy template:** Consider jobs in some natural order. Take each job provided it's compatible with the ones already taken.
 - [Earliest start time] Consider jobs in ascending order of s_j .
 - [Earliest finish time] Consider jobs in ascending order of f_j .
 - [Shortest interval] Consider jobs in ascending order of $f_j - s_j$.
 - [Fewest conflicts] For each job j , count the number of conflicting jobs c_j . Schedule in ascending order of c_j .

Interval Scheduling: Greedy Attempts



counterexample for earliest start time



counterexample for shortest interval



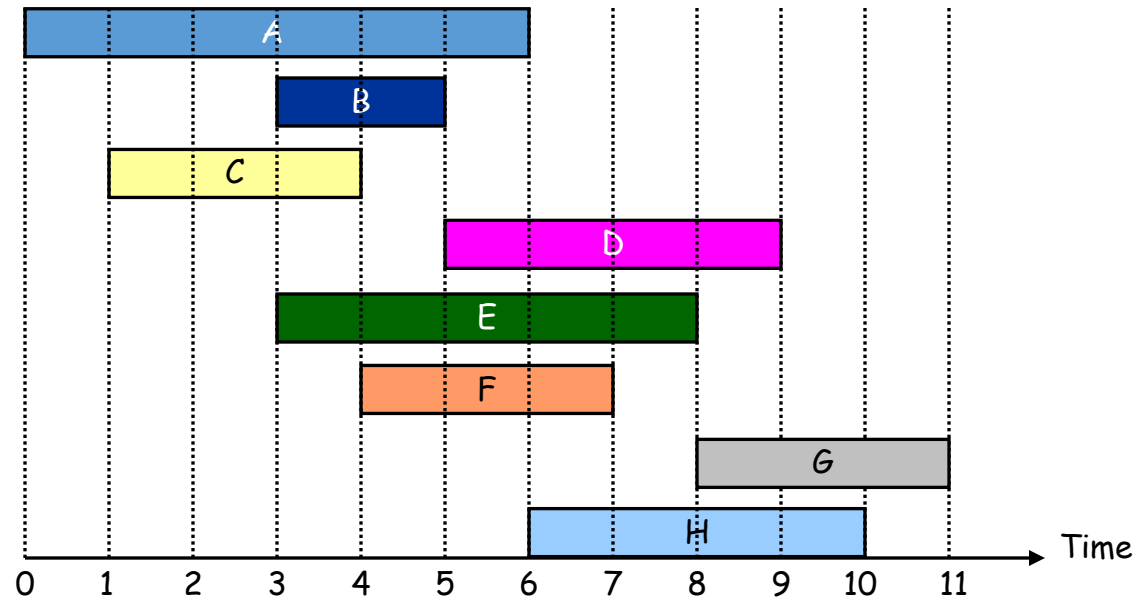
counterexample for fewest conflicts

Interval Scheduling: Greedy Algorithm

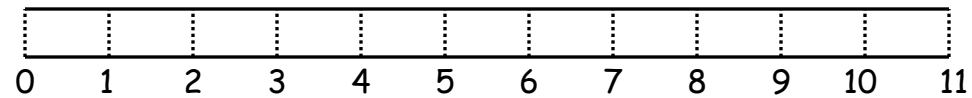
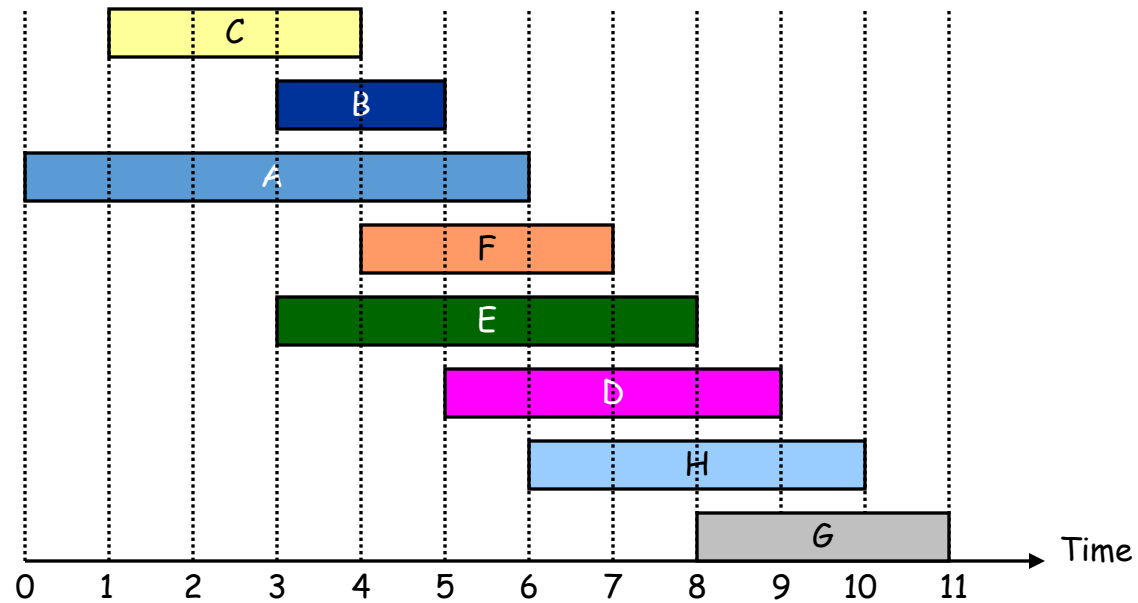
- **Greedy algorithm:** Consider jobs in increasing order of finish time. Take each job provided it's compatible with the ones already taken.

```
Sort jobs by finish times so that  $f_1 \leq f_2 \leq \dots \leq f_n$ .  
    set of jobs selected  
    ↙  
A ←  $\phi$   
for j = 1 to n {  
    if (job j compatible with A)  
        A ← A  $\cup$  {j}  
}  
return A
```

Interval Scheduling: Example

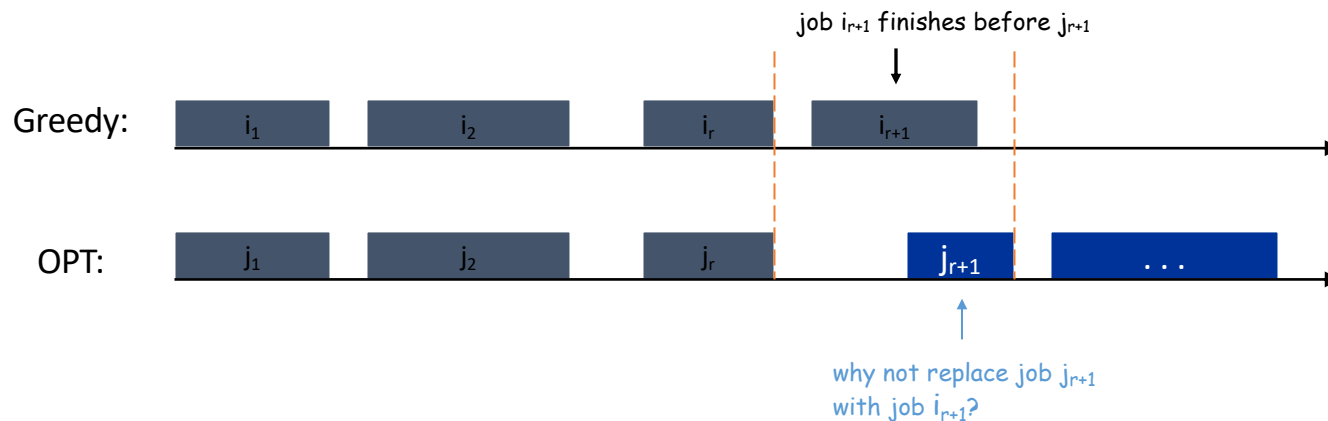


Interval Scheduling: Example



Interval Scheduling: Proof

- Theorem. Greedy algorithm is optimal.
- Pf. (by contradiction) greedy-stays-ahead approach
 - Assume greedy is not optimal, and let's see what happens.
 - Let i_1, i_2, \dots, i_k denote set of intervals selected by greedy.
 - Let j_1, j_2, \dots, j_m denote set of intervals in the optimal solution with $i_1 = j_1, i_2 = j_2, \dots, i_r = j_r$ for the largest possible value of r .



Interval Scheduling: Implementation

- Finding the next earliest finishing time of remaining intervals via linear search:
 - $O(n^2)$.
- Sorting
 - Sort all the requests by finishing time — $O(n \log n)$
 - Iterate through the sorted array taking the next legal request — $O(n)$
 - $O(n \log n)$

Summary

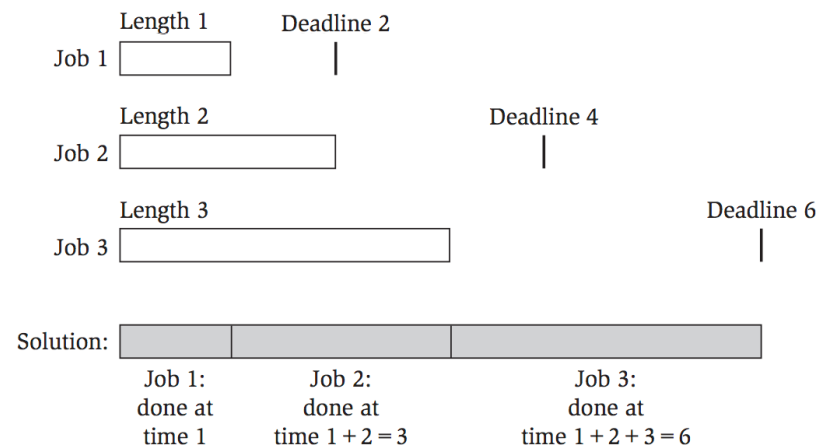
- Scheduling problems are often amenable to greedy approach
- But there may be many greedy choices and it is important to select the right one
- **Main Takeaway:** Greedy-stays-ahead is a useful proof approach

Greedy Algorithms

- a. Interval Scheduling– Greedy Stays Ahead
- b. Minimum Lateness Scheduling

Minimum Lateness Scheduling

- **Input:** n jobs with **length** t_i and **deadline** d_i
 - Simplifying assumption: all deadlines are distinct
- **Output:** a minimum--lateness schedule for the jobs
 - Can only do one job at a time, no overlap
 - The **lateness of job i** is $\max\{f_i - d_i, 0\}$
 - The **lateness of a schedule** is $\max\{\max_i\{f_i - d_i\}, 0\}$



Possible Greedy Rules

- Choose the shortest job first ($\min t_i$)?
- Choose the most urgent job first ($\min d_i - t_i$)?
- Others?

Greedy Algorithm: Earliest Deadline First

- Sort jobs so that $d_1 \leq d_2 \leq \dots \leq d_n$
- For $i = 1, \dots, n$:
 - Schedule job i right after job $i - 1$ finishes

Exchange Argument

- G = greedy schedule, O = some other schedule
- Exchange Argument:
 - We can transform O to G by exchanging pairs of jobs
 - No exchange increases the lateness of O
 - Therefore, the lateness of G is at most that of O
 - G has the minimum possible lateness

Exchange Argument

- G = greedy schedule, O = (supposedly) optimal schedule
- We say that two jobs i, j are **inverted** in O if $d_i < d_j$ but j comes before i in the schedule
 - Observation: greedy has no inversions



Example: two jobs

- Two jobs with deadlines $d_1 < d_2$ and lengths t_1, t_2
- Greedy schedule: **1, 2**
- O : **2, 1** (inversion)

Lateness of O : $\max(t_2 - d_2, t_1 + t_2 - d_1) = t_1 + t_2 - d_1$

Flipping them: $\max(t_1 + t_2 - d_2, t_1 - d_1) \leq t_1 + t_2 - d_1$



Exchange Argument

- We say that two jobs i, j are **inverted** in O if $d_i < d_j$ but j comes before i
- **Claim: an optimal schedule has no inversions**
 - Step 1: suppose O has an inversion, then it has an inversion i, j where i, j are **consecutive**
 - Step 2: if i, j are a consecutive jobs that are inverted then flipping them only reduces the lateness



Exchange Argument

- If i, j are a consecutive jobs that are inverted then flipping them only reduces the lateness



Exchange Argument

- If i, j are a consecutive jobs that are inverted then flipping them only reduces the lateness
 - Does not change the lateness of the other jobs
 - Let's assume these jobs have $d_i < d_j$ and lengths t_i, t_j
 - Assume job 2 starts at time s in schedule O .



Exchange Argument

- If i, j are a consecutive jobs that are inverted then flipping them only reduces the lateness
 - Does not change the lateness of the other jobs
 - Let's assume these jobs have $d_i < d_j$ and lengths t_i, t_j
 - Assume job 2 starts at time s in schedule O .

Max lateness of 1 and 2 before flipping:

$$\max(s + t_j - d_j, s + t_j + t_i - d_i) = s + t_i + t_j - d_i$$



Exchange Argument

- If i, j are a consecutive jobs that are inverted then flipping them only reduces the lateness
 - Does not change the lateness of the other jobs
 - Let's assume these jobs have $d_i < d_j$ and lengths t_i, t_j
 - Assume job 2 starts at time s in schedule O .

Max lateness of 1 and 2 before flipping:

$$\max(s + t_j - d_j, s + t_j + t_i - d_i) = s + t_i + t_j - d_i$$

Max lateness of 1 and 2 after flipping:

$$\max(s + t_i + t_j - d_j, s + t_i - d_i) < s + t_i + t_j - d_i$$



Exchange Argument



Exchange Argument

- We say that two jobs i, j are **inverted** in O if $d_i < d_j$ but j comes before i
- **Claim: an optimal schedule has no inversions**
 - Step 1: suppose O has an inversion, then it has an inversion i, j where i, j are **consecutive**
 - Step 2: if i, j are consecutive jobs that are inverted then **flipping them only reduces the lateness**
- G is the unique schedule with no inversions, $\text{lateness}(G) \leq \text{lateness}(O)$

