CSE 6140/ CX 4140

Computational Science and Engineering ALGORITHMS

Greedy Algorithms - 1

Instructor: Xiuwei Zhang

Assistant Professor

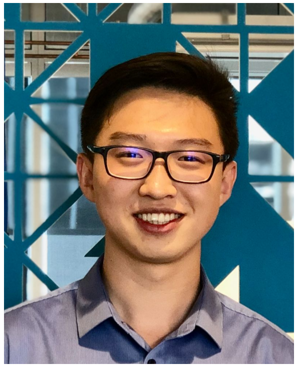
School of Computational Science and Engineering

Based on slides by Prof. Ümit V. Çatalyürek

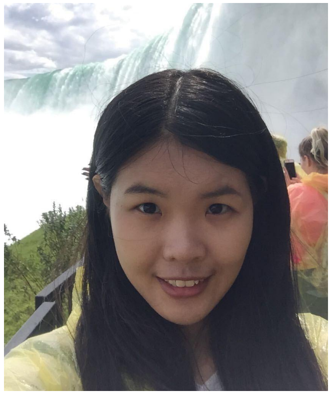
Course logistics

- TA team and updated office hours

Shahrokh Shahi (head TA)
shahi@gatech.edu



Ziqi Zhang
ziqi.zhang@gatech.edu



Yanjun Ding
yjding55@gatech.edu

Benjamin Cobb
bcobb33@gatech.edu

Jiancong Gao
jgao320@gatech.edu

Chenjun Tang
ctang90@gatech.edu

Updated office hour schedule

TA office hours start from the week of 8/24

	Monday	Tuesday	Wednesday	Thursday	Friday
8 AM					
9 AM					
10 AM		Ziqi 10 – 11am			
11 AM				Ben 11am – 12pm	
12 PM					
1 PM				Jiancong 1 – 2pm	Ziqi 1 – 2pm
2 PM	Instructor 2 – 3pm	Yanjun 2 – 3pm	Instructor 2 – 3pm		
3 PM	Shahrokh 3 – 4pm	Shahrokh 3 – 4pm		Yanjun 3 – 4pm	Ben 3 – 4pm
4 PM			Jiancong 4 – 5pm	Chenjun 4 – 5pm	Chenjun 4 – 5pm
5 PM					
6 PM					

You can find the meeting links in Canvas -> Calendar

Updates on exam policy

- Exams (Tests 1, 2, 3)
  - Open-book, proctored through Honorlock
  - Collaboration not allowed, use of internet not allowed during the exam; other equipments like ipad, smart phones are allowed only at the end of the exam to assist the scanning of hand-written answers
  - For submission, one can type in answers with locally installed software, or hand-write the answers and scan&upload. Overleaf is not allowed.
- Exam date update
  - Test 1 moved to Sep. 18 (previously Sep. 16)
- Homework collaboration
  - Can form study groups of up to 3 students
- Project collaboration
  - Can form study groups of up to 4 students

Greedy Algorithms

- **Greedy-choice property**: we can assemble a globally optimal solution by making locally optimal (greedy) choices

- i.e., we make the choice that looks best given the current partial solution

Problems covered with greedy algorithms

- Interval scheduling
- Scheduling to minimize lateness
- Interval partitioning
- Shortest path
- Minimum spanning tree
- Clustering

Problems covered with greedy algorithms

- Interval scheduling
- Scheduling to minimize lateness
- Interval partitioning
- Shortest path
- Minimum spanning tree
- Clustering

Most commonly used types of proofs:

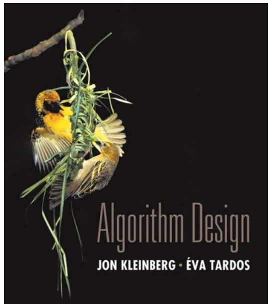
“Greedy stays ahead”

“exchange argument”

INTERVAL SCHEDULING [KT 4]

Adapted from Slides by
Kevin Wayne.

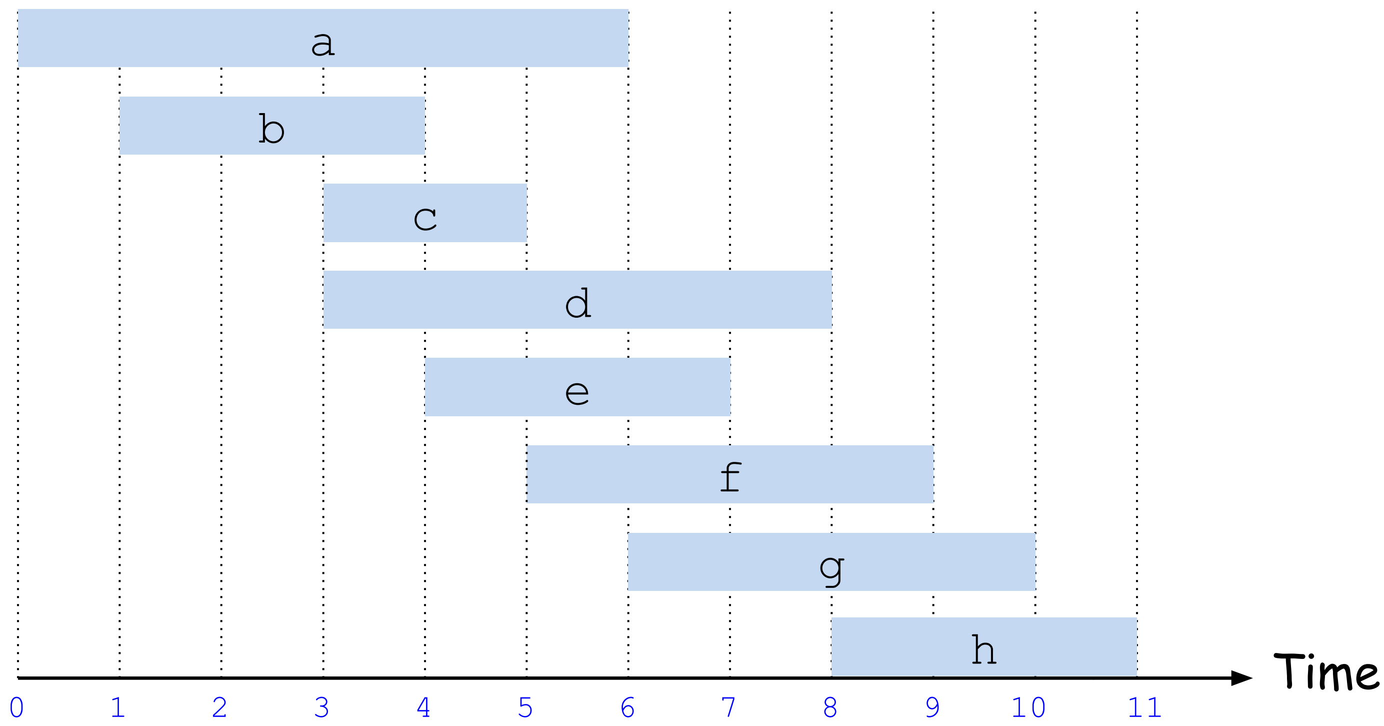
And Bistra Dilkina, Anne Benoit



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 natural order.
Take each job provided it's compatible with the ones already taken.

Interval Scheduling: Greedy Algorithms

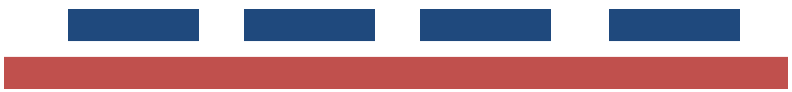
Greedy template. Consider jobs in some natural order.
Take each job provided it's compatible with the ones already taken.

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

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.

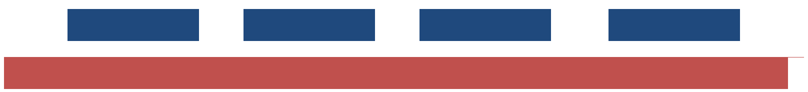


counterexample for earliest start time

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.



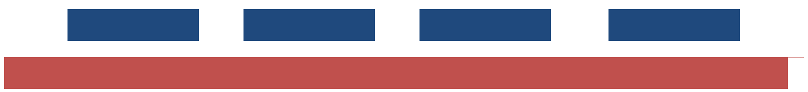
counterexample for earliest start time

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

Interval Scheduling: Greedy Algorithms

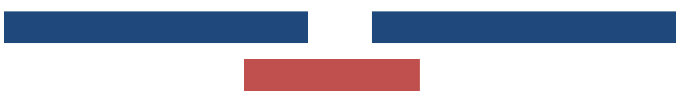
Greedy template. Consider jobs in some natural order.
Take each job provided it's compatible with the ones already taken.

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



counterexample for earliest start time

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

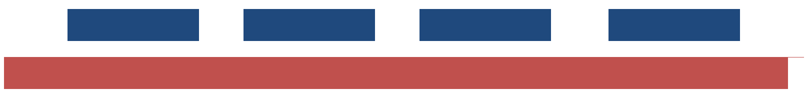


counterexample for shortest interval

Interval Scheduling: Greedy Algorithms

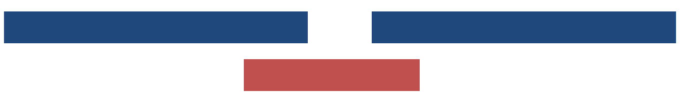
Greedy template. Consider jobs in some natural order.
Take each job provided it's compatible with the ones already taken.

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



counterexample for earliest start time

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



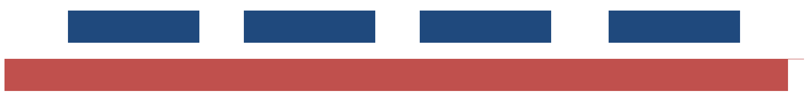
counterexample for shortest interval

- [Fewest conflicts] For each job j, count the number of conflicting jobs c_j. Schedule in ascending order of c_j.

Interval Scheduling: Greedy Algorithms

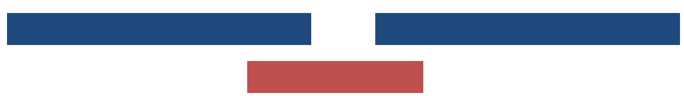
Greedy template. Consider jobs in some natural order.
Take each job provided it's compatible with the ones already taken.

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



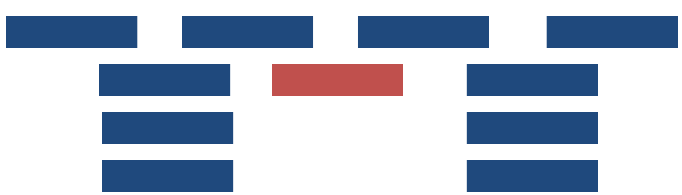
counterexample for earliest start time

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



counterexample for shortest interval

- [Fewest conflicts] For each job j, count the number of conflicting jobs c_j. Schedule in ascending order of c_j.



counterexample for fewest conflicts

Interval Scheduling: Greedy Algorithm

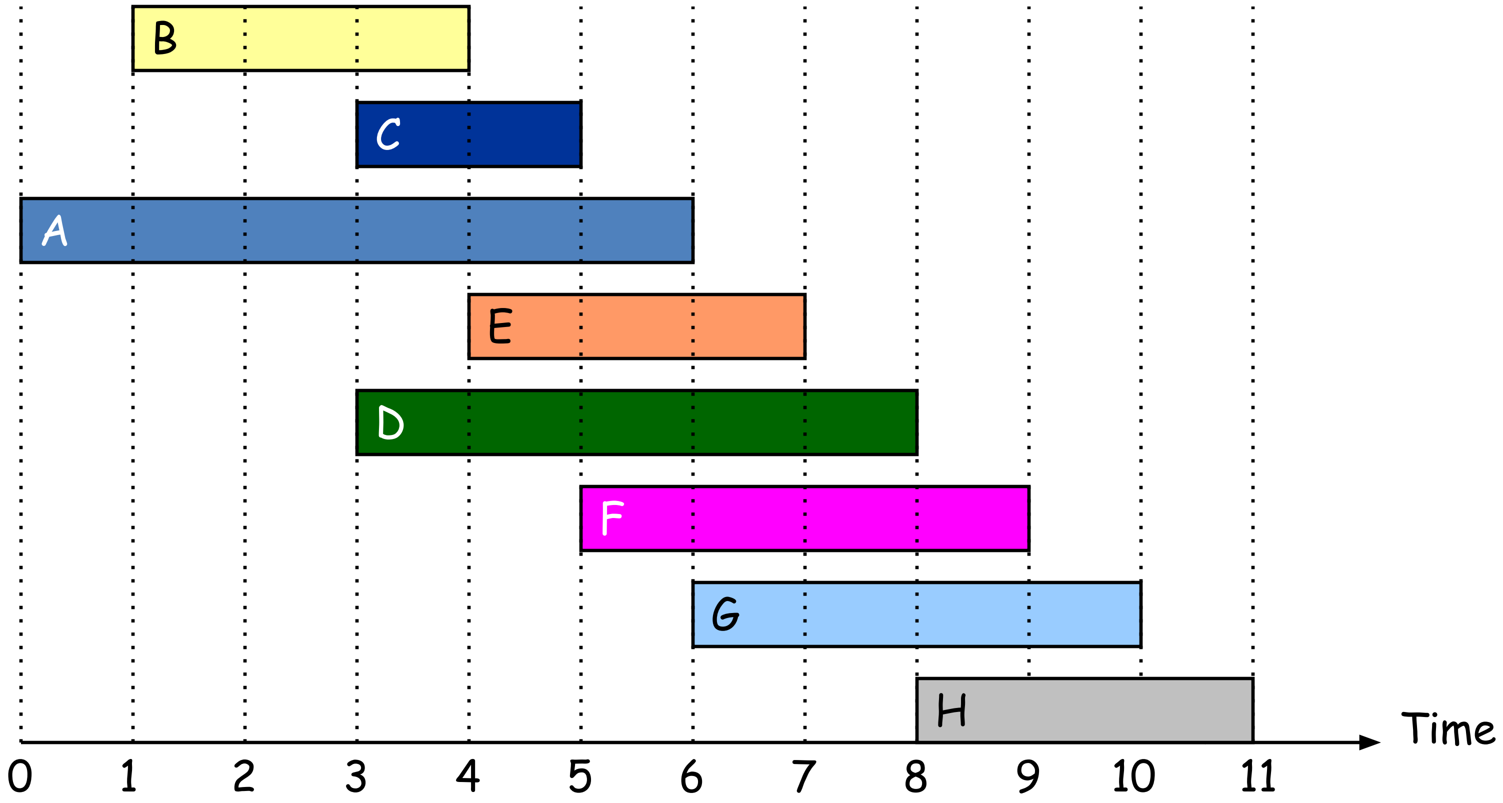
[Earliest finish time] Consider jobs in ascending order of f_j.

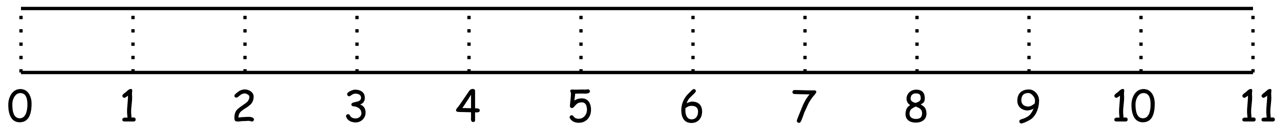
Greedy algorithm. Choose next job to add to solution as the one with earliest finish time that it is compatible with the ones already taken.

(natural order = finish time)

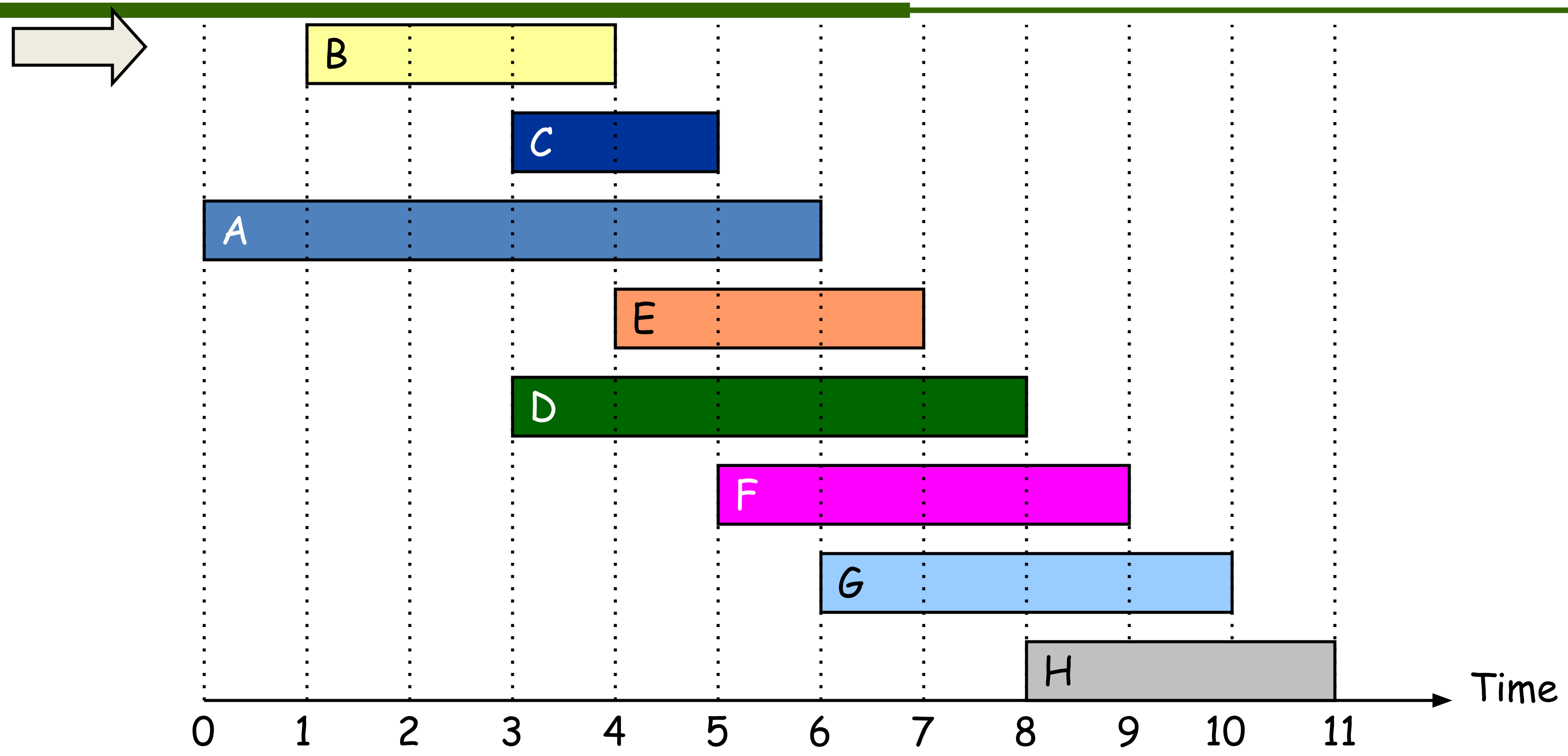
```
Sort jobs by finish times so that f1 ≤ f2 ≤ ... ≤ fn.
        set of jobs selected

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

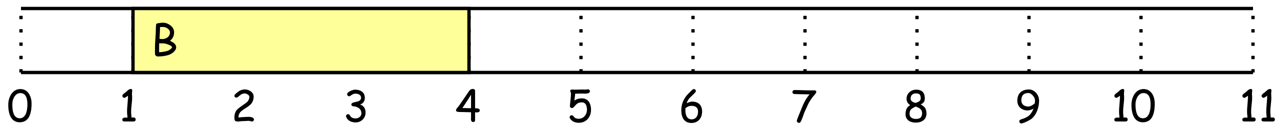
Interval Scheduling



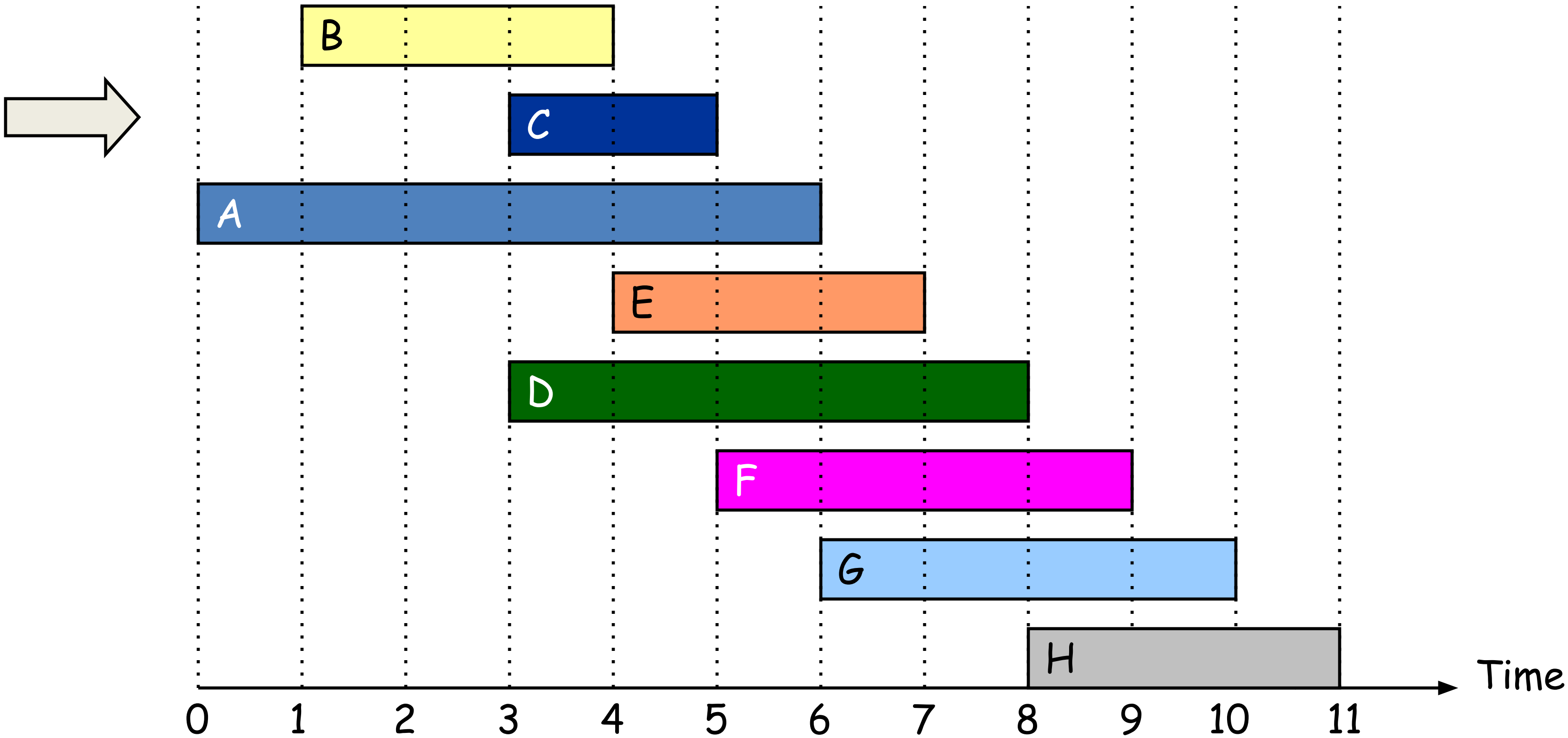


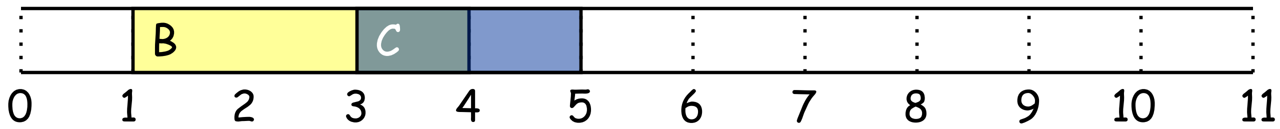
Interval Scheduling



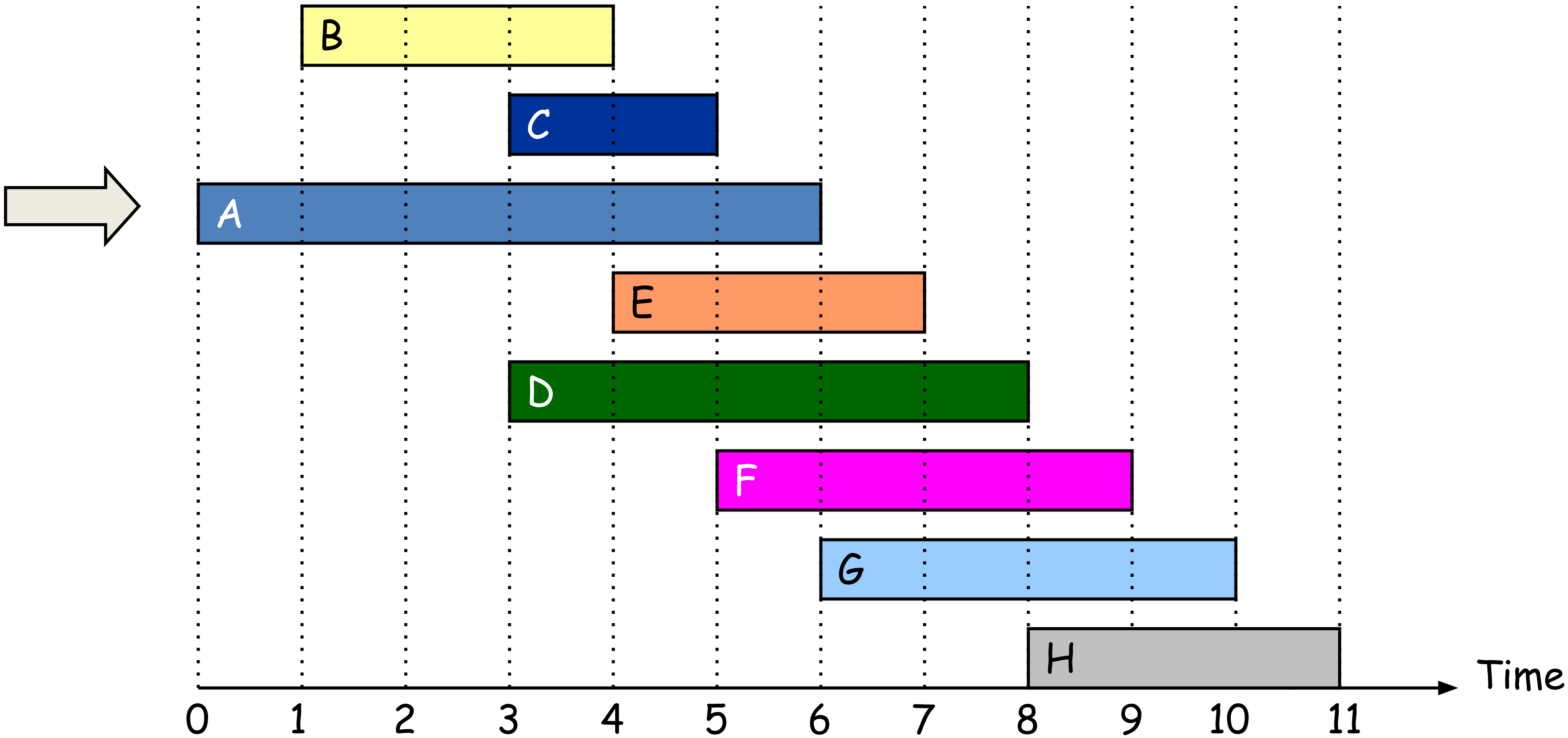


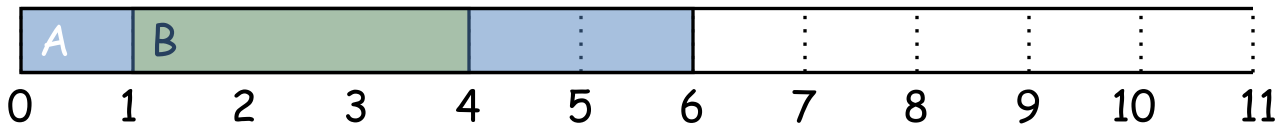
Interval Scheduling



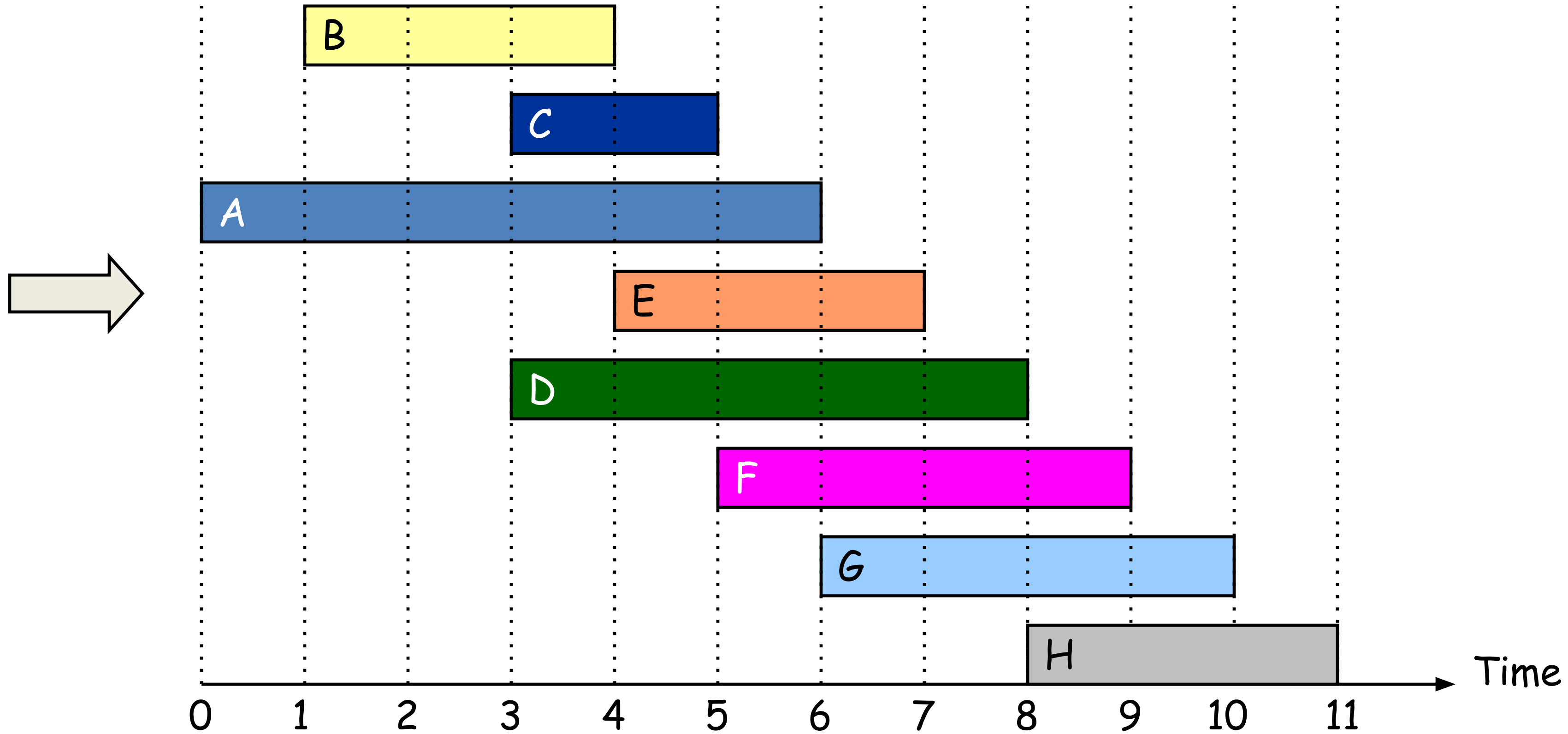


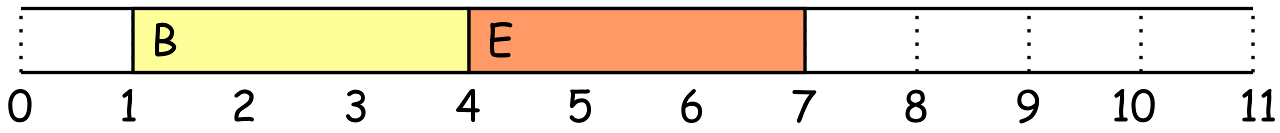
Interval Scheduling



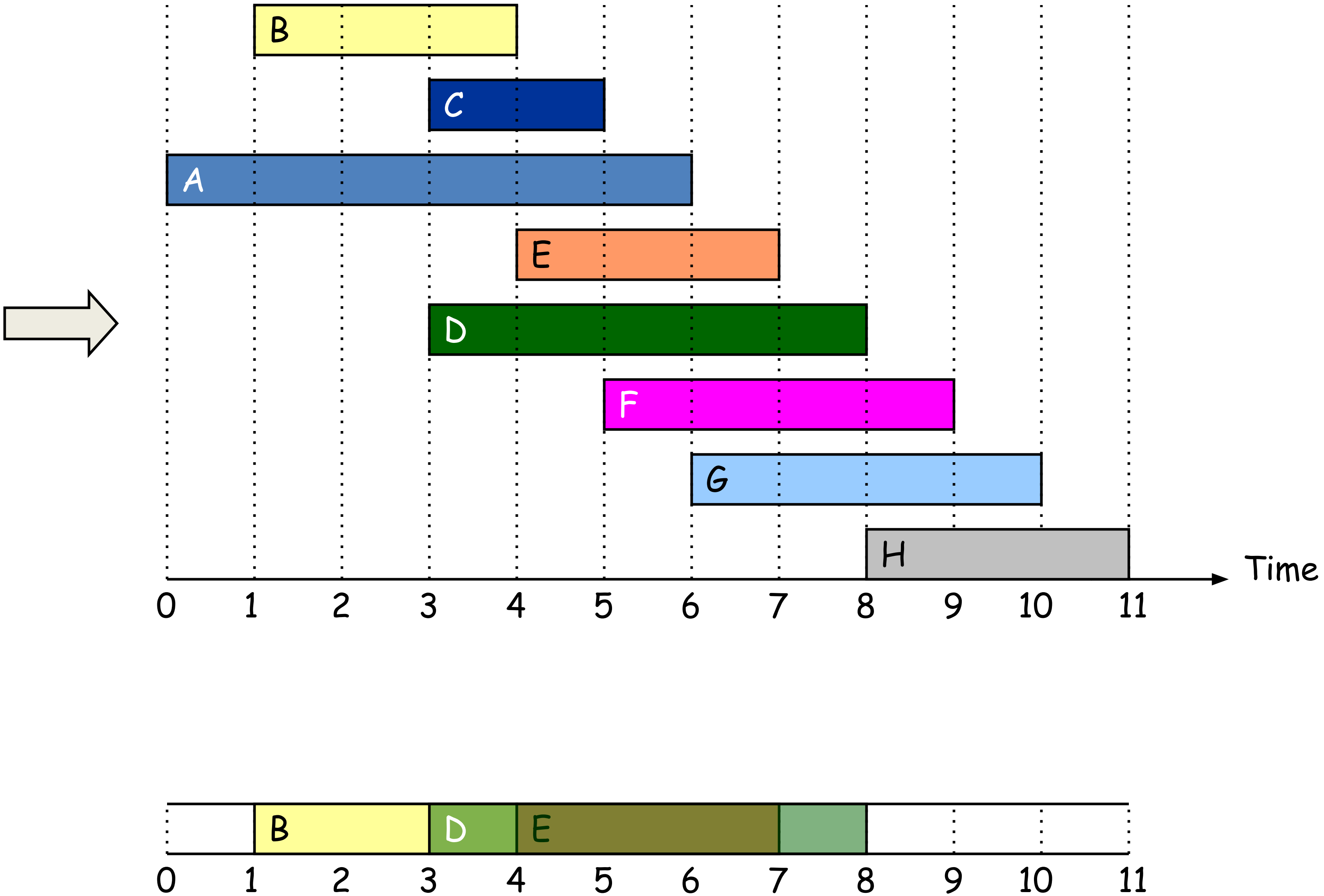


Interval Scheduling

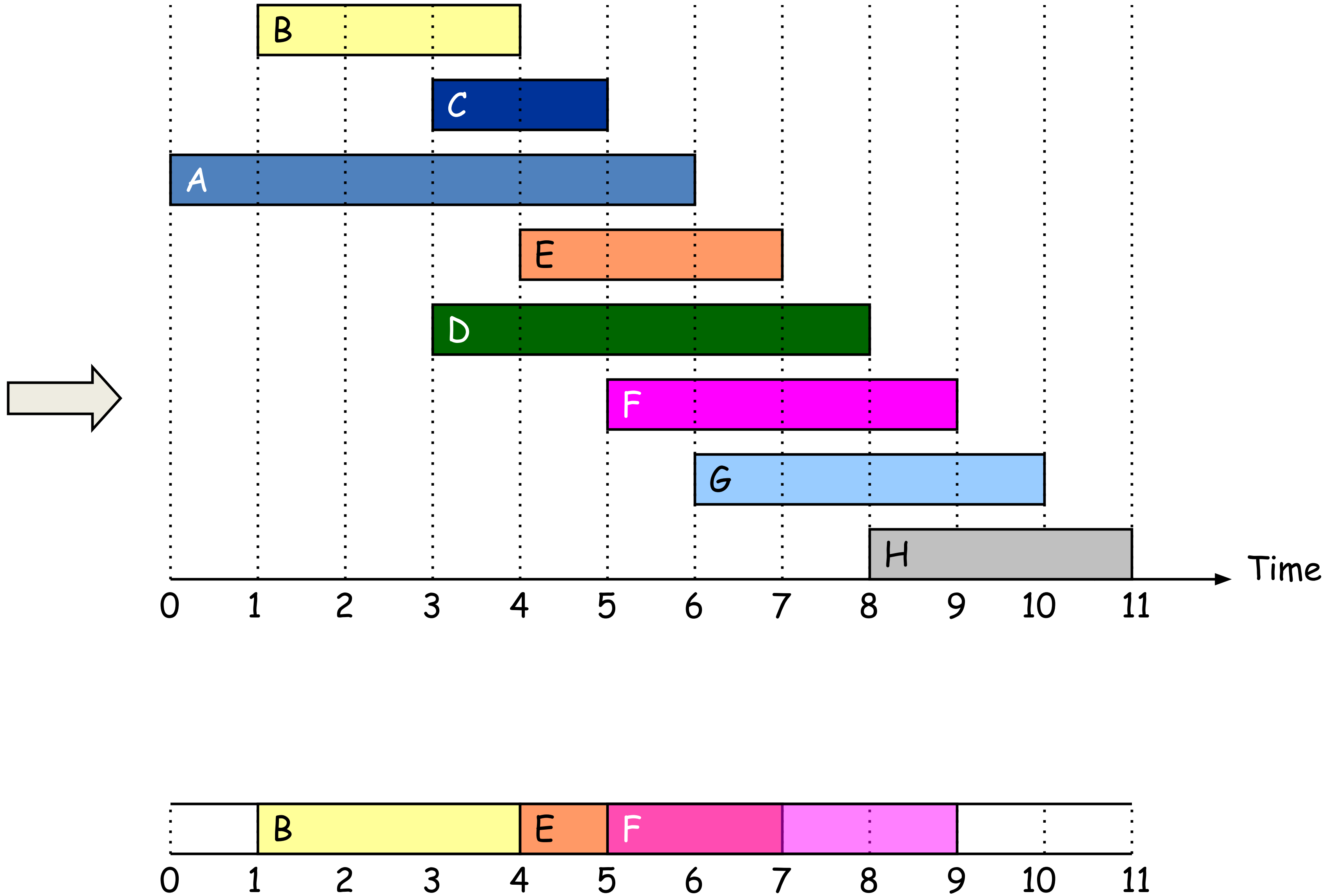




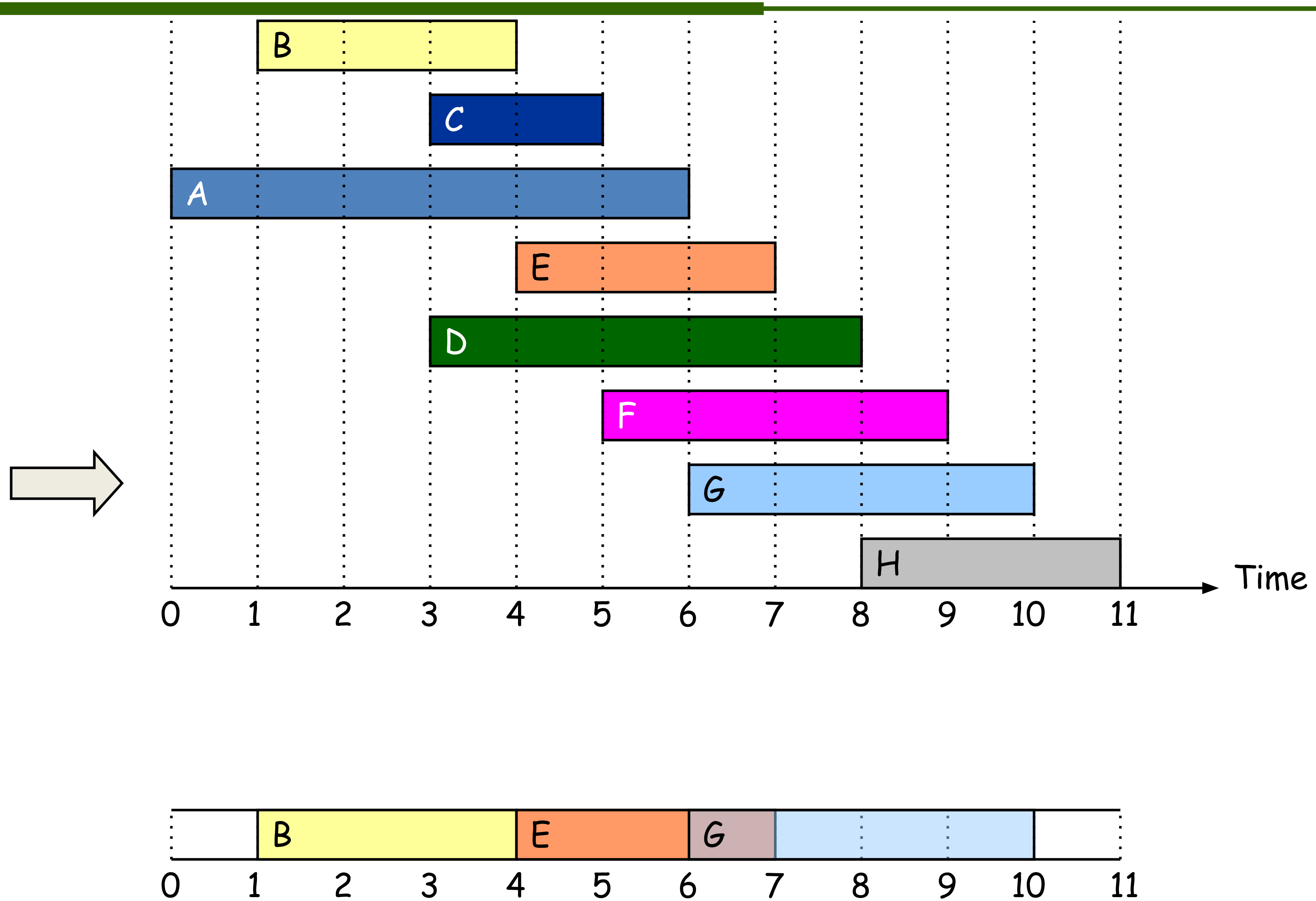
Interval Scheduling



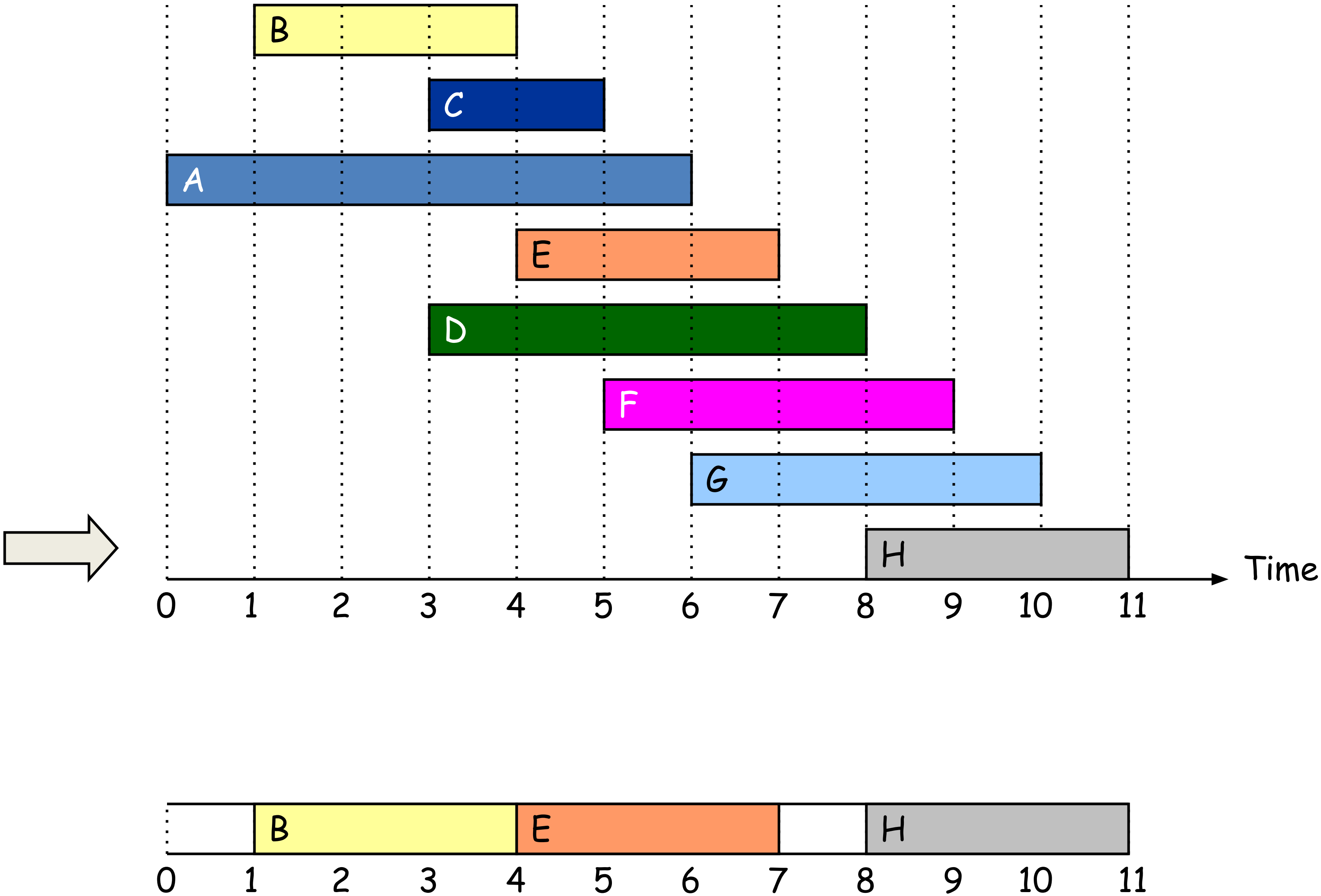
Interval Scheduling



Interval Scheduling



Interval Scheduling



Interval Scheduling: Greedy Algorithm

Greedy algorithm. Choose next job to add to solution as the one with earliest finish time that it is compatible with the ones already taken.

(natural order = finish time)

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

set of jobs selected

$A \leftarrow \varphi$
for j = 1 to n {
  if (job j compatible with A)
    $A \leftarrow A \cup \{j\}$
}
return A

Why is this optimal?

GREEDY IS OPTIMAL
(GREEDY STAYS AHEAD ARGUMENT)

Interval Scheduling: Greedy stays ahead argument

- Let A: $a_1, a_2, \dots\ a_k$ denote set of jobs selected by greedy.
- Let O: $o_1, o_2, \dots\ o_m$ denote set of jobs in the optimal solution.

What do we know about finish times of jobs in A?

$$f(a_1) < f(a_2) < \cdots < f(a_k)$$

We order the optimal solution in that way

$$f(o_1) < f(o_2) < \cdots < f(o_m)$$

Claim: For all indices $r \leq k$, $f(a_r) \leq f(o_r)$

Interval Scheduling: Greedy stays ahead argument

Claim: For all indices $r \leq k$, $f(a_r) \leq f(o_r)$

Interval Scheduling: Greedy stays ahead argument

Claim: For all indices $r \leq k$, $f(a_r) \leq f(o_r)$

Pf. (by induction)

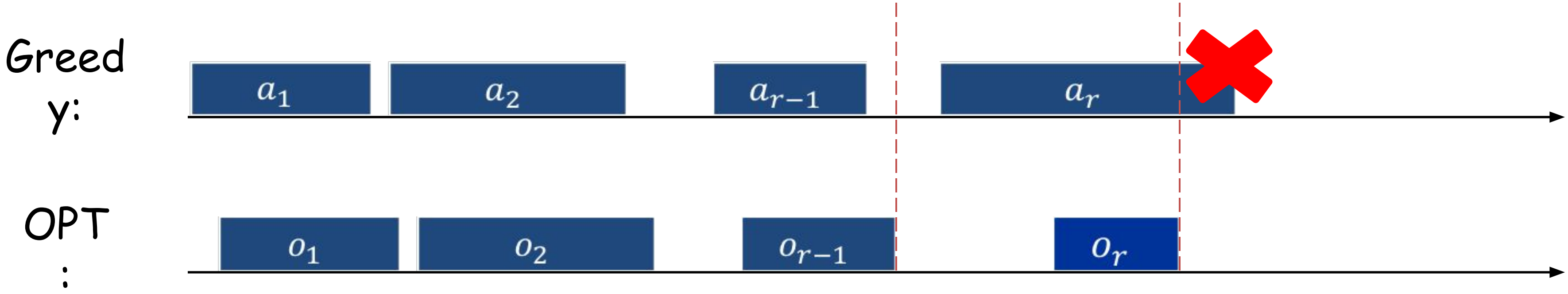
Base case: $r = 1$

$$f(a_1) \leq f(o_1)$$

True by greedy choice of earliest finish time.

Inductive hypothesis: Holds for $r-1$, i.e., $f(a_{r-1}) \leq f(o_{r-1})$

Inductive step:



$$f(o_{r-1}) \leq s(o_r) \leq f(o_r)$$

Interval Scheduling: Greedy stays ahead argument

$f(o_{r-1}) \leq s(o_r) \leq f(o_r)$ by feasibility of optimum solution

$f(a_{r-1}) \leq f(o_{r-1})$ by ind. Hypothesis

$f(a_{r-1}) \leq s(o_r)$

$\Rightarrow o_r$ is compatible with $a_1, a_2, \dots, a_{r-1}$

and was an option for greedy

a_r was the greedy choice among all compatible jobs at iteration r

$\Rightarrow f(a_r) \leq f(o_r)$

Interval Scheduling: Greedy stays ahead argument

$f(o_{r-1}) \leq s(o_r) \leq f(o_r)$ by feasibility of optimum solution

$f(a_{r-1}) \leq f(o_{r-1})$ by ind. Hypothesis

$f(a_{r-1}) \leq s(o_r)$

$\Rightarrow o_r$ is compatible with $a_1, a_2, \dots, a_{r-1}$

and was an option for greedy

a_r was the greedy choice among all compatible jobs at iteration r

$\Rightarrow f(a_r) \leq f(o_r)$

Are we done?

What if optimal solution has more jobs, i.e., m>k

Interval Scheduling: Greedy stays ahead argument

Theorem. Greedy algorithm is optimal $(k = m)$

A: $a_1, a_2, \ldots\ a_k$
O: $o_1, o_2, \ldots\ o_m$

Assume for the sake of contradiction $k < m$

k: $f(a_k) \le f(o_k)$

Optimal solution must have o_{k+1} $(k < m)$

Optimal solution O is feasible

$\Rightarrow f(o_k) \le s(o_{k+1}) \le f(o_{k+1})$

$\Rightarrow o_{k+1}$ was still an option for greedy because $f(a_k) \le s(o_{k+1})$ after iteration k

$\Rightarrow$ Contradiction greedy stopping at iteration k

$\Rightarrow k = m$ ■

Interval Scheduling: Greedy Algorithm

Greedy algorithm. Choose next job to add to solution as the one with earliest finish time that it is compatible with the ones already taken.

(natural order = finish time)

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

set of jobs selected

$A \leftarrow \varphi$
for $j = 1$ to n {
  if (job j compatible with A)
    $A \leftarrow A \cup \{j\}$
}
return A

Interval Scheduling: Greedy Algorithm

Greedy algorithm. Choose next job to add to solution as the one with earliest finish time that it is compatible with the ones already taken.

(natural order = finish time)

$O(n \log n)$

Naïve $O(n^2)$

Naïve $O(n)$

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

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

Running time: $O(n^2)$.

Can we make this faster?

Interval Scheduling: Greedy Algorithm

Greedy algorithm. Choose next job to add to solution as the one with earliest finish time that it is compatible with the ones already taken.

(natural order = finish time)

$O(n \log n)$

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

set of jobs selected

$O(n)$

```
A ← φ
f_j* = 0
for j = 1 to n {
   if (job j compatible with A : s_j ≥ f_j*) {
      A ← A ∪ {j}
      f_j* = f_j
   }
}
return A
```

Running time: $O(n \log n)$

Interval scheduling: quiz 1

- If there are multiple feasible jobs with the same finish time, how to we choose which one to add?

Interval scheduling: quiz 2

- Suppose that each job also has a positive weight and the goal is to find a maximum weight subset of mutually compatible intervals. Is the earliest-finish-time-first algorithm still optimal?
  a) Yes, because greedy algorithms are always optimal.
  b) Yes, because the same proof of correctness is valid.
  c) No, because the same proof of correctness is no longer valid.
  d) No, because you could assign a huge weight to a job that overlaps the job with the earliest finish time.

Greedy algorithms I: quiz

- Suppose that each job also has a positive weight and the goal is to find a maximum weight subset of mutually compatible intervals. Is the earliest-finish-time-first algorithm still optimal?

  a) Yes, because greedy algorithms are always optimal.

  b) Yes, because the same proof of correctness is valid.

  c) No, because the same proof of correctness is no longer valid.

  d) No, because you could assign a huge weight to a job that overlaps the job with the earliest finish time.

counterexample for earliest finish time