南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Algorithm Design and Analysis (H)

CS216

Prof. Shiqi Yu (于仕琪)

yusq@sustech.edu.cn

http://faculty.sustech.edu.cn/yusq/

PEARSON
Addison Wesley

Slides by Kevin Wayne. Copyright © 2005 Pearson-Addison Wesley. All rights reserved.

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Greedy Algorithms

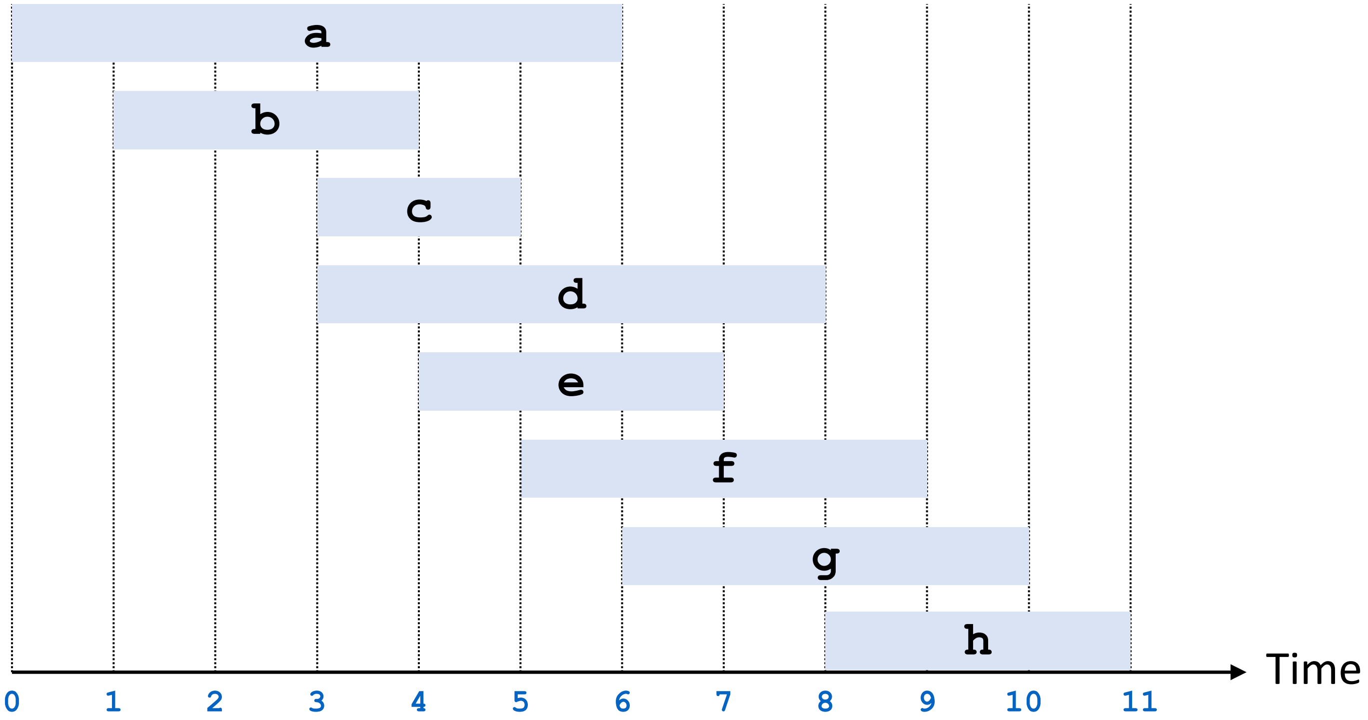
Greedy algorithms

- Build up a solution in small steps,
- Choose a decision at each step myopically to optimize some underlying criterion.
- May not produce an optimal solution,
- But can yield locally optimal solutions that approximate a globally optimal solution in a reasonable amount of time.

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.



Slides by Kevin Wayne. Copyright © 2005 Pearson-Addison Wesley. All rights reserved.

Interval Scheduling: Greedy Algorithms

- **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.

Slides by Kevin Wayne. Copyright © 2005 Pearson-Addison Wesley. All rights reserved.

Interval Scheduling: Greedy Algorithms

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

counterexample for earliest start time

counterexample for shortest interval

counterexample for fewest conflicts

Slides by Kevin Wayne. Copyright © 2005 Pearson-Addison Wesley. All rights reserved.

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.

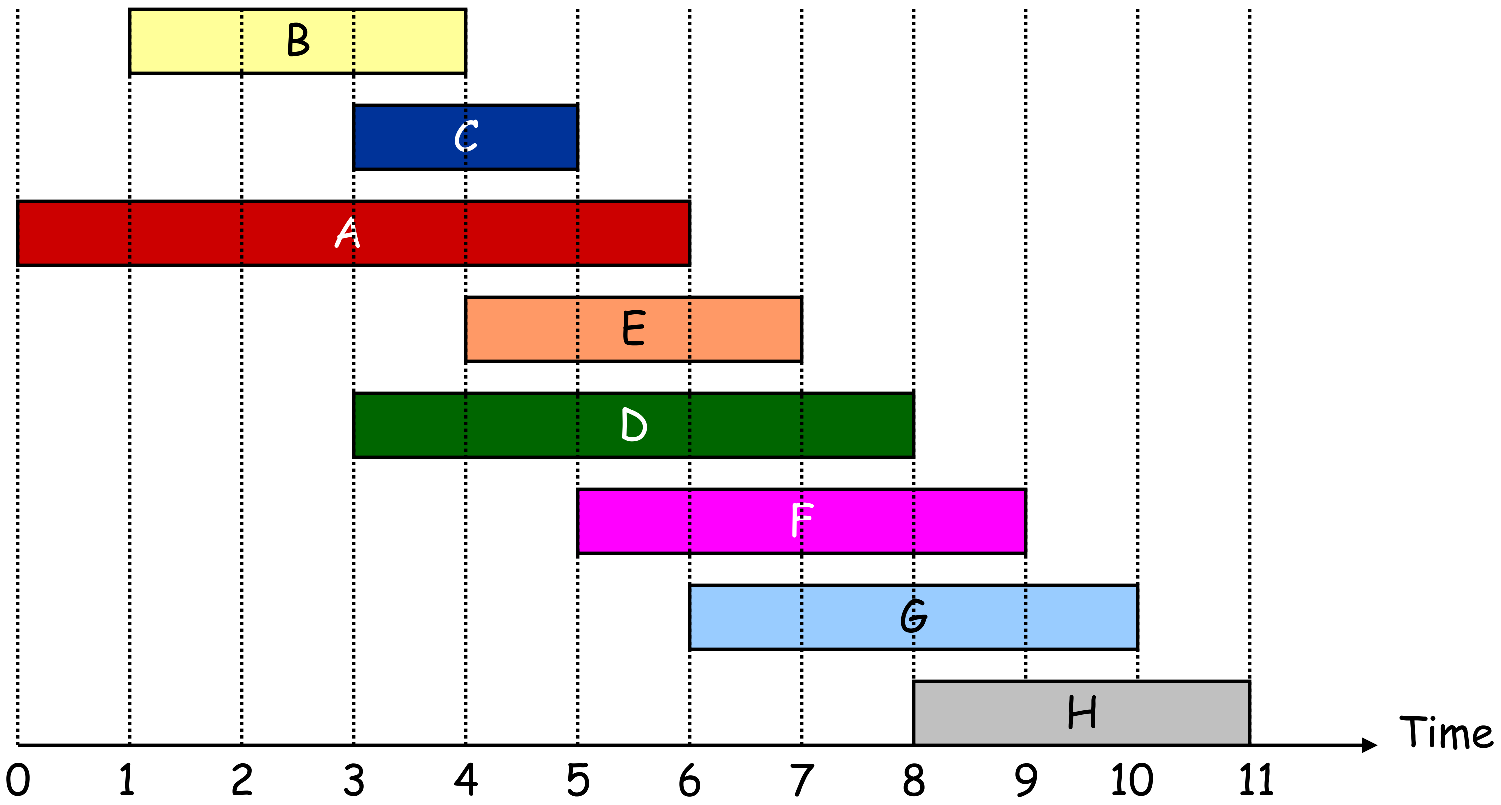
```
Sort jobs by finish times so that f1 ≤ f2 ≤ ... ≤ fn.

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

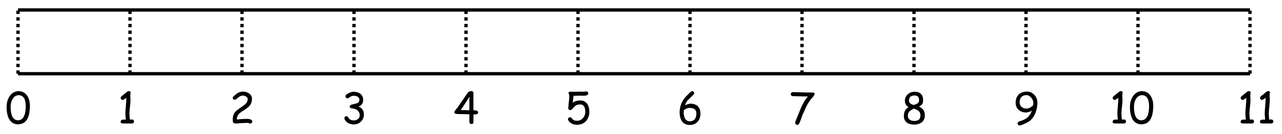
set of jobs selected

- Implementation. O(n log n).
  - Remember job j* that was added last to A.
  - Job j is compatible with A if $s_j \geq f_{j^*}$.

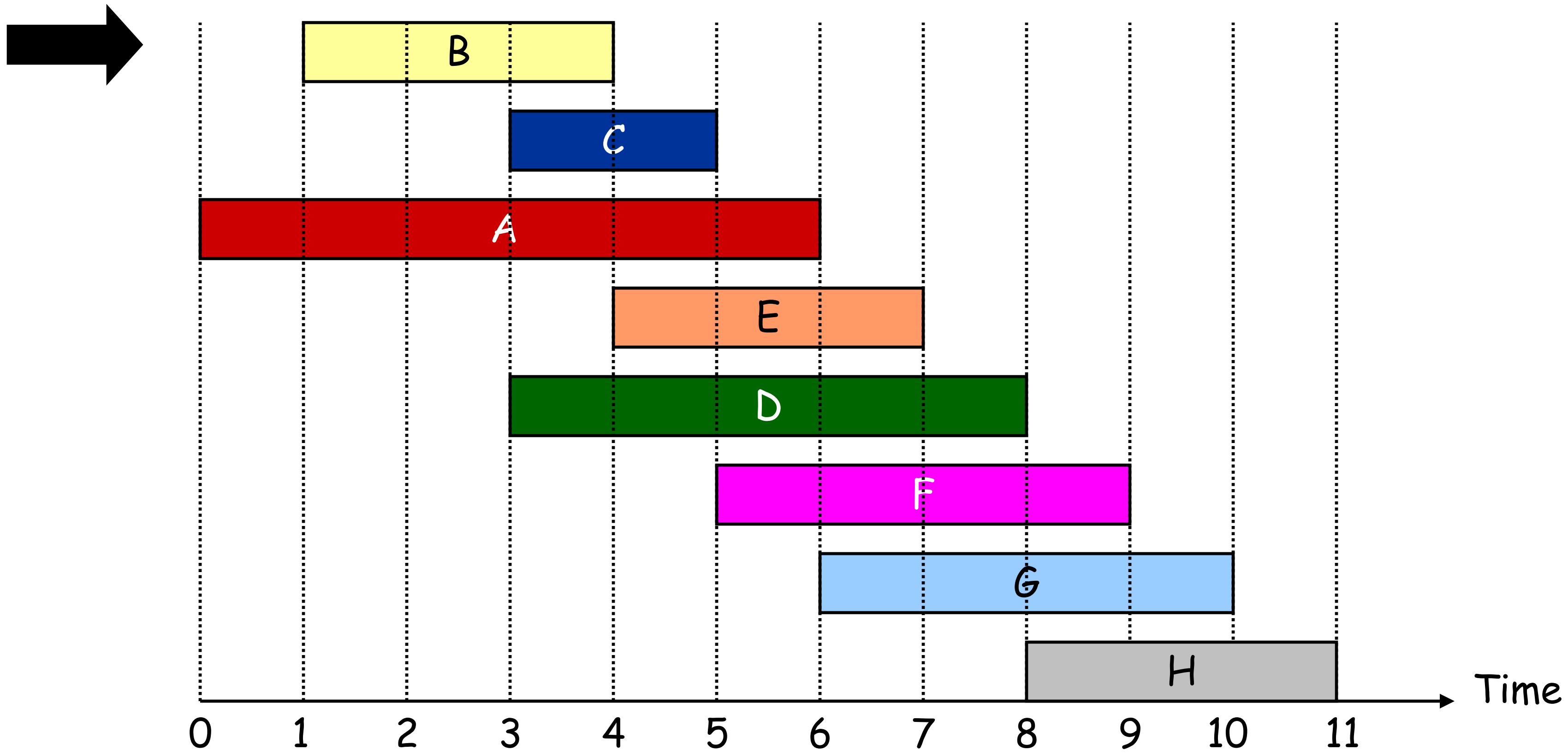
Slides by Kevin Wayne. Copyright © 2005 Pearson-Addison Wesley. All rights reserved.

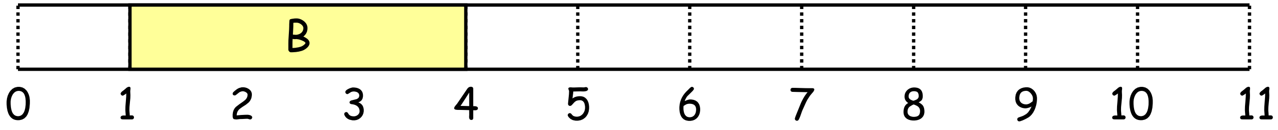
Interval Scheduling



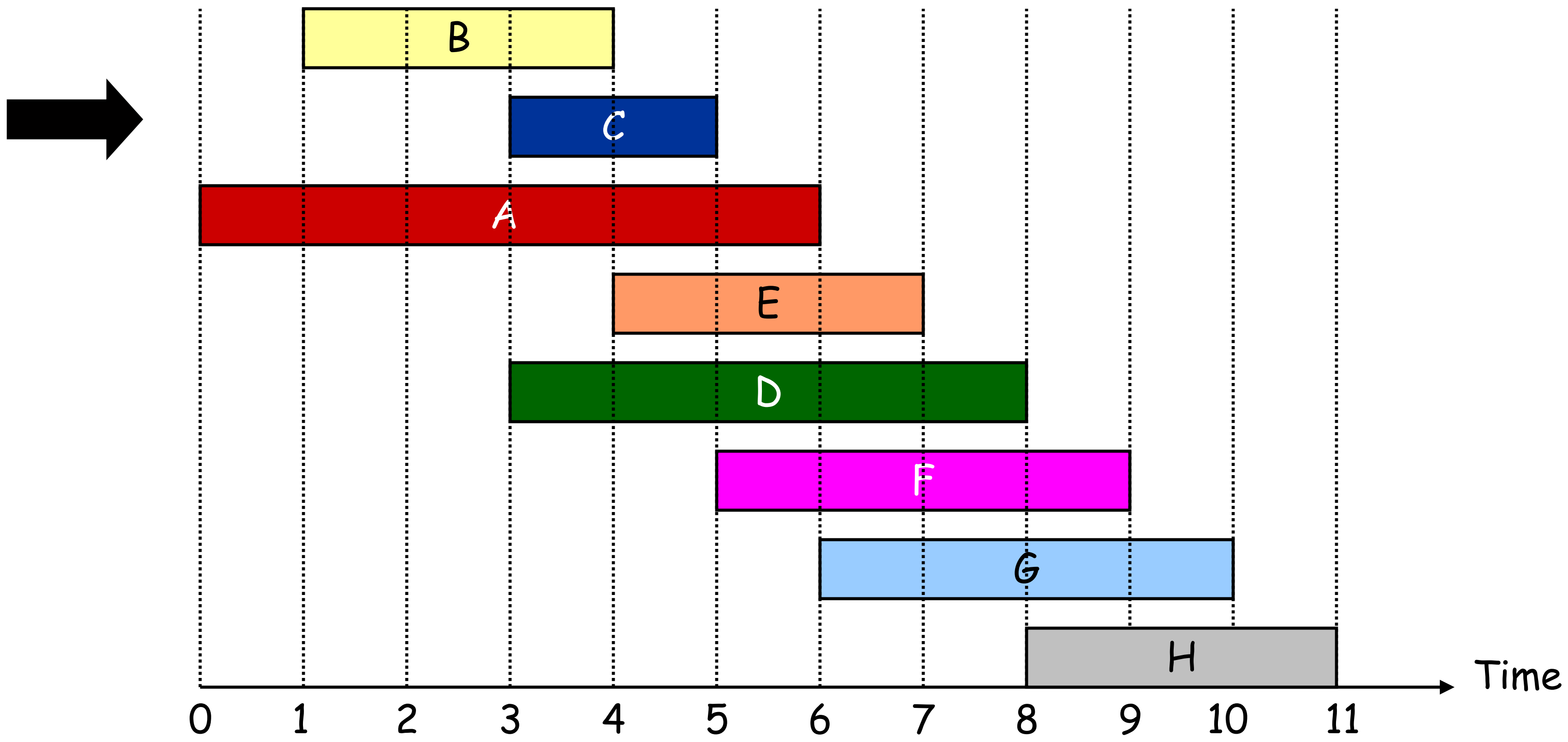


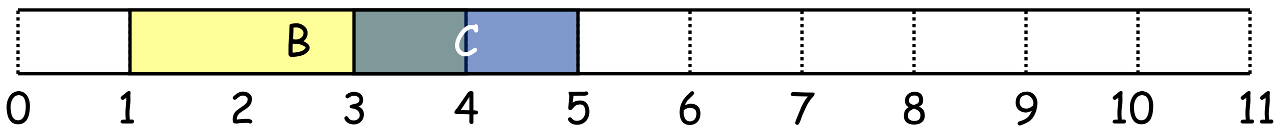
Interval Scheduling



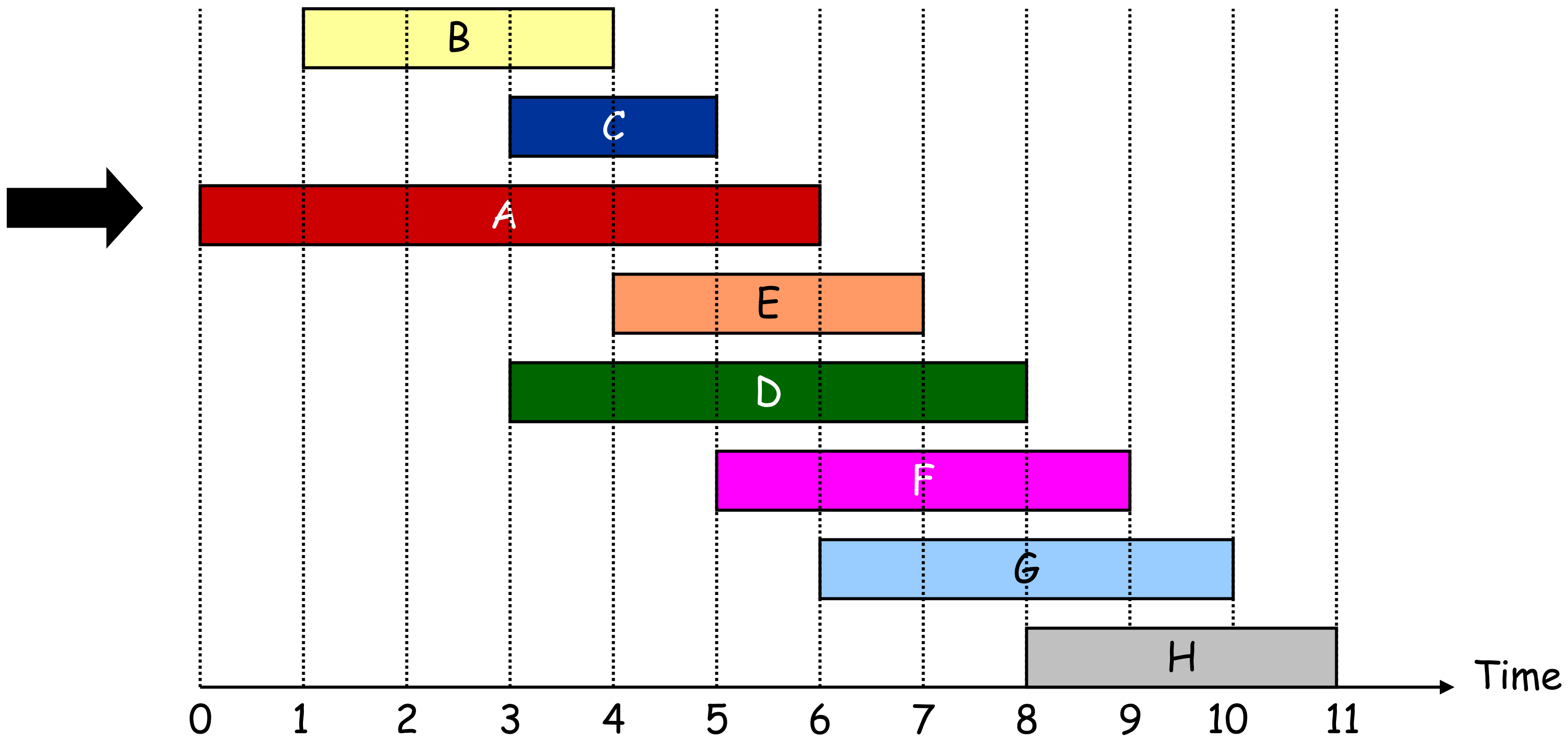


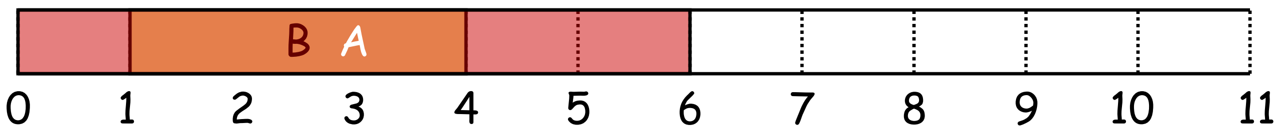
Interval Scheduling



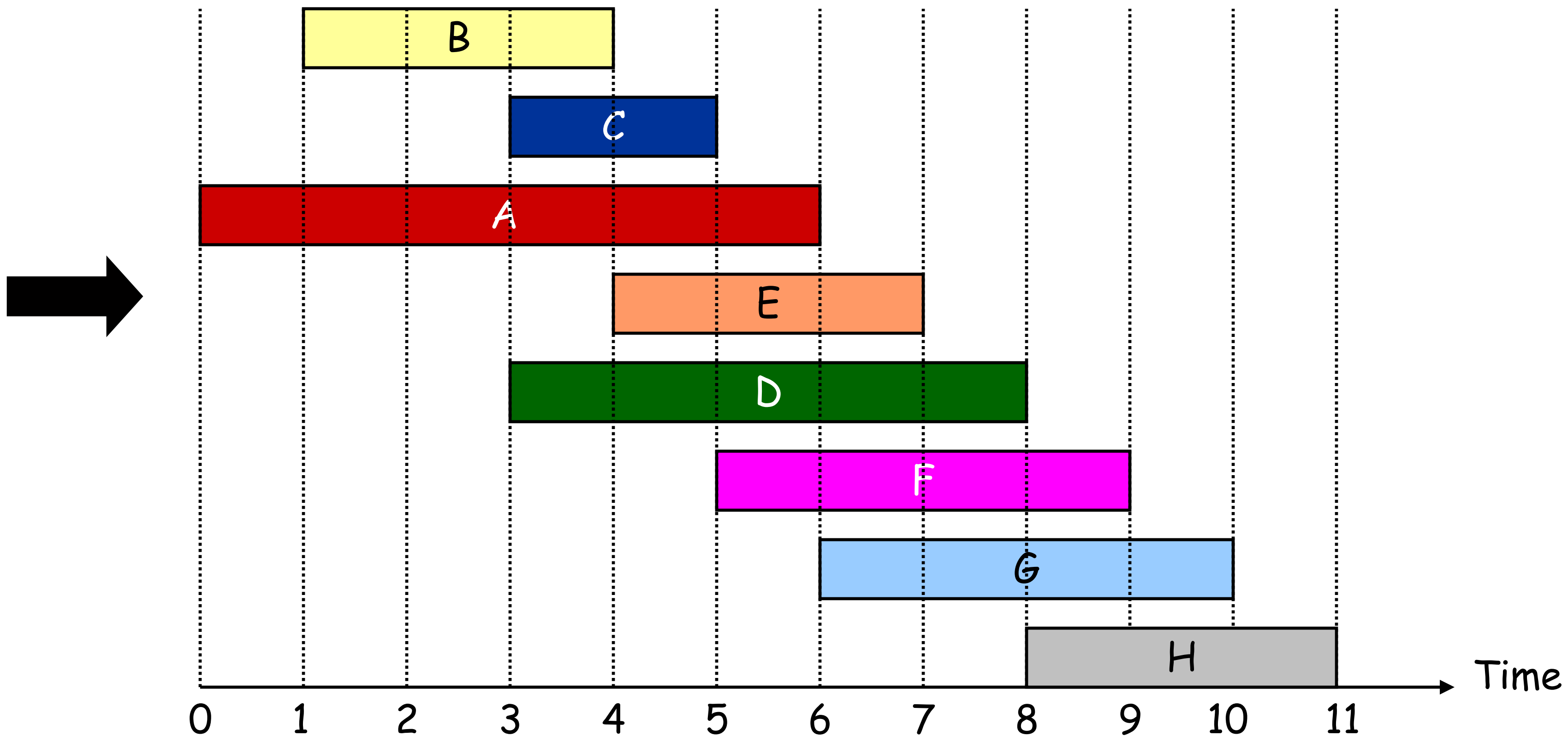


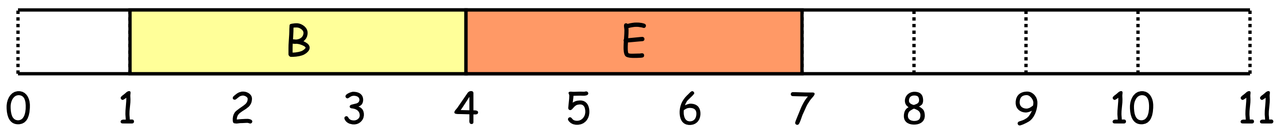
Interval Scheduling



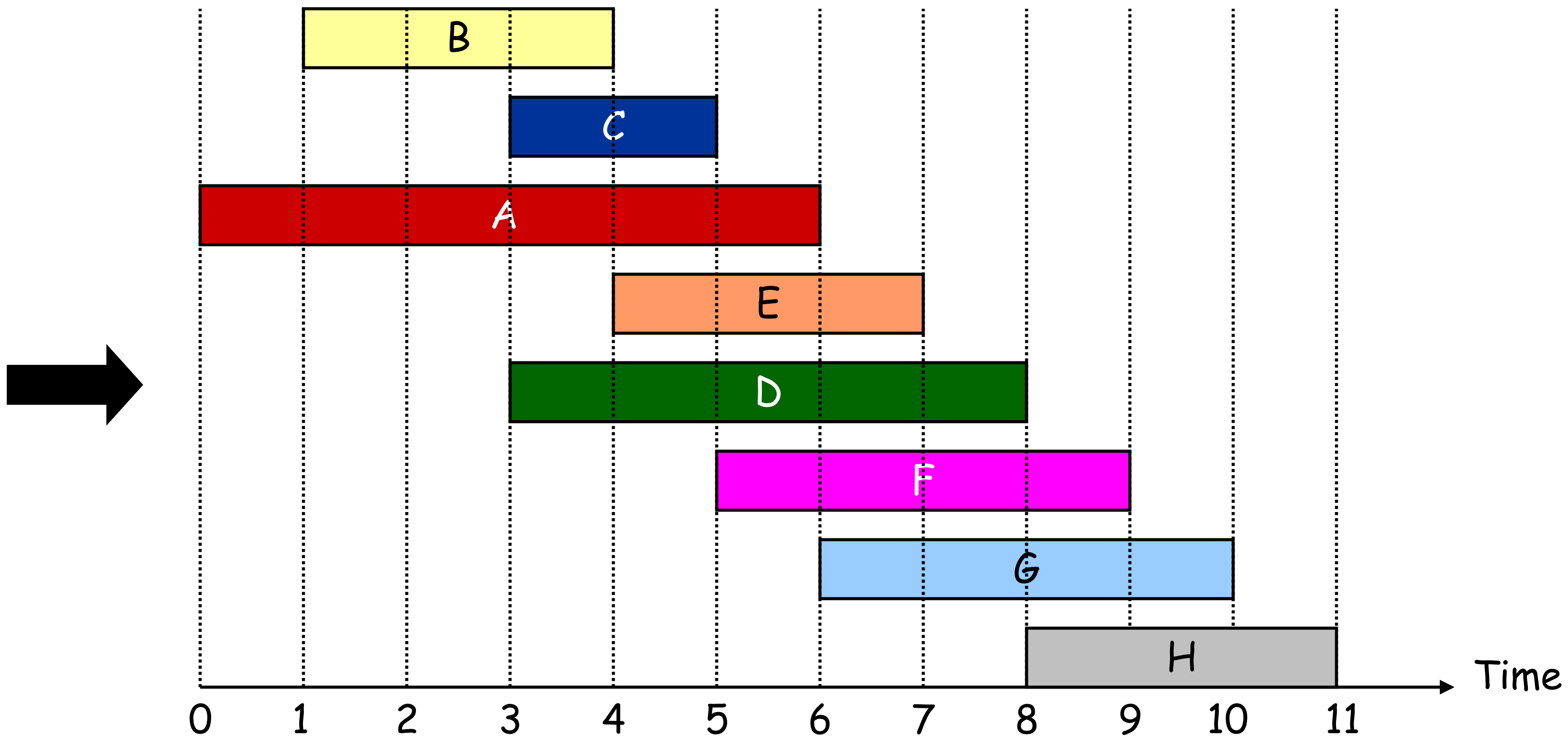


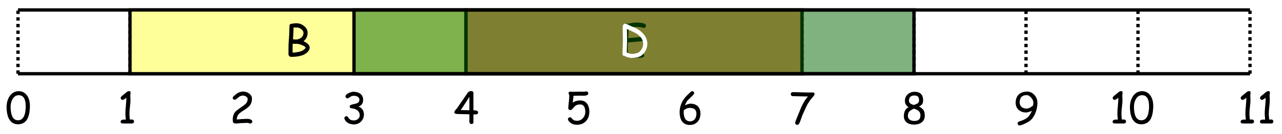
Interval Scheduling



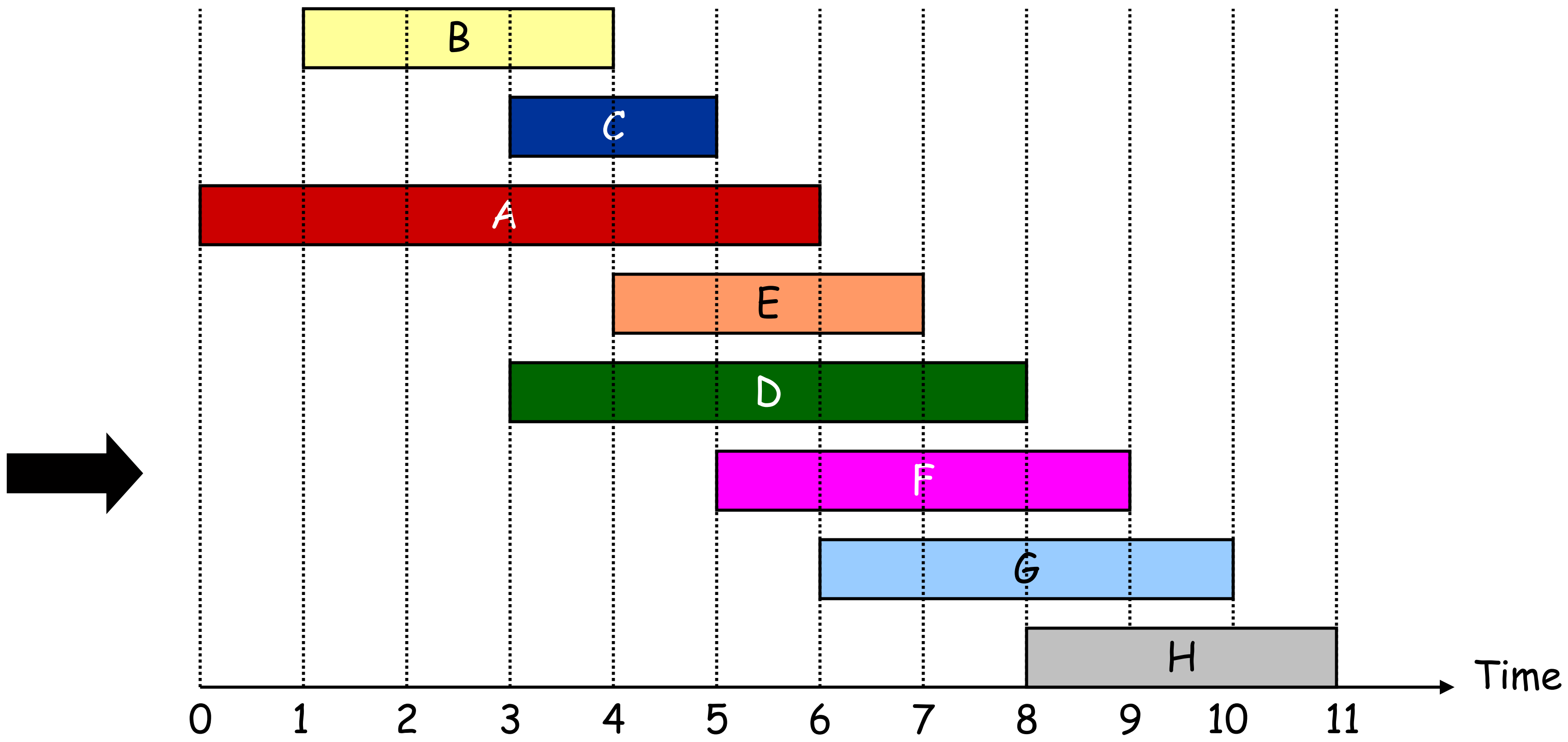


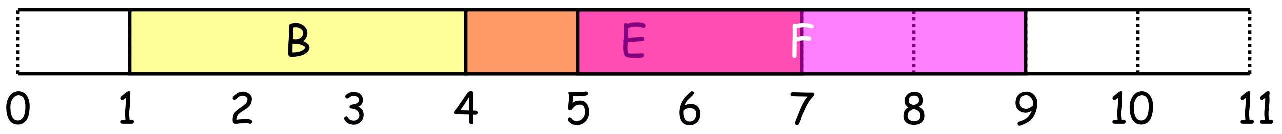
Interval Scheduling



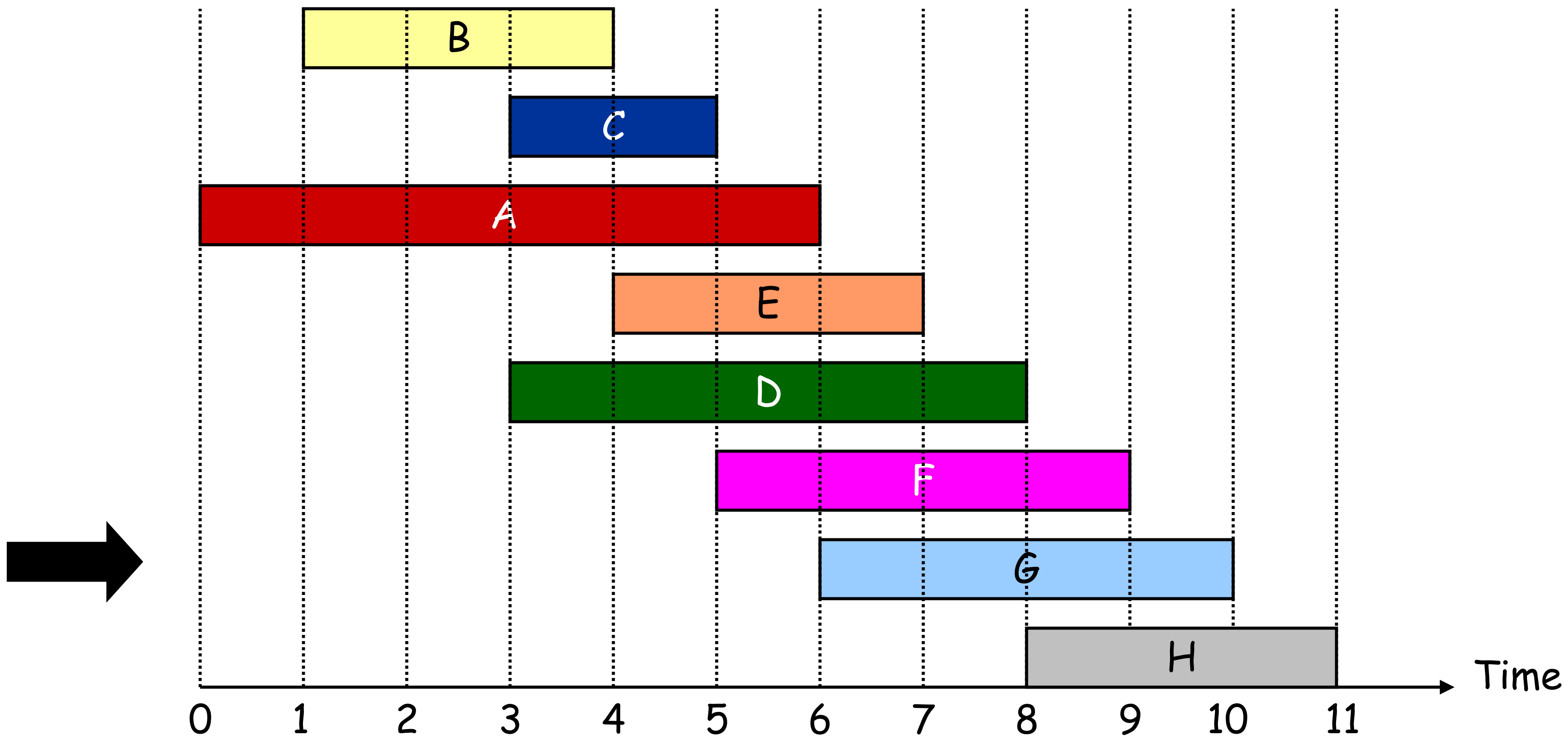


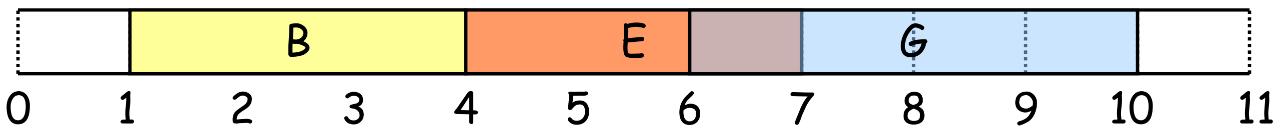
Interval Scheduling



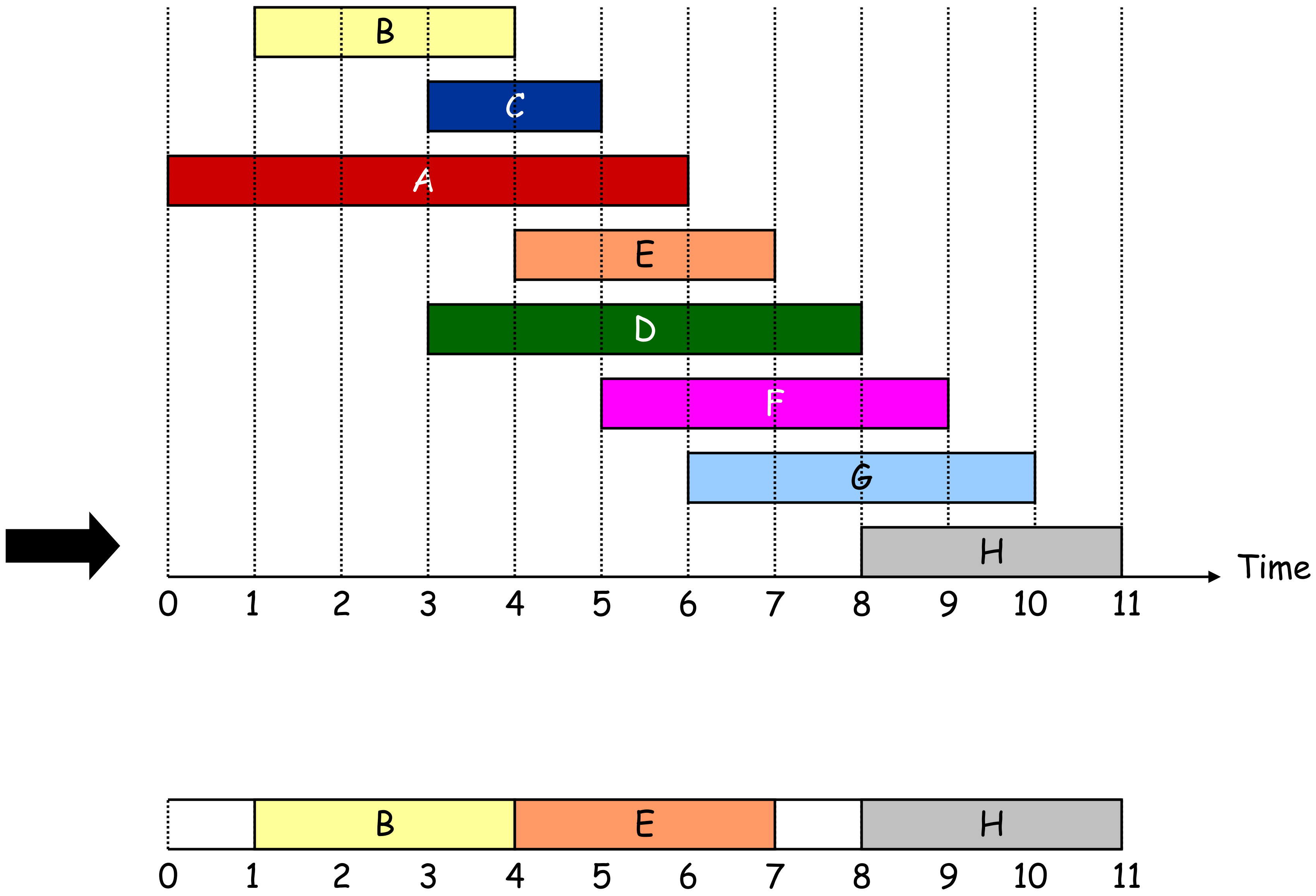


Interval Scheduling



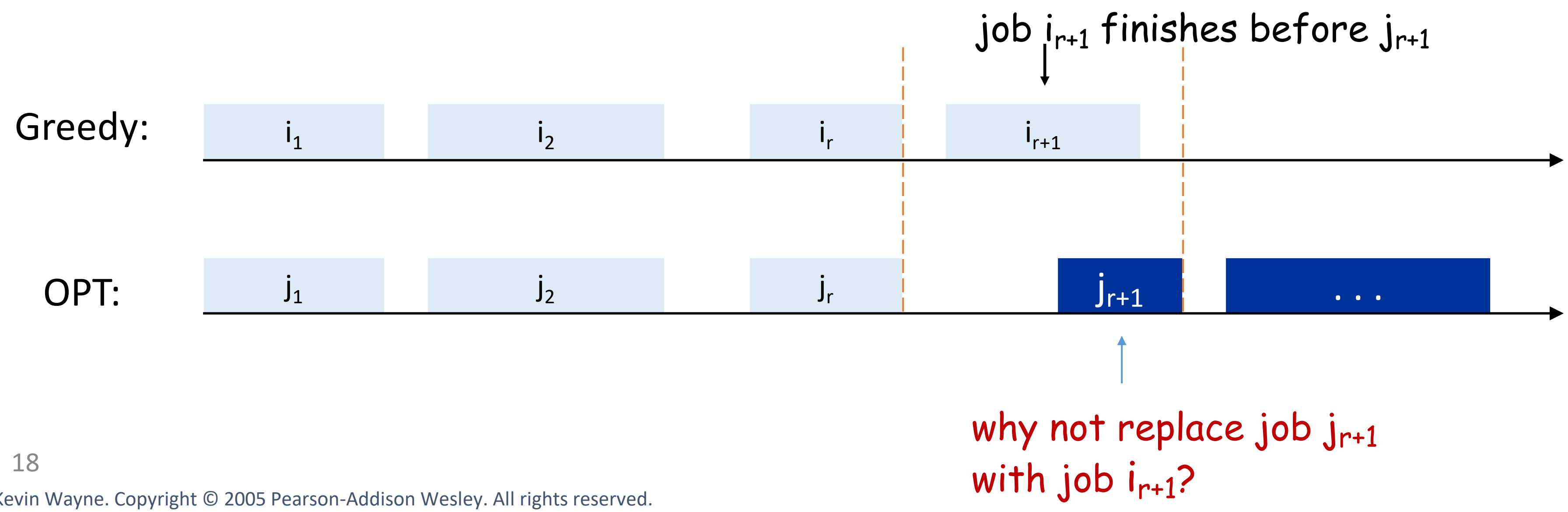


Interval Scheduling



Interval Scheduling: Analysis

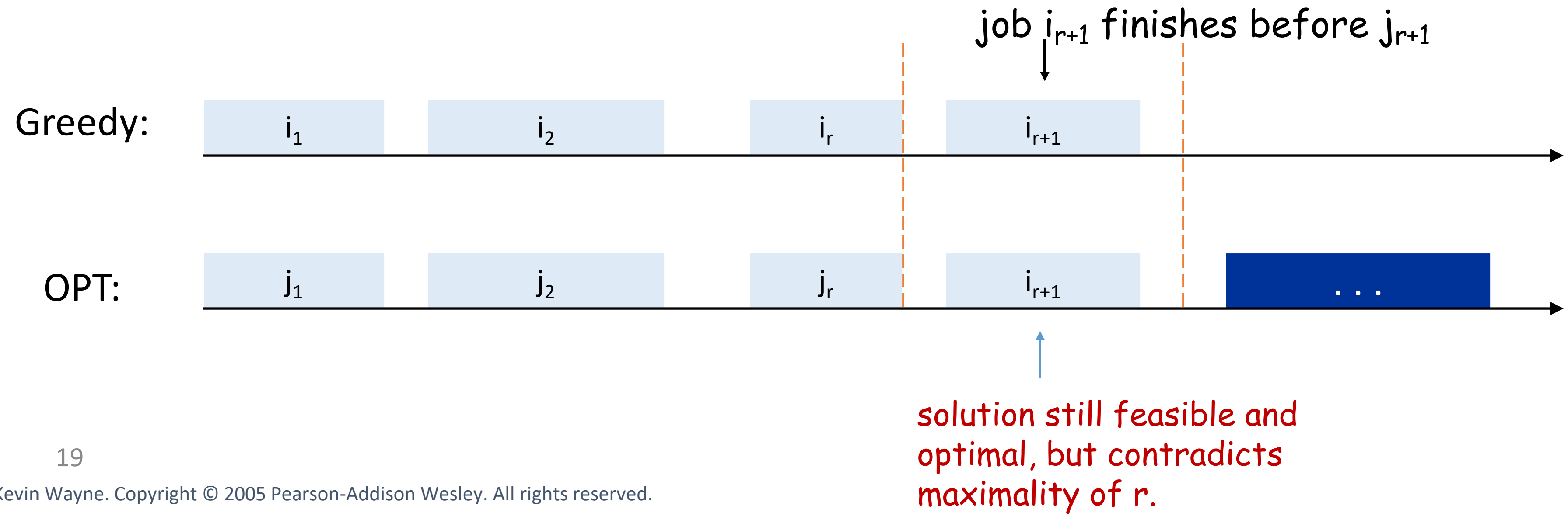
- Theorem. Greedy algorithm is optimal.
- Pf. (by contradiction)
  - Assume greedy is not optimal, and let's see what happens.
  - Let $i_1, i_2, \ldots i_k$ denote set of jobs selected by greedy.
  - Let $j_1, j_2, \ldots j_m$ denote set of jobs in the optimal solution with $i_1 = j_1, i_2 = j_2, \ldots,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



Slides by Kevin Wayne. Copyright © 2005 Pearson-Addison Wesley. All rights reserved.

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.



Slides by Kevin Wayne. Copyright © 2005 Pearson-Addison Wesley. All rights reserved.

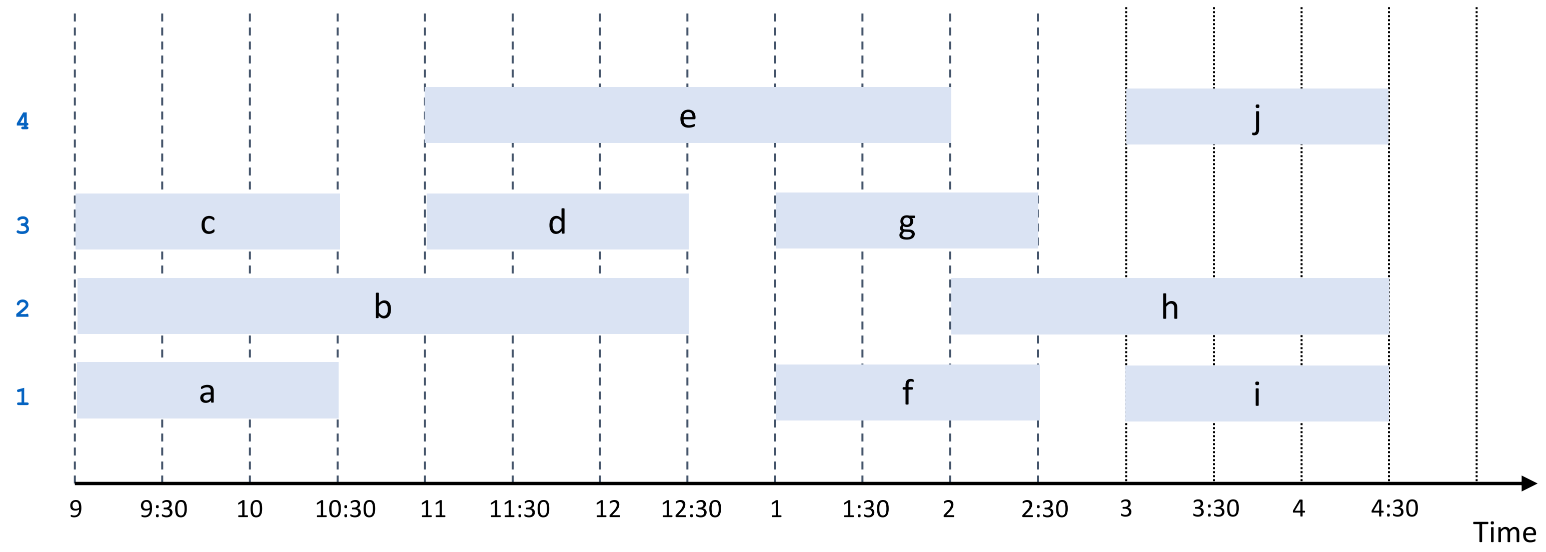
2. Interval Partitioning

Slides by Kevin Wayne. Copyright © 2005 Pearson-Addison Wesley. All rights reserved.

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.

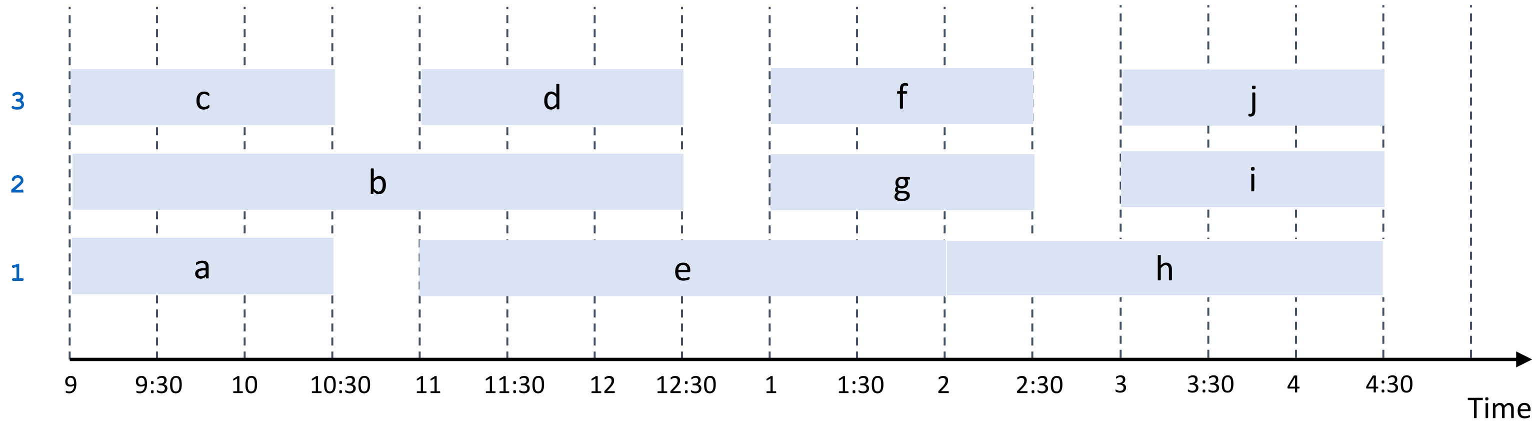


Slides by Kevin Wayne. Copyright © 2005 Pearson-Addison Wesley. All rights reserved.

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.



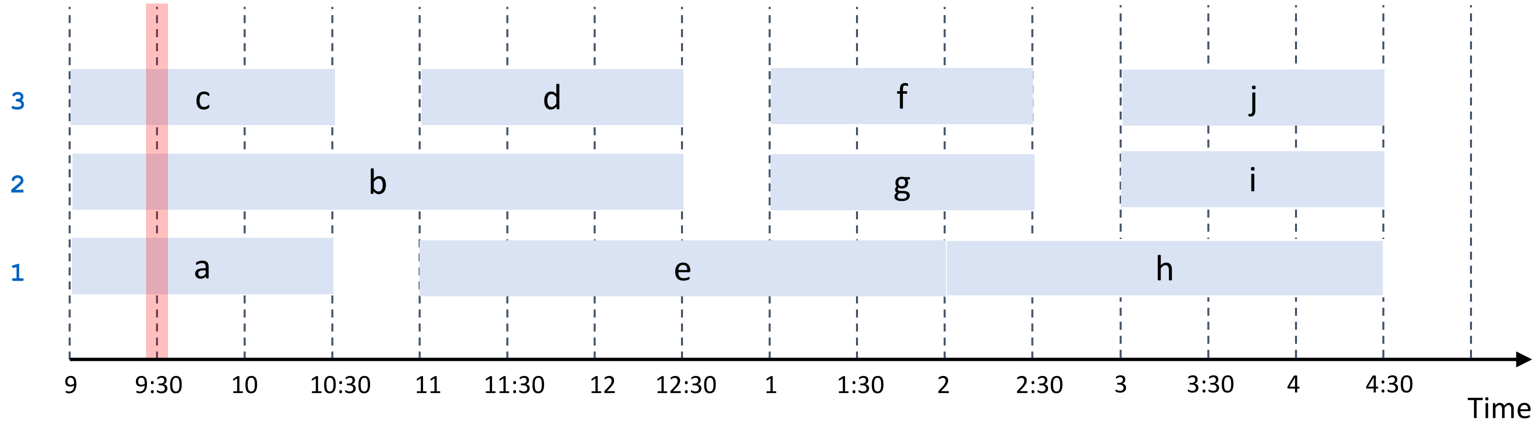
Slides by Kevin Wayne. Copyright © 2005 Pearson-Addison Wesley. All rights reserved.

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 $\geq$ 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?



Slides by Kevin Wayne. Copyright © 2005 Pearson-Addison Wesley. All rights reserved.

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 s1 ≤ s2 ≤ ... ≤ sn.
d ← 0      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 ← 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.

Slides by Kevin Wayne. Copyright © 2005 Pearson-Addison Wesley. All rights reserved.

Interval Partitioning: Greedy Analysis

- 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.
  - These d jobs each end after s_j.
  - 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. ▪

Slides by Kevin Wayne. Copyright © 2005 Pearson-Addison Wesley. All rights reserved.

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

3. Scheduling to Minimize Lateness

PEARSON Addison Wesley

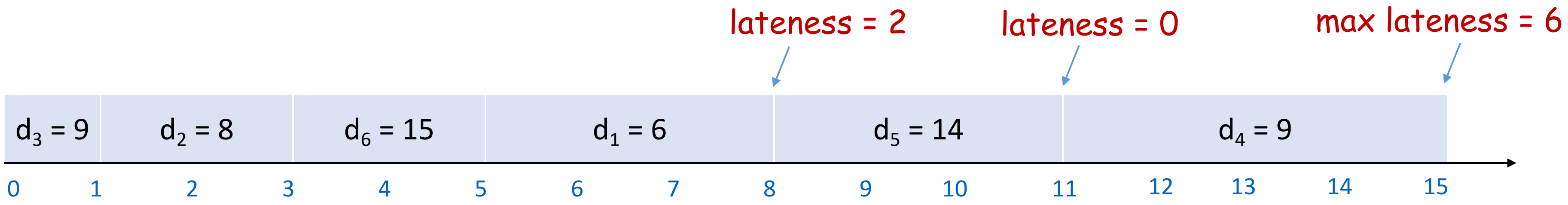
Slides by Kevin Wayne. Copyright © 2005 Pearson-Addison Wesley. All rights reserved.

Scheduling to Minimizing Lateness

- Minimizing lateness problem.
  - Single resource processes one job at a time.
  - Job j requires t_j units of processing time and is due at time d_j.
  - If j starts at time s_j, it finishes at time $f_j = s_j + t_j$.
  - Lateness: $\ell_j = \max \{ 0, \ f_j - d_j \}$.
  - Goal: schedule all jobs to minimize maximum lateness $L = \max \ell_j$.

- Ex:

	1	2	3	4	5	6
t_j	3	2	1	4	3	2
d_j	6	8	9	9	14	15



Slides by Kevin Wayne. Copyright © 2005 Pearson-Addison Wesley. All rights reserved.

Minimizing Lateness: Greedy Algorithms

- Greedy template. Consider jobs in some order.

  - [Shortest processing time first] Consider jobs in ascending order of processing time t_j.

  - [Earliest deadline first] Consider jobs in ascending order of deadline d_j.

  - [Smallest slack] Consider jobs in ascending order of slack $d_j - t_j$.

Slides by Kevin Wayne. Copyright © 2005 Pearson-Addison Wesley. All rights reserved.

Minimizing Lateness: Greedy Algorithms

- Greedy template. Consider jobs in some order.

  - [Shortest processing time first] Consider jobs in ascending order of processing time t_j.

	1	2
t_j	1	10
d_j	100	10

counterexample

  - [Smallest slack] Consider jobs in ascending order of slack $d_j - t_j$.

	1	2
t_j	1	10
d_j	2	10

counterexample

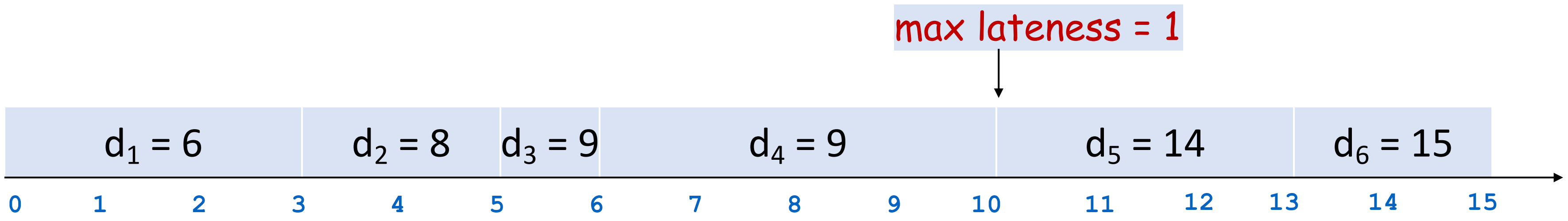
Slides by Kevin Wayne. Copyright © 2005 Pearson-Addison Wesley. All rights reserved.

Minimizing Lateness: Greedy Algorithm

- Greedy algorithm. Earliest deadline first.

Sort n jobs by deadline so that $d_1 \le d_2 \le \ldots \le d_n$

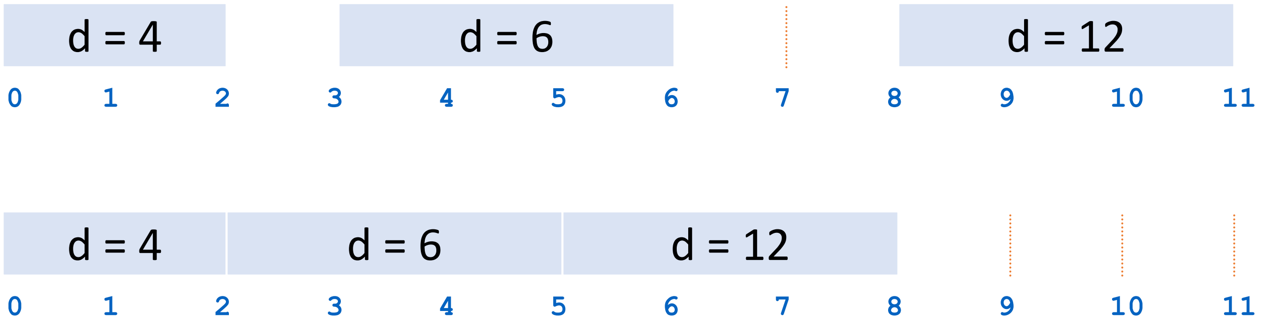
$t \leftarrow 0$
for $j = 1$ to n
  Assign job j to interval $[t,\ t + t_j]$
  $s_j \leftarrow t,\ f_j \leftarrow t + t_j$
  $t \leftarrow t + t_j$
output intervals $[s_j,\ f_j]$



Slides by Kevin Wayne. Copyright © 2005 Pearson-Addison Wesley. All rights reserved.

Minimizing Lateness: No Idle Time

- Observation. There exists an optimal schedule with no idle time.

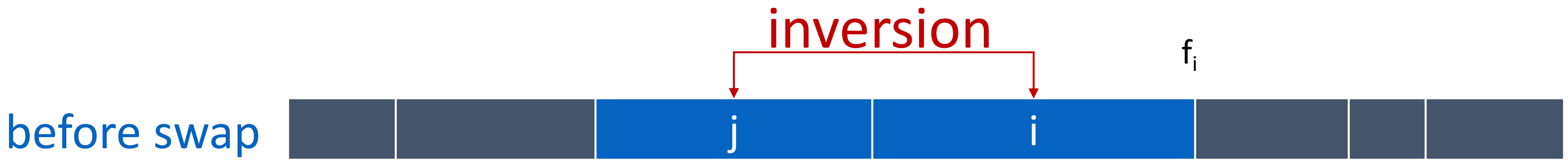


- Observation. The greedy schedule has no idle time.

Slides by Kevin Wayne. Copyright © 2005 Pearson-Addison Wesley. All rights reserved.

Minimizing Lateness: Inversions

- Def. Given a schedule S, an inversion is a pair of jobs i and j such that: $d_i < d_j$ but j scheduled before i.



[as before, we assume jobs are numbered so that $d_1 \le d_2 \le \ldots \le d_n$]

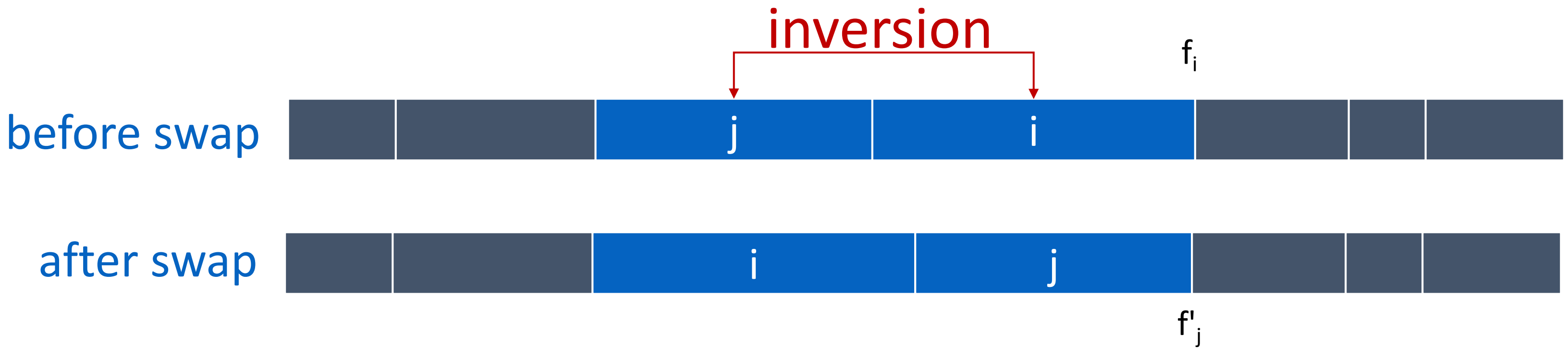
- Observation. Greedy schedule has no inversions.

- Observation. If a schedule (with no idle time) has an inversion, it has one with a pair of inverted jobs scheduled consecutively.

Slides by Kevin Wayne. Copyright © 2005 Pearson-Addison Wesley. All rights reserved.

Minimizing Lateness: Inversions

- Def. Given a schedule S, an inversion is a pair of jobs i and j such that: $d_i < d_j$ but j scheduled before i.



- Claim. Swapping two consecutive, inverted jobs reduces the number of inversions by one and does not increase the max lateness.

- Pf. Let ℓ be the lateness before the swap, and let ℓ' be it afterwards.
  - $\ell'_k = \ell_k$ for all $k \neq i, j$
  - $\ell'_i \leq \ell_i$
  - $\ell'_j = f'_j - d_j = f_i - d_j < f_i - d_i = \ell_i$

Slides by Kevin Wayne. Copyright © 2005 Pearson-Addison Wesley. All rights reserved.

Minimizing Lateness: Analysis of Greedy Algorithm

- Theorem. Greedy schedule S is optimal.
- Pf. Define S* to be an optimal schedule that has the fewest number of inversions, and let's see what happens.
  - Can assume S* has no idle time.
  - If S* has no inversions, then S = S*.
  - If S* has an inversion, let i-j be an adjacent inversion.
    - ✓ swapping i and j does not increase the maximum lateness and strictly decreases the number of inversions
    - ✓ this contradicts definition of S* ▪

Slides by Kevin Wayne. Copyright © 2005 Pearson-Addison Wesley. All rights reserved.

Greedy Analysis Strategies

- Greedy algorithm stays ahead. Show that after each step of the greedy algorithm, its solution is at least as good as any other algorithm's.

- Structural. Discover a simple "structural" bound asserting that every possible solution must have a certain value. Then show that your algorithm always achieves this bound.

- Exchange argument. Gradually transform any solution to the one found by the greedy algorithm without hurting its quality.

- Other greedy algorithms. Kruskal, Prim, Dijkstra, Huffman, …

Slides by Kevin Wayne. Copyright © 2005 Pearson-Addison Wesley. All rights reserved.