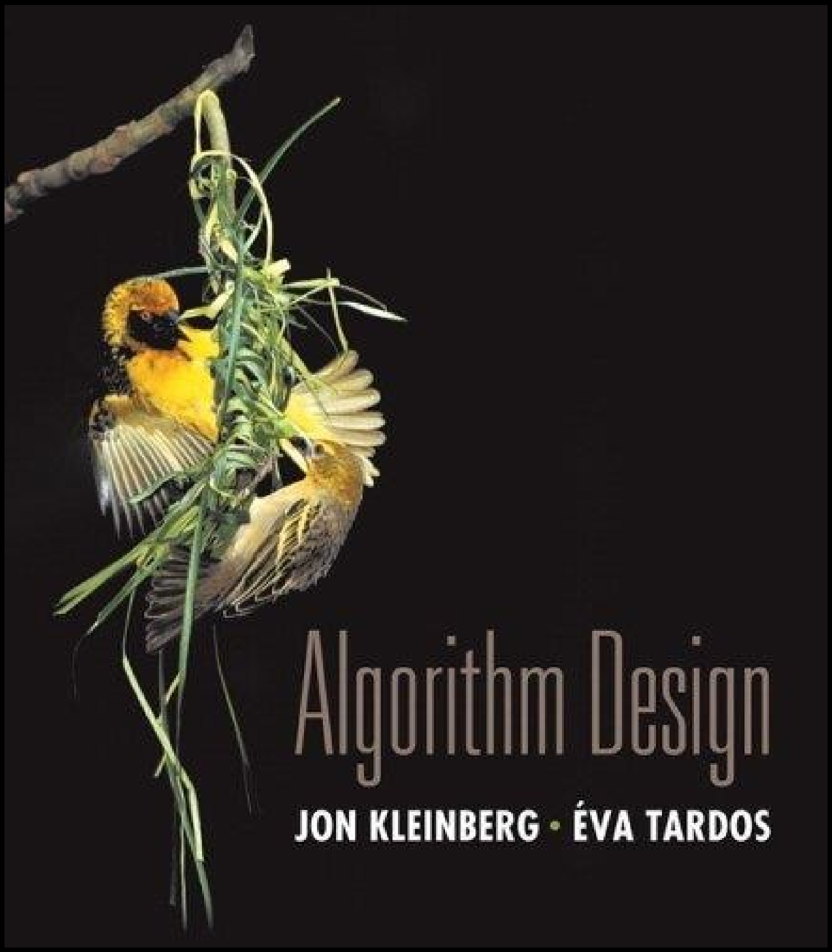


Chapter 4

Greedy Algorithms

PEARSON
Addison
Wesley

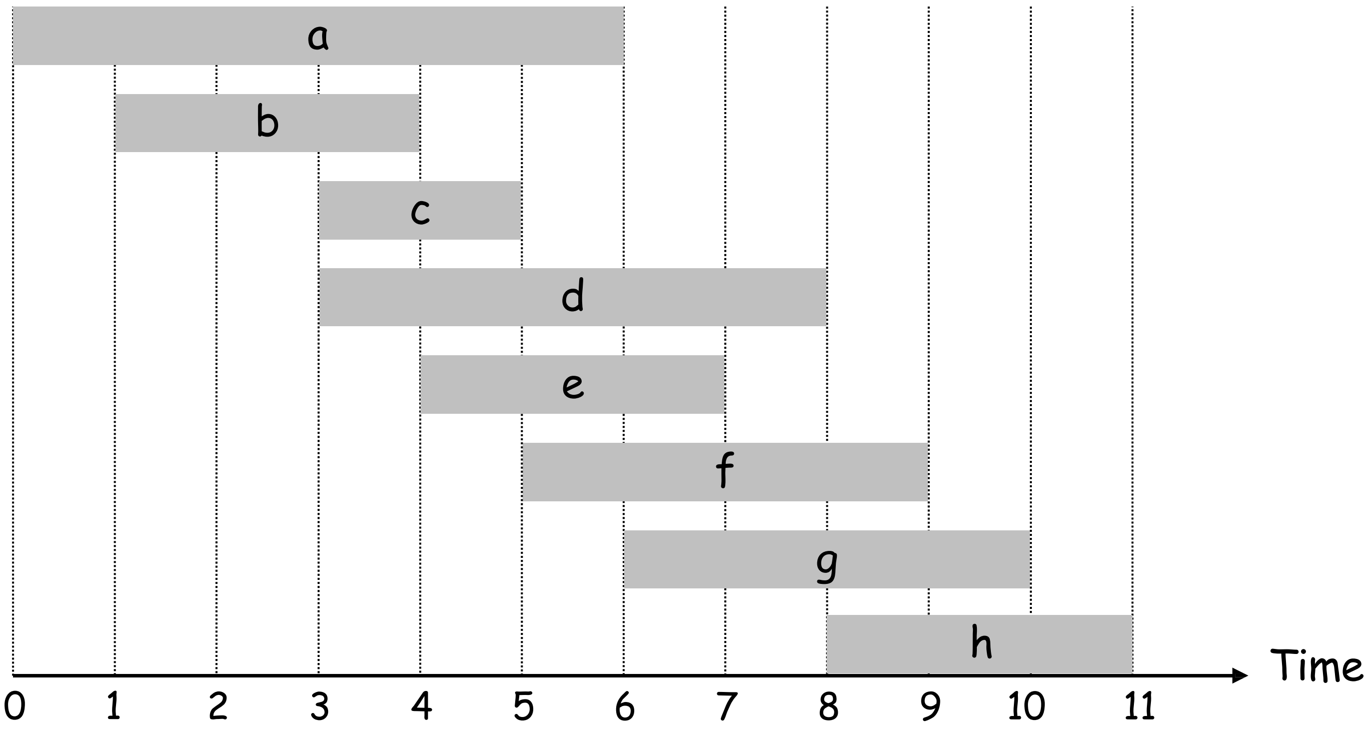
Slides by Kevin Wayne.

4.1 Interval Scheduling

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: Greedy Algorithms

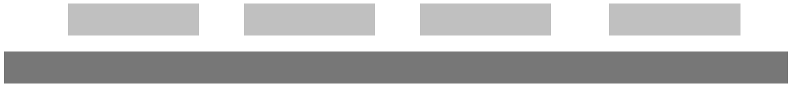
Greedy template. Consider jobs in some order. Take each job provided it's compatible with the ones already taken.

- [Earliest start time] Consider jobs in ascending order of start time s_j.

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.
  - If the earliest request *i* is for a very long interval, then by accepting request *i* we may have to reject a lot of requests for shorter time intervals.



Interval Scheduling: Greedy Algorithms

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

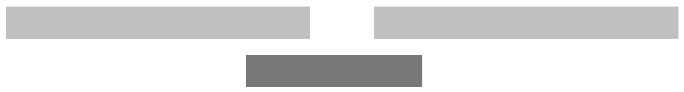
- [Shortest interval] Consider jobs in ascending order of interval length $f_j - s_j$.

Interval Scheduling: Greedy Algorithms

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

- [Shortest interval] Consider jobs in ascending order of interval length $f_j - s_j$.

  - Accepting the short interval in the middle (see Figure) would prevent us from accepting the other two, which form an optimal solution.



Interval Scheduling: Greedy Algorithms

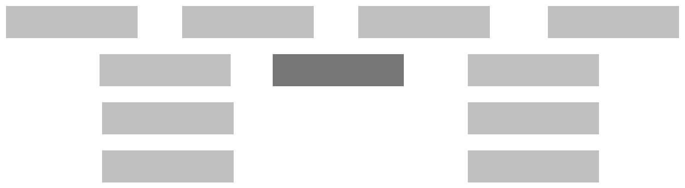
Greedy template. Consider jobs in some order. Take each job provided it's compatible with the ones already taken.

- [Fewest conflicts] For each request j, count the number of other requests c_j that are not compatible, and accept the request that has the fewest number of noncompatible requests (i.e. 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.

- [Fewest conflicts] For each request j, count the number of other requests c_j that are not compatible, and accept the request that has the fewest number of noncompatible requests (i.e. schedule in ascending order of conflicts c_j).
  - The unique optimal solution in the example is to accept the four requests in the top row. The greedy method accepts the middle request in the second row and so ensures a solution of size no greater than three.



Interval Scheduling: Greedy Algorithms

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

- [Earliest finish time] Consider jobs in ascending order of finish time f_j.

Interval Scheduling: Greedy Algorithm

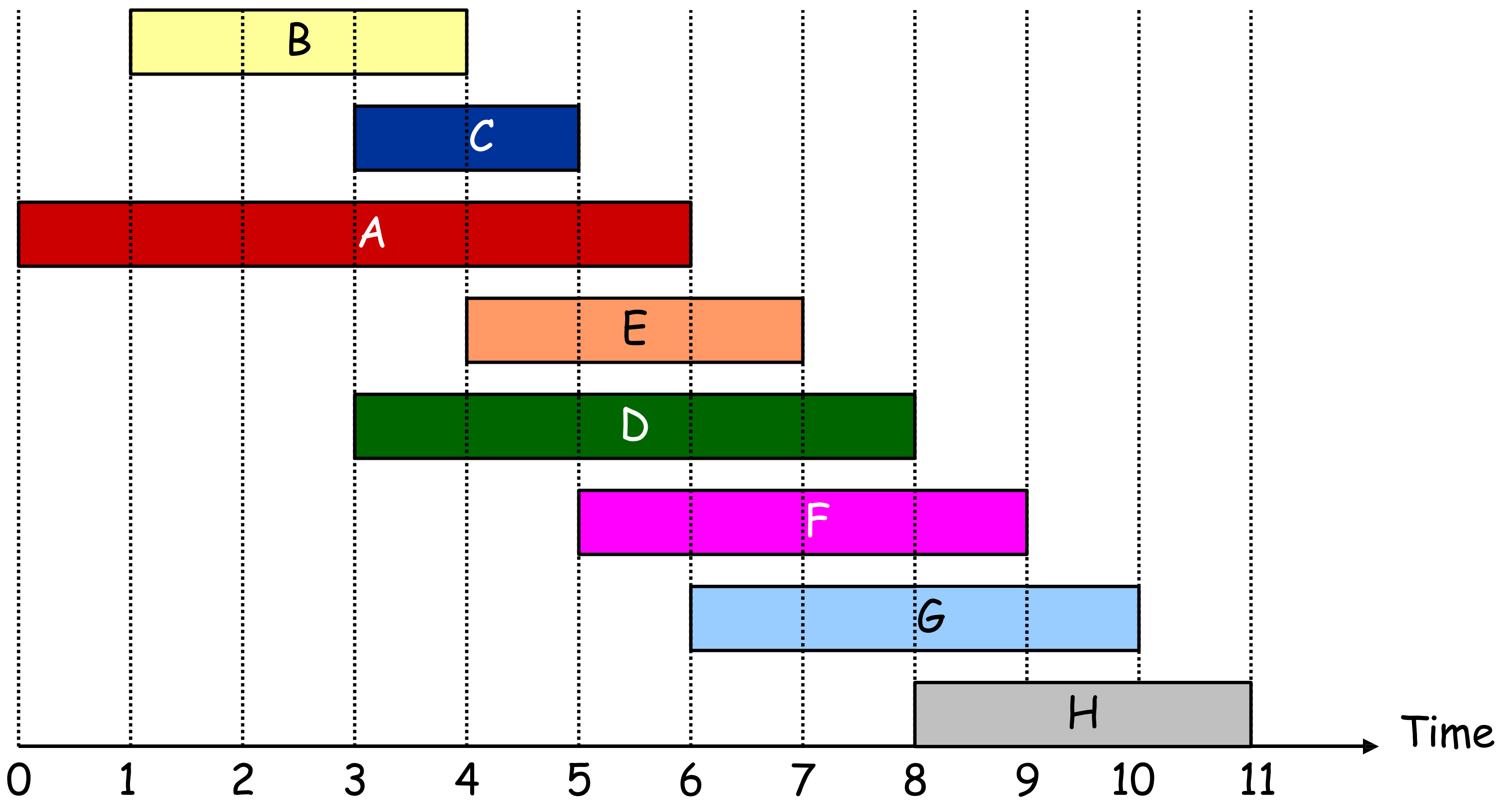
Greedy algorithm. Consider jobs in increasing order of finish time. Take each job provided it's compatible with the ones already taken.

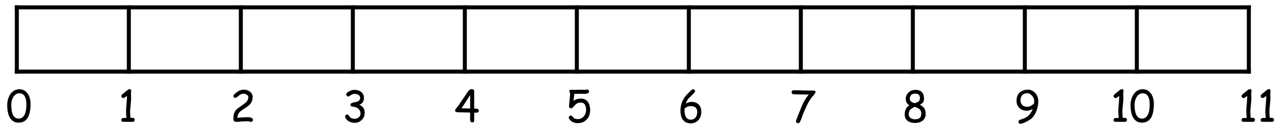
```
Sort jobs by finish times so that $f_1 \le f_2 \le ... \le f_n$.

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

Implementation. O(n log n).

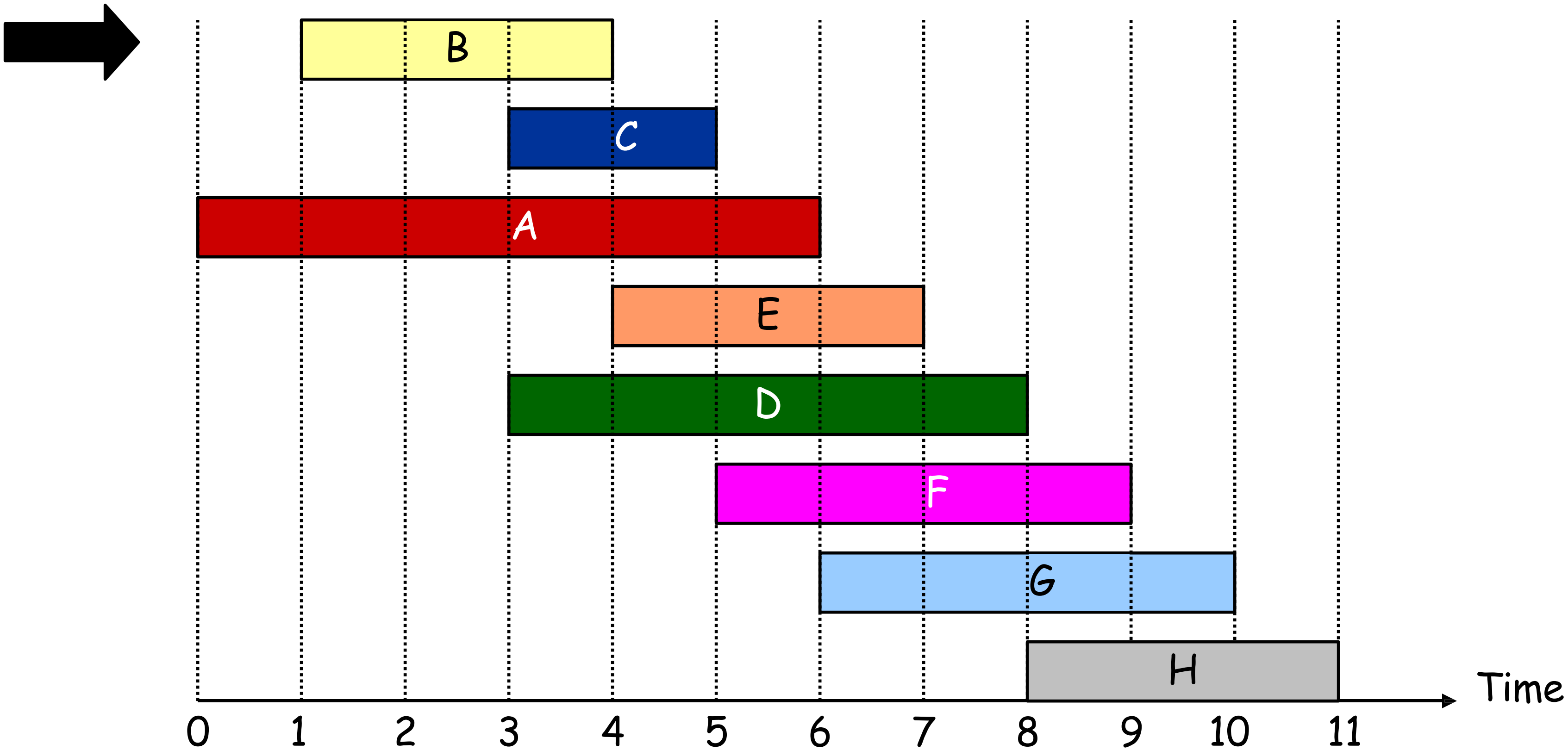
- Remember job j* that was added last to A.
- Job j is compatible with A if $s_j \ge f_{j^*}$.

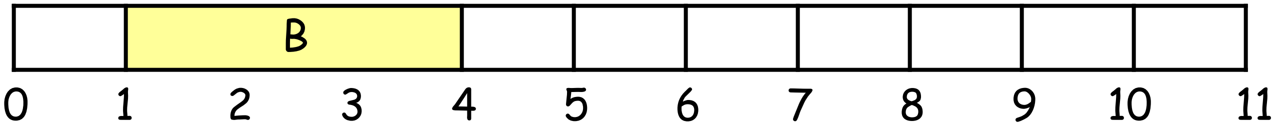
Interval Scheduling



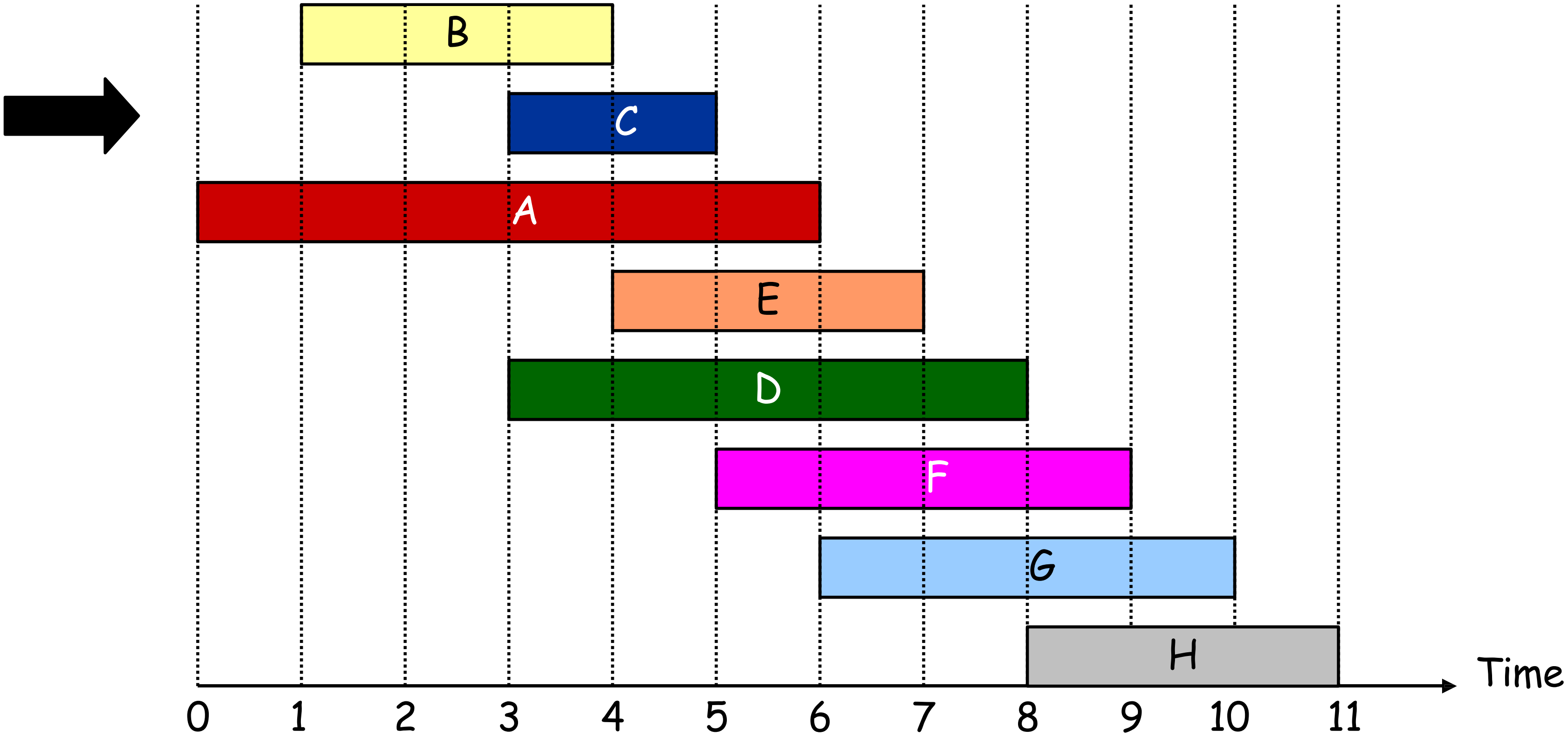


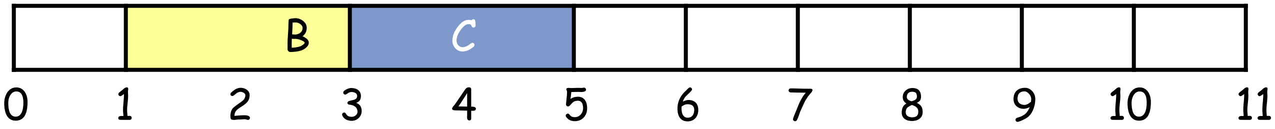
Interval Scheduling



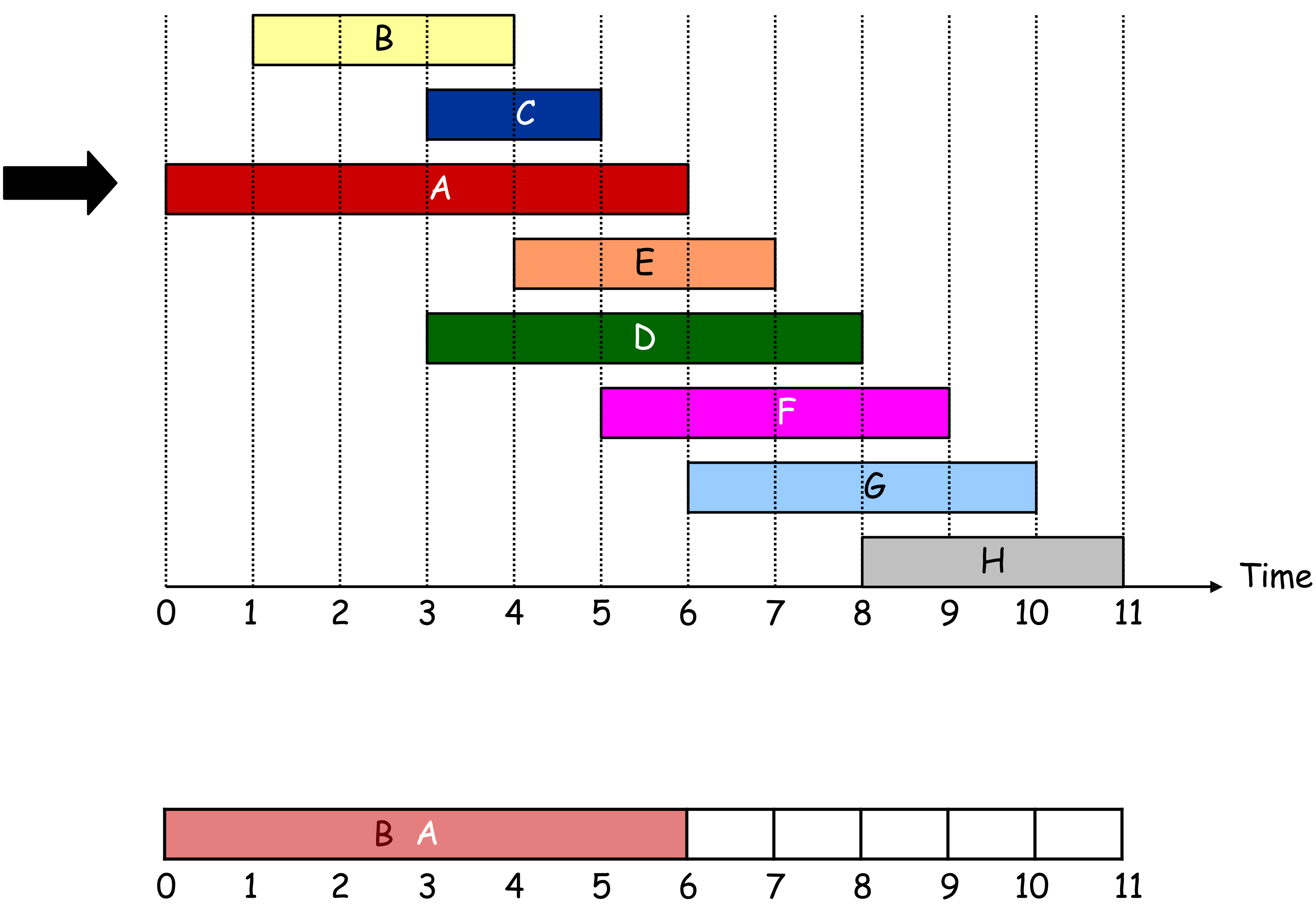


Interval Scheduling

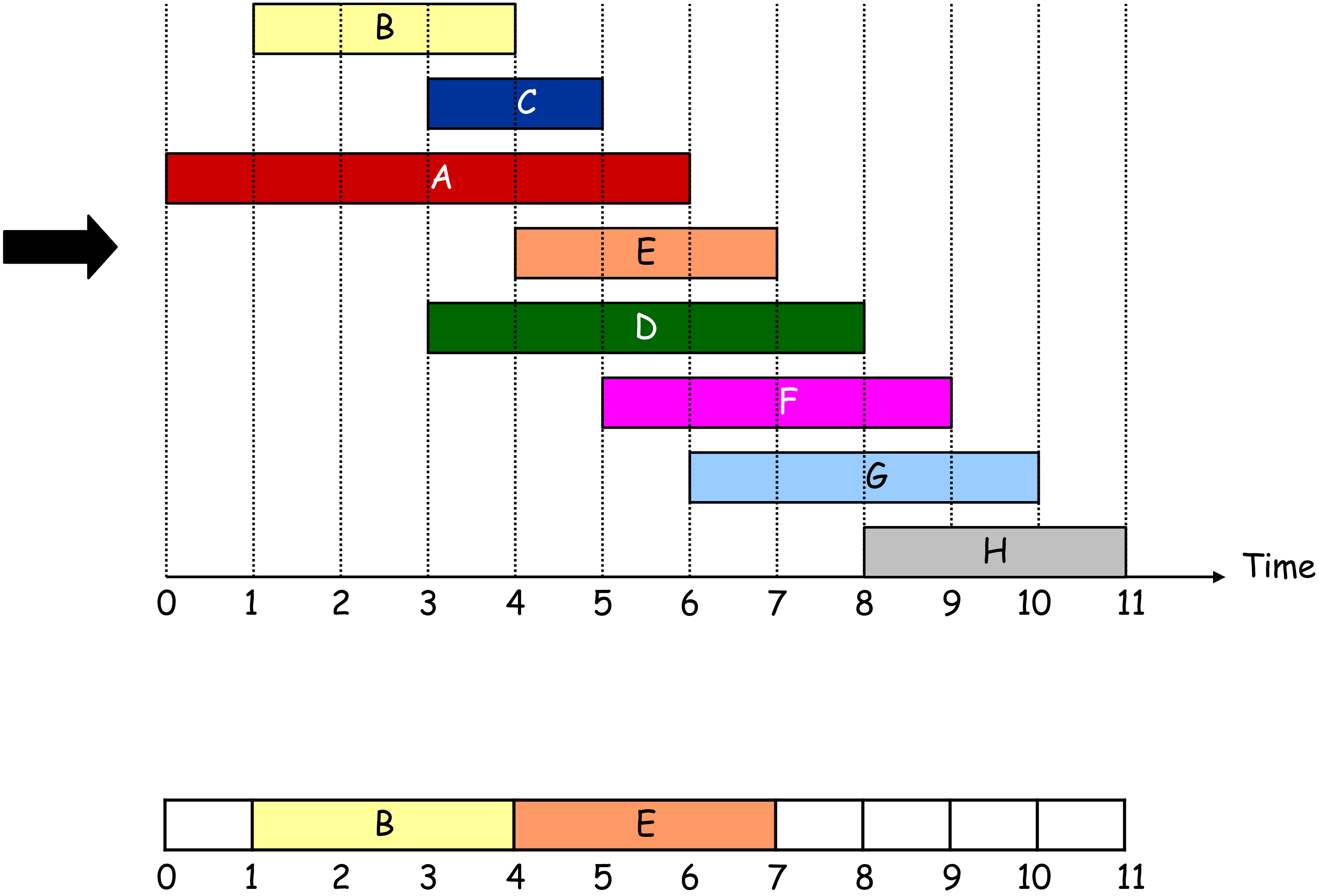




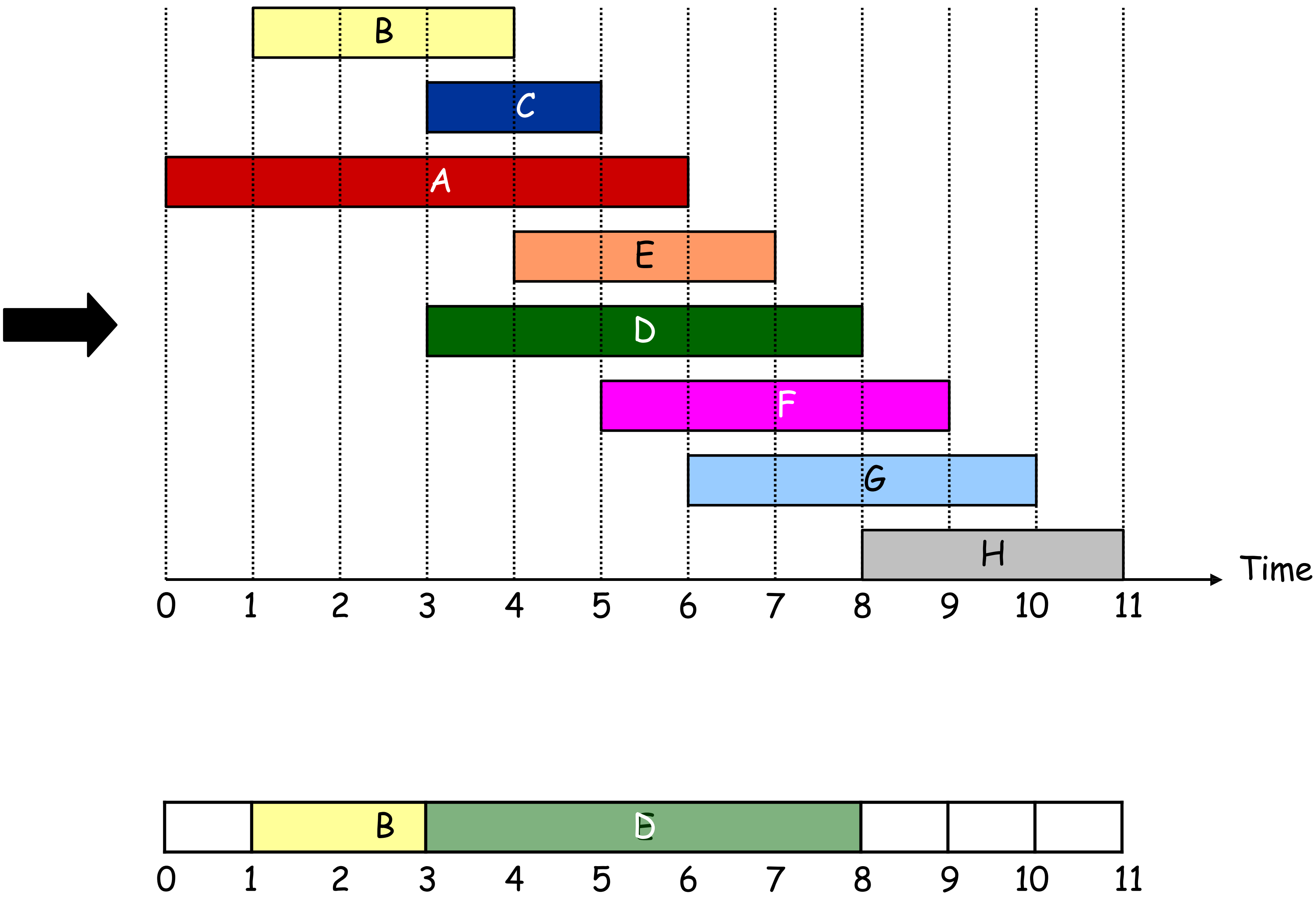
Interval Scheduling



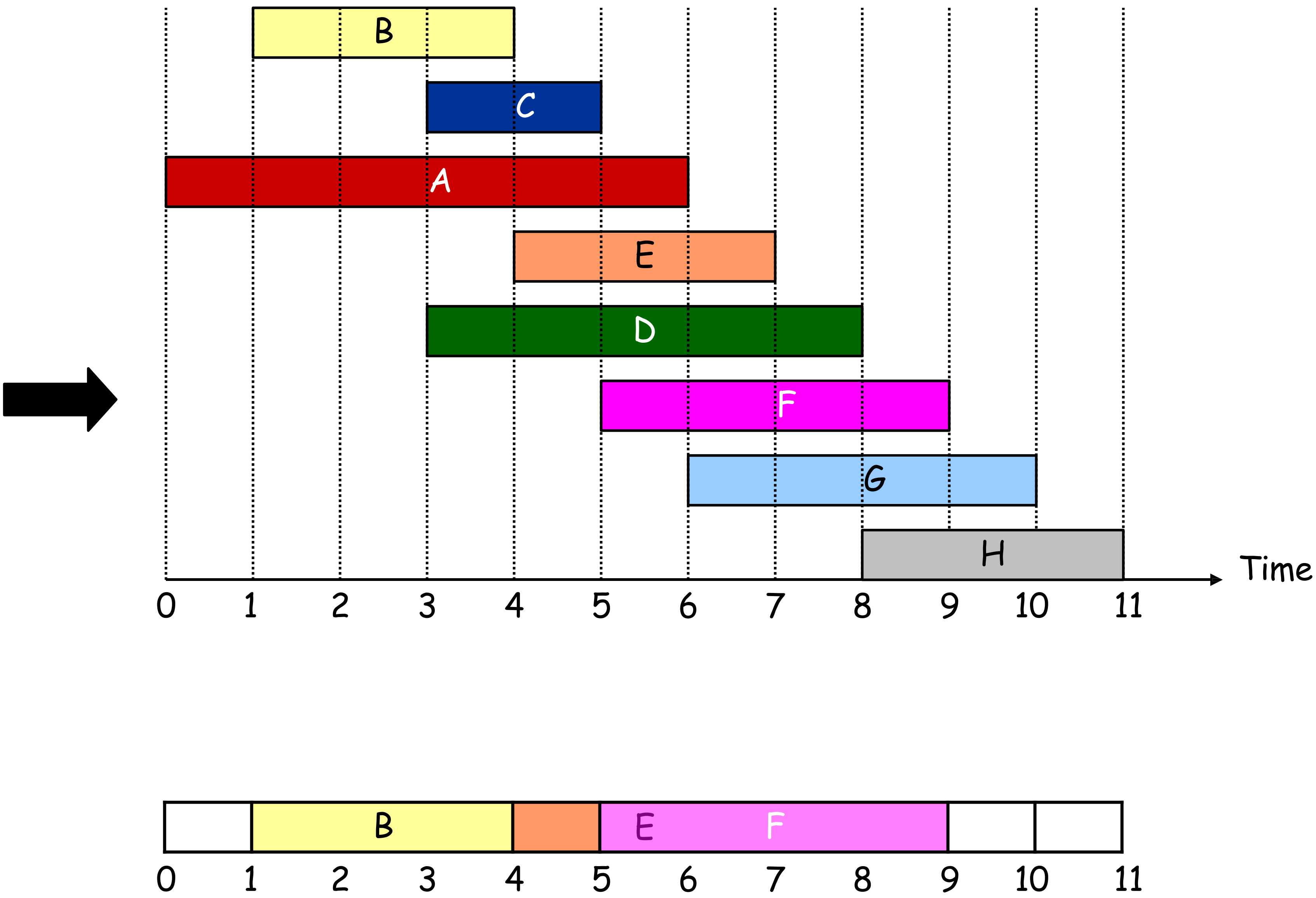
Interval Scheduling



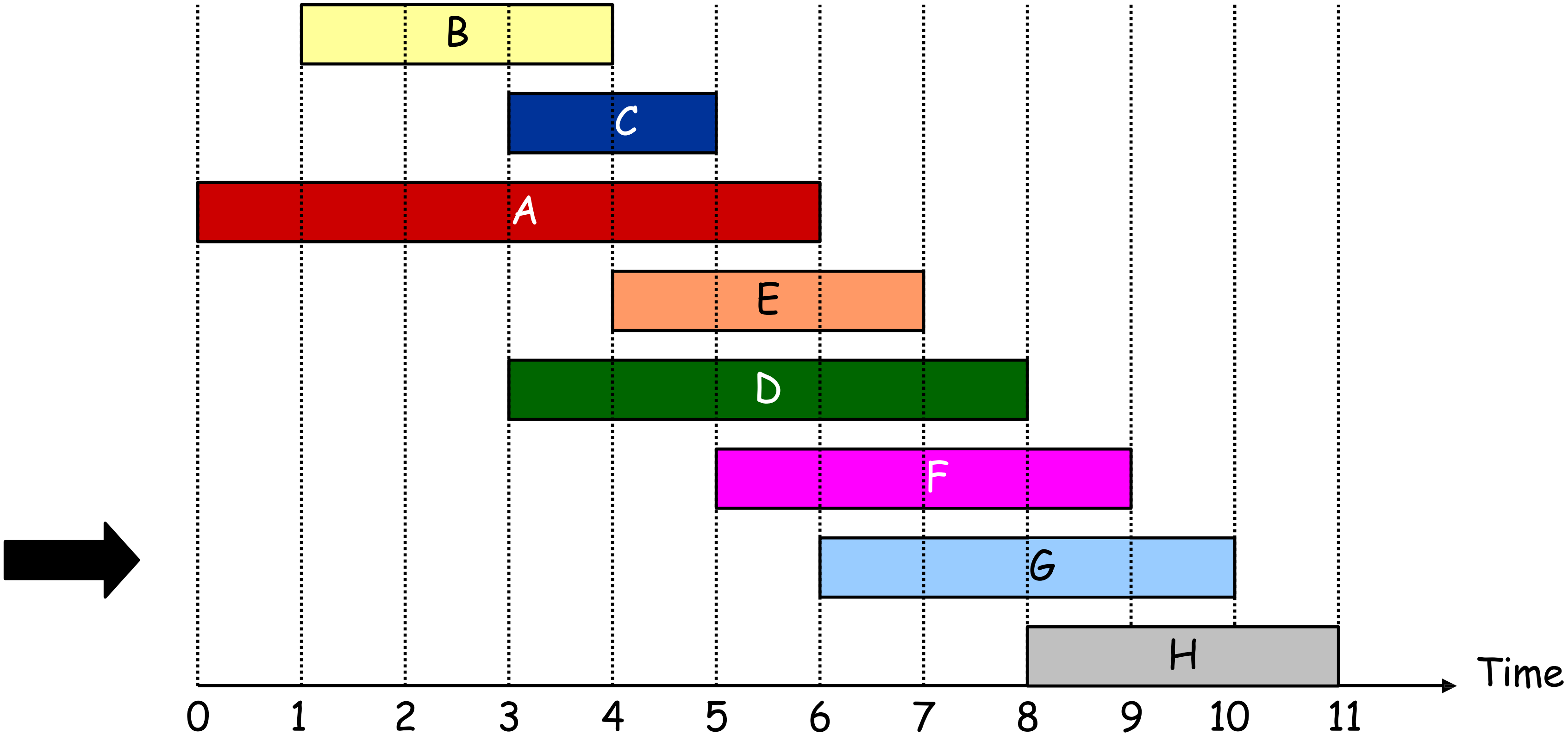
Interval Scheduling

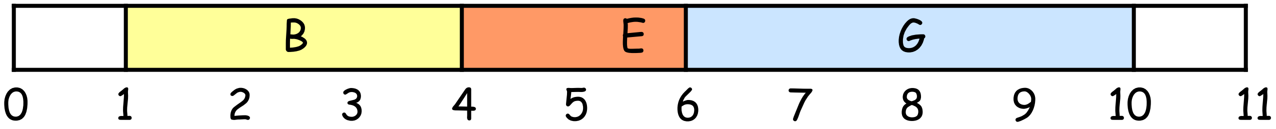


Interval Scheduling

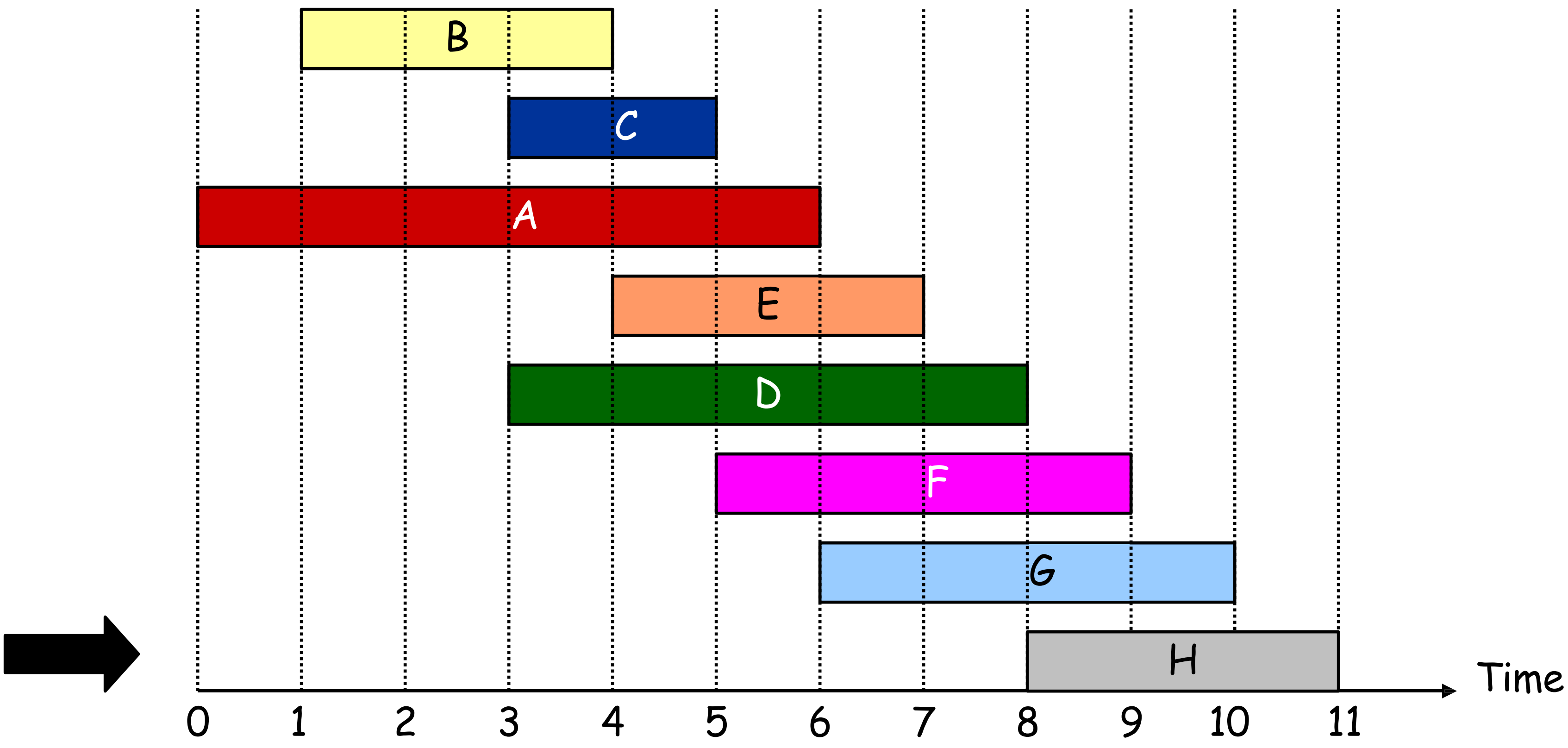


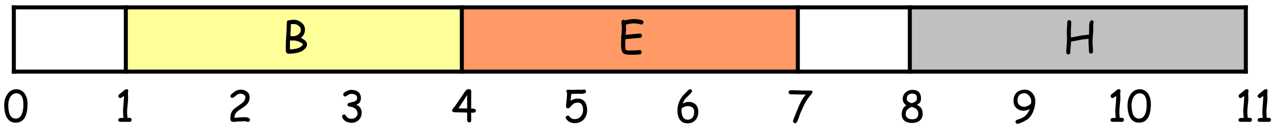
Interval Scheduling





Interval Scheduling





Interval Scheduling: Analysis

Theorem. Greedy algorithm is optimal.

Interval Scheduling: Analysis

Theorem. Greedy algorithm is optimal.

Pf. (by contradiction)

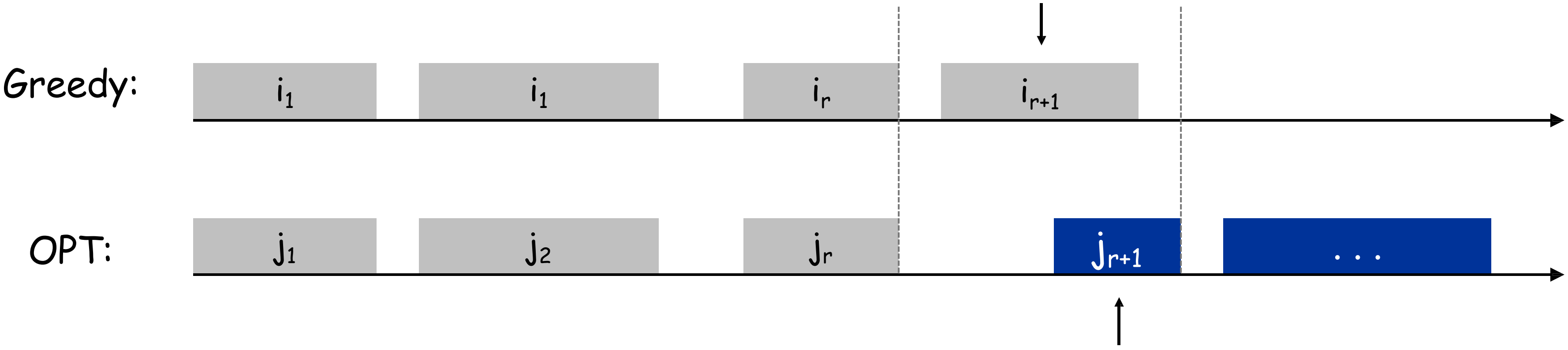
Interval Scheduling: Analysis

Theorem. Greedy algorithm is optimal.

Pf. (by contradiction)

- Assume greedy is not optimal, and let's see what happens.
- Let i_1, i_2, ... i_k denote set of jobs selected by greedy.
- Let j_1, j_2, ... 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.

job i_{r+1} finishes before j_{r+1}



why not replace job j_{r+1} with job i_{r+1}?

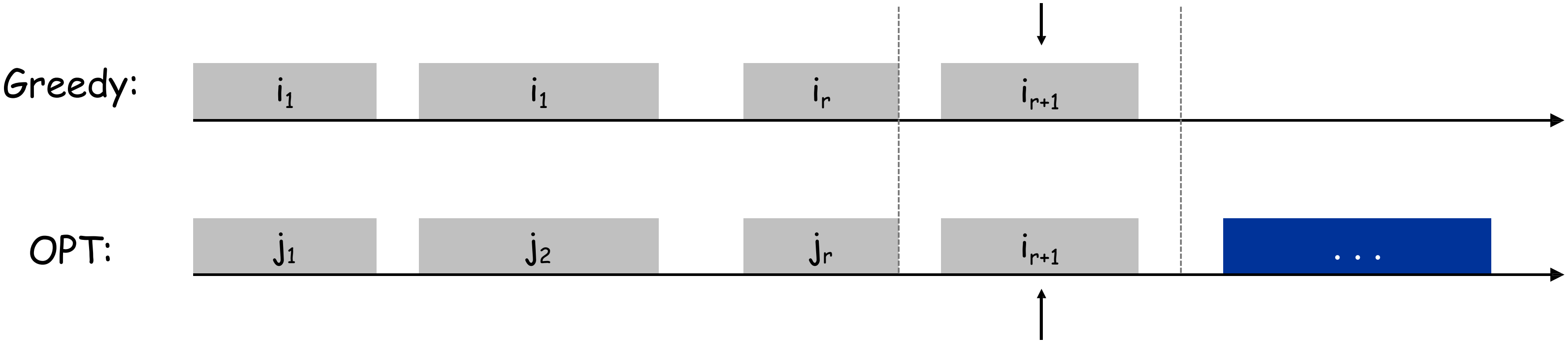
Interval Scheduling: Analysis

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.

job i_{r+1} finishes before j_{r+1}



solution still feasible and optimal,
but contradicts maximality of r.