CSE 431/531: Algorithm Analysis and Design (Fall 2024)
Greedy Algorithms

Lecturer: Kelin Luo

Department of Computer Science and Engineering
University at Buffalo

Announcements: Quiz 4

- Posted on Ublearns
- Should take $<$ 30 minutes, 2 attempts
- Due Tue 17 Sep @ 11:59PM

Outline

1. Toy Example: Box Packing

2. Interval Scheduling
   - Interval Partitioning

Generic Greedy Algorithm

```
while the instance is non-trivial do
    make the choice using the greedy strategy
    reduce the instance
```

Lemma Generic algorithm is correct if and only if the greedy strategy is safe.

Generic Greedy Algorithm

1: **while** the instance is non-trivial **do**
2: make the choice using the greedy strategy
3: reduce the instance

Lemma Generic algorithm is correct if and only if the greedy strategy is safe.

- Greedy strategy is safe: we will not miss the optimum solution

Generic Greedy Algorithm

1: **while** the instance is non-trivial **do**
2:     make the choice using the greedy strategy
3:     reduce the instance

Lemma Generic algorithm is correct if and only if the greedy strategy is safe.

- Greedy strategy is safe: we will not miss the optimum solution
- Greedy stretegy is not safe: we will miss the optimum solution for some instance, since the choices we made are irrevocable.

Greedy Algorithm

- Build up the solutions in steps
- At each step, make an irrevocable decision using a "reasonable" strategy

Greedy Algorithm

- Build up the solutions in steps
- At each step, make an irrevocable decision using a "reasonable" strategy

Analysis of Greedy Algorithm

- Safety: Prove that the reasonable strategy is "safe"
- Self-reduce: Show that the remaining task after applying the strategy is to solve a (many) smaller instance(s) of the same problem

Greedy Algorithm

- Build up the solutions in steps
- At each step, make an irrevocable decision using a "reasonable" strategy

Analysis of Greedy Algorithm

- Safety: Prove that the reasonable strategy is "safe"
- Self-reduce: Show that the remaining task after applying the strategy is to solve a (many) smaller instance(s) of the same problem

Def. A strategy is "safe" if there is always an optimum solution that is "consistent" with the decision made according to the strategy.

Exchange argument: Proof of Safety of a Strategy

- let S be an arbitrary optimum solution.
- if S is consistent with the greedy choice, done.
- otherwise, show that it can be modified to another optimum solution S' that is consistent with the choice.

Exchange argument: Proof of Safety of a Strategy

- let S be an arbitrary optimum solution.
- if S is consistent with the greedy choice, done.
- otherwise, show that it can be modified to another optimum solution S' that is consistent with the choice.

- The procedure is not a part of the algorithm.

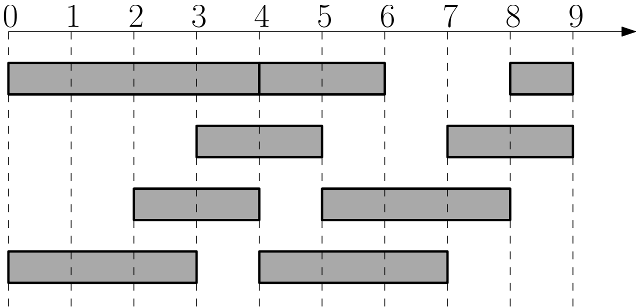
Outline

Interval Scheduling

Input: n jobs, job i with start time s_i and finish time f_i

i and j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ are disjoint

Output: A maximum-size subset of mutually compatible jobs

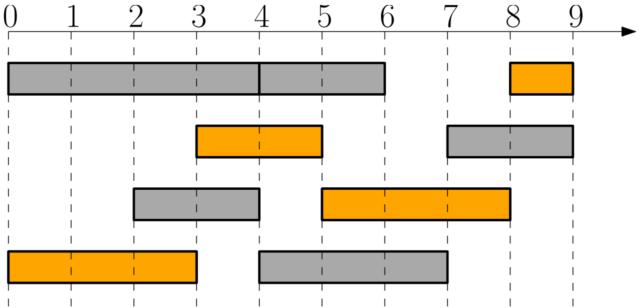


Interval Scheduling

Input: n jobs, job i with start time s_i and finish time f_i

i and j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ are disjoint

Output: A maximum-size subset of mutually compatible jobs



Greedy Algorithm for Interval Scheduling

- Which of the following strategies are safe?

Greedy Algorithm for Interval Scheduling

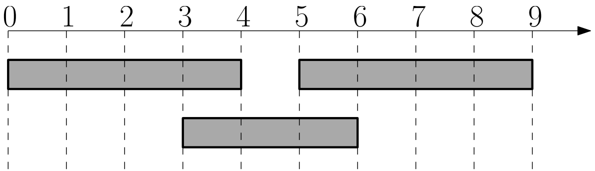
- Which of the following strategies are safe?
- Schedule the job with the smallest size?

Greedy Algorithm for Interval Scheduling

- Which of the following strategies are safe?
- Schedule the job with the smallest size? No!

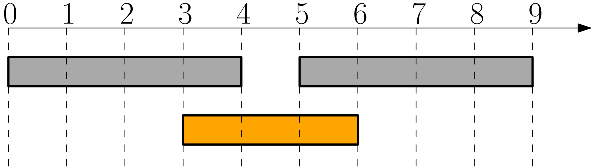
Greedy Algorithm for Interval Scheduling

- Which of the following strategies are safe?
- Schedule the job with the smallest size? No!



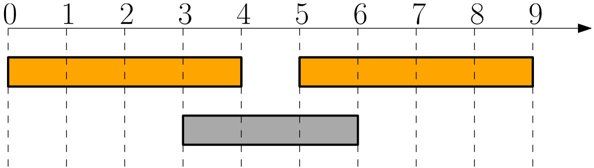
Greedy Algorithm for Interval Scheduling

- Which of the following strategies are safe?
- Schedule the job with the smallest size? No!



Greedy Algorithm for Interval Scheduling

- Which of the following strategies are safe?
- Schedule the job with the smallest size? No!



Greedy Algorithm for Interval Scheduling

- Which of the following strategies are safe?
- Schedule the job with the smallest size? No!

Greedy Algorithm for Interval Scheduling

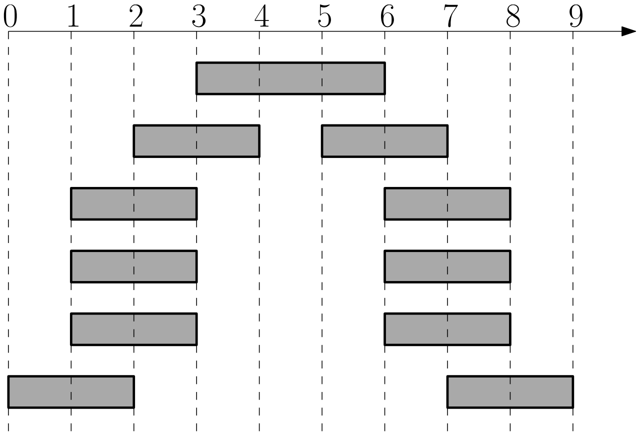
- Which of the following strategies are safe?
- Schedule the job with the smallest size? No!
- Schedule the job conflicting with smallest number of other jobs?

Greedy Algorithm for Interval Scheduling

- Which of the following strategies are safe?
- Schedule the job with the smallest size? No!
- Schedule the job conflicting with smallest number of other jobs? No!

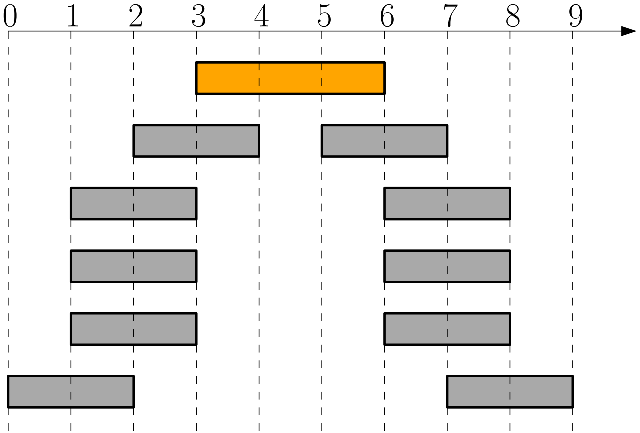
Greedy Algorithm for Interval Scheduling

- Which of the following strategies are safe?
- Schedule the job with the smallest size? No!
- Schedule the job conflicting with smallest number of other jobs? No!



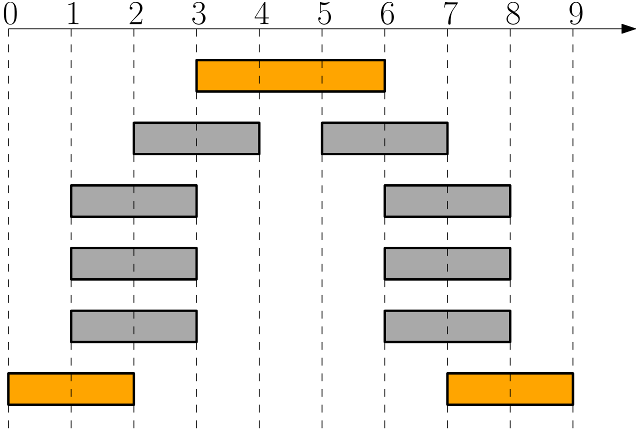
Greedy Algorithm for Interval Scheduling

- Which of the following strategies are safe?
- Schedule the job with the smallest size? No!
- Schedule the job conflicting with smallest number of other jobs? No!



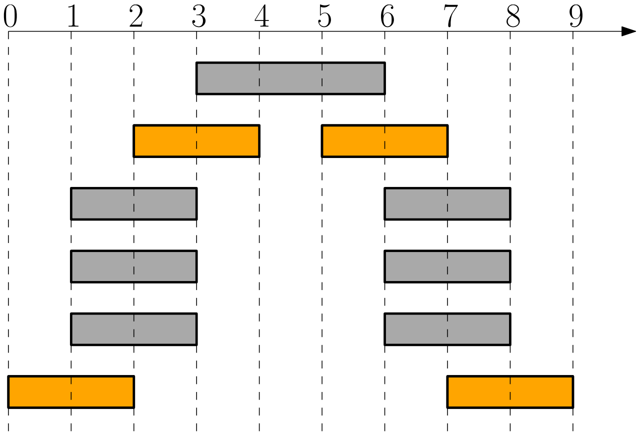
Greedy Algorithm for Interval Scheduling

- Which of the following strategies are safe?
- Schedule the job with the smallest size? No!
- Schedule the job conflicting with smallest number of other jobs? No!



Greedy Algorithm for Interval Scheduling

- Which of the following strategies are safe?
- Schedule the job with the smallest size? No!
- Schedule the job conflicting with smallest number of other jobs? No!



Greedy Algorithm for Interval Scheduling

- Which of the following strategies are safe?
- Schedule the job with the smallest size? No!
- Schedule the job conflicting with smallest number of other jobs? No!

Greedy Algorithm for Interval Scheduling

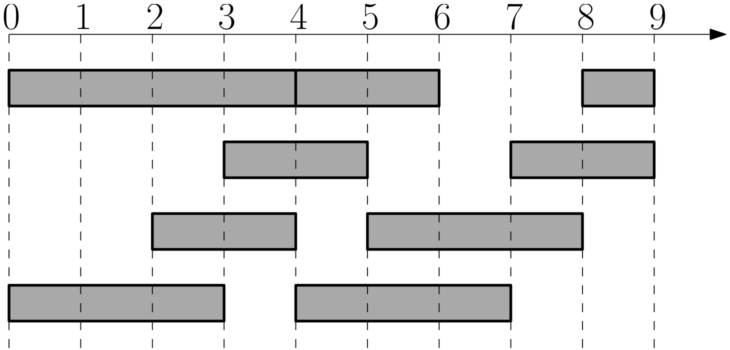
- Which of the following strategies are safe?
- Schedule the job with the smallest size? No!
- Schedule the job conflicting with smallest number of other jobs? No!
- Schedule the job with the earliest finish time?

Greedy Algorithm for Interval Scheduling

- Which of the following strategies are safe?
- Schedule the job with the smallest size? No!
- Schedule the job conflicting with smallest number of other jobs? No!
- Schedule the job with the earliest finish time? Yes!

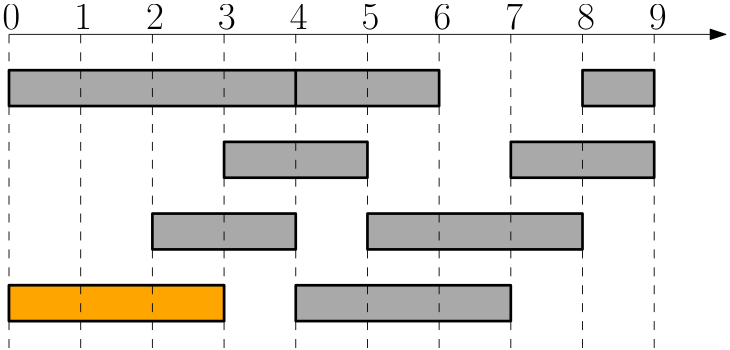
Greedy Algorithm for Interval Scheduling

- Which of the following strategies are safe?
- Schedule the job with the smallest size? No!
- Schedule the job conflicting with smallest number of other jobs? No!
- Schedule the job with the earliest finish time? Yes!



Greedy Algorithm for Interval Scheduling

- Which of the following strategies are safe?
- Schedule the job with the smallest size? No!
- Schedule the job conflicting with smallest number of other jobs? No!
- Schedule the job with the earliest finish time? Yes!



Greedy Algorithm for Interval Scheduling

Lemma It is safe to schedule the job j with the earliest finish time: There is an optimum solution where the job j with the earliest finish time is scheduled.

Proof.

Greedy Algorithm for Interval Scheduling

Lemma It is safe to schedule the job j with the earliest finish time: There is an optimum solution where the job j with the earliest finish time is scheduled.

Proof.

- Take an arbitrary optimum solution S

Greedy Algorithm for Interval Scheduling

Lemma It is safe to schedule the job j with the earliest finish time: There is an optimum solution where the job j with the earliest finish time is scheduled.

Proof.

- Take an arbitrary optimum solution S
- If it contains j, done

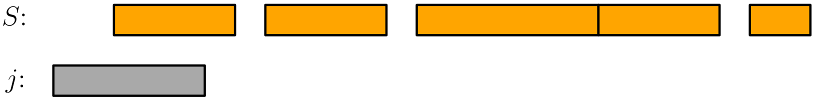
S:

Greedy Algorithm for Interval Scheduling

Lemma It is safe to schedule the job j with the earliest finish time: There is an optimum solution where the job j with the earliest finish time is scheduled.

Proof.

- Take an arbitrary optimum solution S
- If it contains j, done

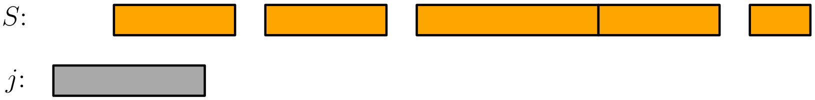


Greedy Algorithm for Interval Scheduling

Lemma It is safe to schedule the job j with the earliest finish time: There is an optimum solution where the job j with the earliest finish time is scheduled.

Proof.

- Take an arbitrary optimum solution S
- If it contains j, done
- Otherwise, replace the first job in S with j to obtain another optimum schedule S'. $\square$

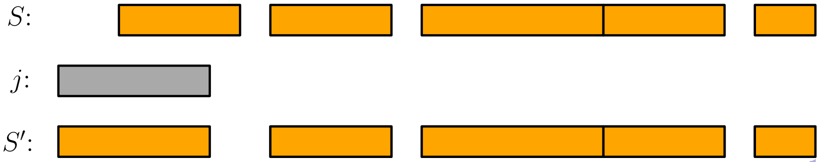


Greedy Algorithm for Interval Scheduling

Lemma It is safe to schedule the job j with the earliest finish time: There is an optimum solution where the job j with the earliest finish time is scheduled.

Proof.

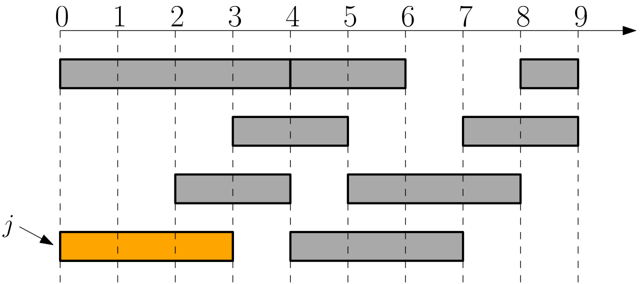
- Take an arbitrary optimum solution S
- If it contains j, done
- Otherwise, replace the first job in S with j to obtain another optimum schedule S'. □



Greedy Algorithm for Interval Scheduling

Lemma It is safe to schedule the job j with the earliest finish time: There is an optimum solution where the job j with the earliest finish time is scheduled.

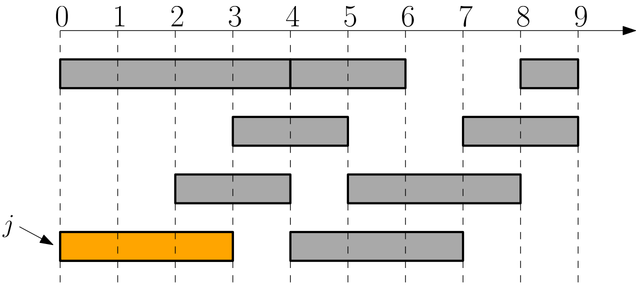
- What is the remaining task after we decided to schedule j?
- Is it another instance of interval scheduling problem?



Greedy Algorithm for Interval Scheduling

Lemma It is safe to schedule the job j with the earliest finish time: There is an optimum solution where the job j with the earliest finish time is scheduled.

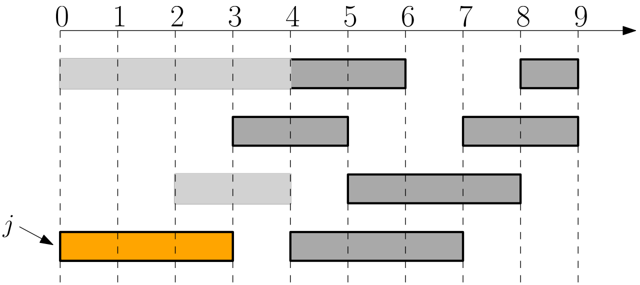
- What is the remaining task after we decided to schedule j?
- Is it another instance of interval scheduling problem? Yes!



Greedy Algorithm for Interval Scheduling

Lemma It is safe to schedule the job j with the earliest finish time: There is an optimum solution where the job j with the earliest finish time is scheduled.

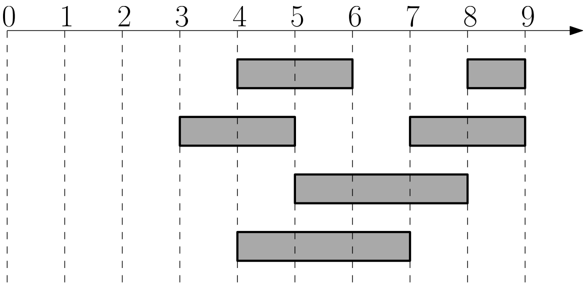
- What is the remaining task after we decided to schedule j?
- Is it another instance of interval scheduling problem? Yes!



Greedy Algorithm for Interval Scheduling

Lemma It is safe to schedule the job j with the earliest finish time: There is an optimum solution where the job j with the earliest finish time is scheduled.

- What is the remaining task after we decided to schedule j?
- Is it another instance of interval scheduling problem? Yes!



Greedy Algorithm for Interval Scheduling

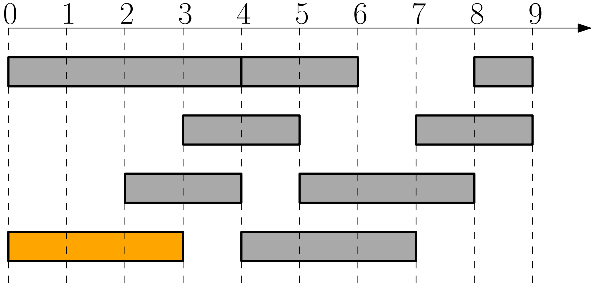
Schedule(s, f, n)

$A \leftarrow \{1, 2, \cdots, n\}, S \leftarrow \emptyset$
while $A \neq \emptyset$ **do**
    $j \leftarrow \arg\min_{j' \in A} f_{j'}$
    $S \leftarrow S \cup \{j\}$; $A \leftarrow \{j' \in A : s_{j'} \geq f_j\}$
return S

Greedy Algorithm for Interval Scheduling

Schedule(s, f, n)

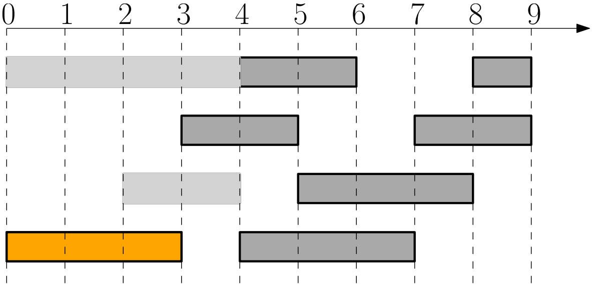
1: $A \leftarrow \{1, 2, \cdots, n\}, S \leftarrow \emptyset$
2: **while** $A \neq \emptyset$ **do**
3: $\quad j \leftarrow \arg\min_{j' \in A} f_{j'}$
4: $\quad S \leftarrow S \cup \{j\}$; $A \leftarrow \{j' \in A : s_{j'} \geq f_j\}$
5: **return** S



Greedy Algorithm for Interval Scheduling

Schedule(s, f, n)

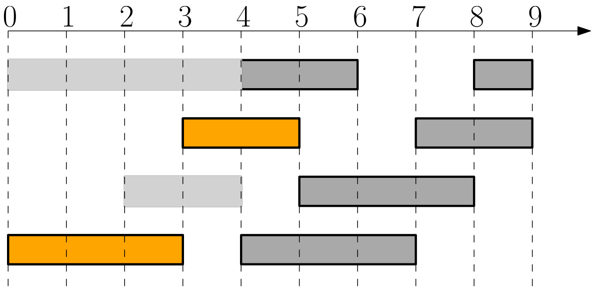
```
A ← {1, 2, ⋯, n}, S ← ∅
while A ≠ ∅ do
    j ← arg min_{j'∈A} f_{j'}
    S ← S ∪ {j}; A ← {j' ∈ A : s_{j'} ≥ f_j}
return S
```



Greedy Algorithm for Interval Scheduling

Schedule(s, f, n)

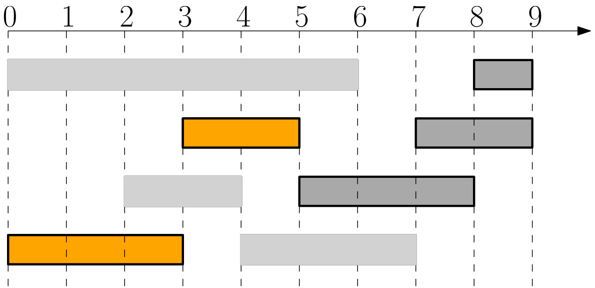
1: $A \leftarrow \{1, 2, \cdots, n\}, S \leftarrow \emptyset$
2: **while** $A \neq \emptyset$ **do**
3: $\quad j \leftarrow \arg\min_{j' \in A} f_{j'}$
4: $\quad S \leftarrow S \cup \{j\}$; $A \leftarrow \{j' \in A : s_{j'} \geq f_j\}$
5: **return** S



Greedy Algorithm for Interval Scheduling

Schedule(s, f, n)

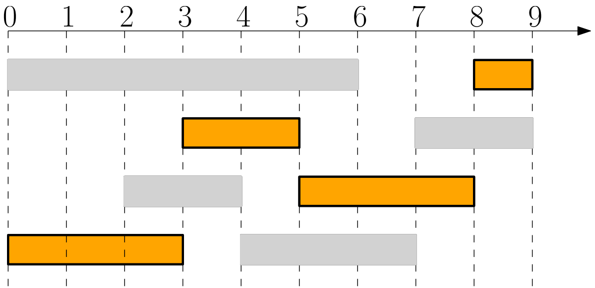
1: $A \leftarrow \{1, 2, \cdots, n\}, S \leftarrow \emptyset$
2: **while** $A \neq \emptyset$ **do**
3: $\quad j \leftarrow \arg\min_{j' \in A} f_{j'}$
4: $\quad S \leftarrow S \cup \{j\}$; $A \leftarrow \{j' \in A : s_{j'} \geq f_j\}$
5: **return** S



Greedy Algorithm for Interval Scheduling

Schedule(s, f, n)

$A \leftarrow \{1, 2, \cdots, n\}, S \leftarrow \emptyset$
while $A \neq \emptyset$ **do**
    $j \leftarrow \arg\min_{j' \in A} f_{j'}$
    $S \leftarrow S \cup \{j\}$; $A \leftarrow \{j' \in A : s_{j'} \geq f_j\}$
return S



Greedy Algorithm for Interval Scheduling

Schedule(s, f, n)

$A \leftarrow \{1, 2, \cdots, n\}, S \leftarrow \emptyset$
while $A \neq \emptyset$ **do**
    $j \leftarrow \arg\min_{j' \in A} f_{j'}$
    $S \leftarrow S \cup \{j\}$; $A \leftarrow \{j' \in A : s_{j'} \geq f_j\}$
return S

Running time of algorithm?

Greedy Algorithm for Interval Scheduling

Schedule(s, f, n)

$A \leftarrow \{1, 2, \cdots, n\}, S \leftarrow \emptyset$
while $A \neq \emptyset$ **do**
  $j \leftarrow \arg\min_{j' \in A} f_{j'}$
  $S \leftarrow S \cup \{j\}$; $A \leftarrow \{j' \in A : s_{j'} \geq f_j\}$
return S

Running time of algorithm?

- Naive implementation: $O(n^2)$ time

Greedy Algorithm for Interval Scheduling

Schedule(s, f, n)

$A \leftarrow \{1, 2, \cdots, n\}, S \leftarrow \emptyset$
while $A \neq \emptyset$ **do**
    $j \leftarrow \arg\min_{j' \in A} f_{j'}$
    $S \leftarrow S \cup \{j\}$; $A \leftarrow \{j' \in A : s_{j'} \geq f_j\}$
return S

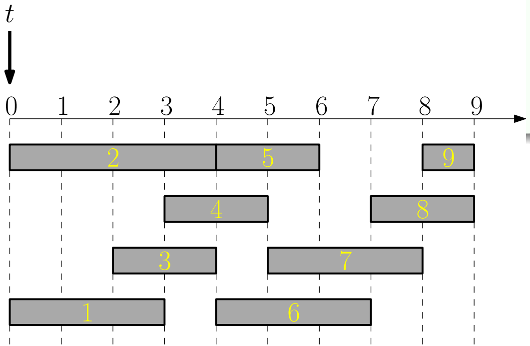
Running time of algorithm?

- Naive implementation: $O(n^2)$ time
- Clever implementation: $O(n \lg n)$ time

Clever Implementation of Greedy Algorithm

Schedule(s, f, n)

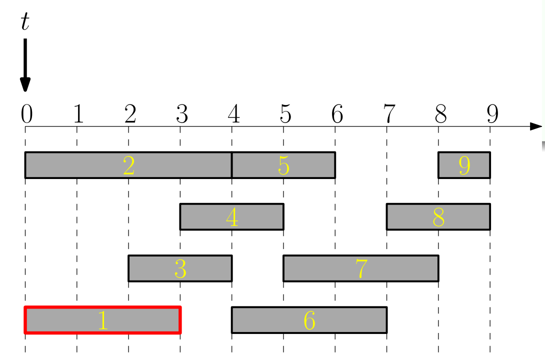
sort jobs according to f values
$t \leftarrow 0$, $S \leftarrow \emptyset$
for every $j \in [n]$ according to non-decreasing order of f_j **do**
    if $s_j \geq t$ **then**
        $S \leftarrow S \cup \{j\}$
        $t \leftarrow f_j$
return S



Clever Implementation of Greedy Algorithm

Schedule(s, f, n)

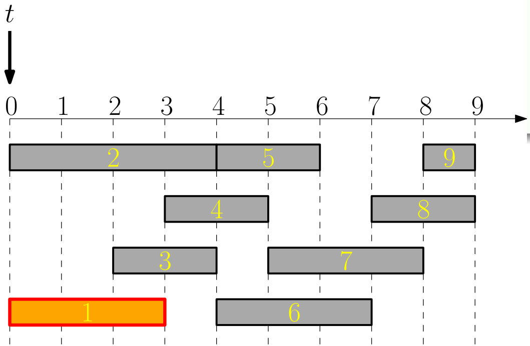
sort jobs according to f values
$t \leftarrow 0$, $S \leftarrow \emptyset$
for every $j \in [n]$ according to non-decreasing order of f_j **do**
    if $s_j \geq t$ **then**
        $S \leftarrow S \cup \{j\}$
        $t \leftarrow f_j$
return S



Clever Implementation of Greedy Algorithm

Schedule(s, f, n)

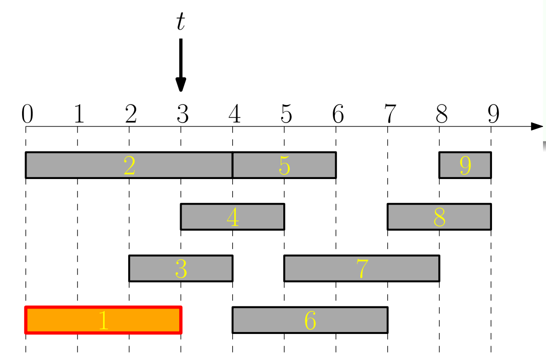
sort jobs according to f values
$t \leftarrow 0$, $S \leftarrow \emptyset$
for every $j \in [n]$ according to non-decreasing order of f_j **do**
  if $s_j \geq t$ **then**
    $S \leftarrow S \cup \{j\}$
    $t \leftarrow f_j$
return S



Clever Implementation of Greedy Algorithm

Schedule(s, f, n)

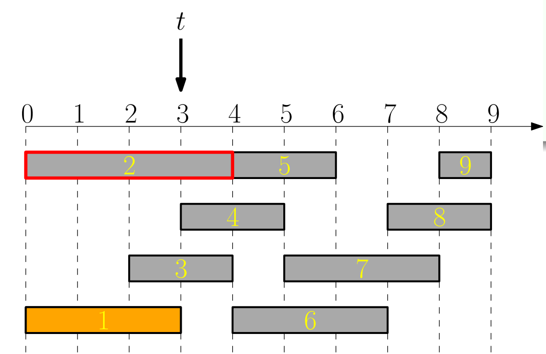
sort jobs according to f values
$t \leftarrow 0$, $S \leftarrow \emptyset$
for every $j \in [n]$ according to non-decreasing order of f_j **do**
    if $s_j \geq t$ **then**
        $S \leftarrow S \cup \{j\}$
        $t \leftarrow f_j$
return S



Clever Implementation of Greedy Algorithm

Schedule(s, f, n)

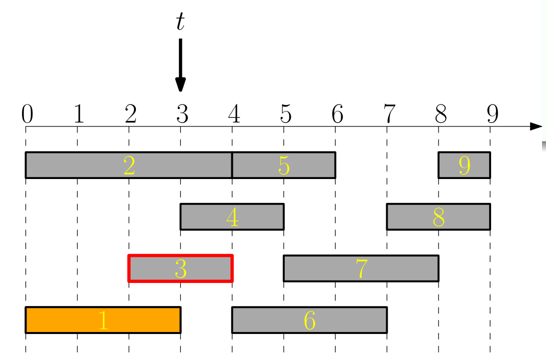
```
sort jobs according to f values
t ← 0, S ← ∅
for every j ∈ [n] according to non-decreasing order of f_j do
    if s_j ≥ t then
        S ← S ∪ {j}
        t ← f_j
return S
```



Clever Implementation of Greedy Algorithm

Schedule(s, f, n)

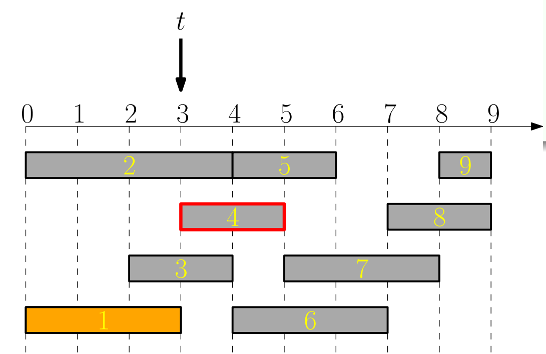
sort jobs according to f values
$t \leftarrow 0$, $S \leftarrow \emptyset$
for every $j \in [n]$ according to non-decreasing order of f_j **do**
  if $s_j \geq t$ **then**
    $S \leftarrow S \cup \{j\}$
    $t \leftarrow f_j$
return S



Clever Implementation of Greedy Algorithm

Schedule(s, f, n)

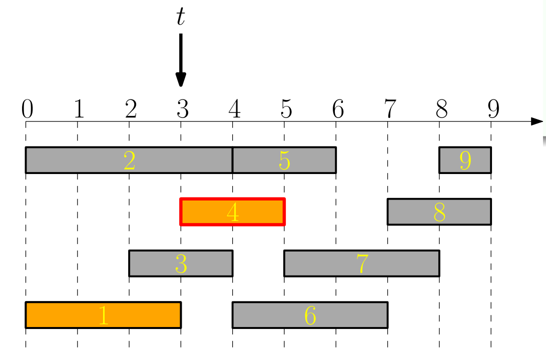
sort jobs according to f values
$t \leftarrow 0$, $S \leftarrow \emptyset$
for every $j \in [n]$ according to non-decreasing order of f_j **do**
    if $s_j \geq t$ **then**
        $S \leftarrow S \cup \{j\}$
        $t \leftarrow f_j$
return S



Clever Implementation of Greedy Algorithm

Schedule(s, f, n)

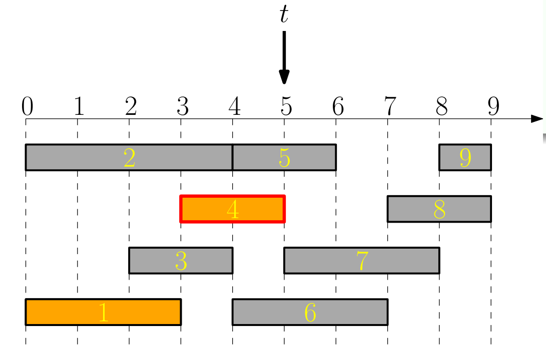
```
sort jobs according to f values
t ← 0, S ← ∅
for every j ∈ [n] according to non-decreasing order of f_j do
    if s_j ≥ t then
        S ← S ∪ {j}
        t ← f_j
return S
```



Clever Implementation of Greedy Algorithm

Schedule(s, f, n)

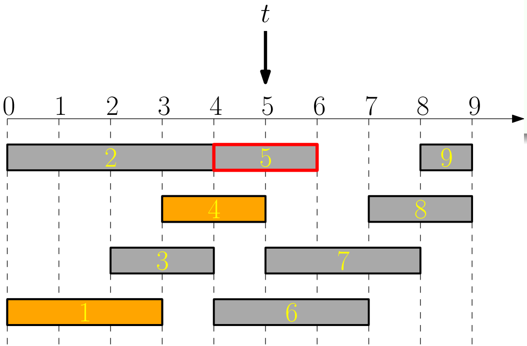
sort jobs according to f values
$t \leftarrow 0$, $S \leftarrow \emptyset$
for every $j \in [n]$ according to non-decreasing order of f_j **do**
  if $s_j \geq t$ **then**
    $S \leftarrow S \cup \{j\}$
    $t \leftarrow f_j$
return S



Clever Implementation of Greedy Algorithm

Schedule(s, f, n)

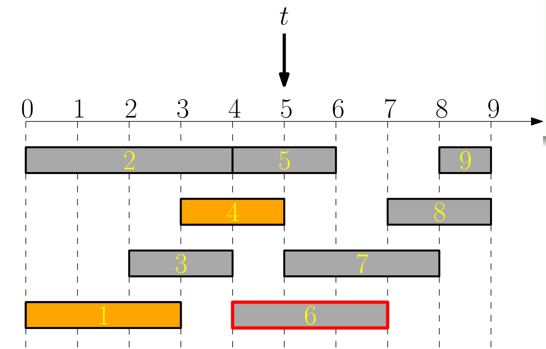
sort jobs according to f values
$t \leftarrow 0$, $S \leftarrow \emptyset$
for every $j \in [n]$ according to non-decreasing order of f_j **do**
  if $s_j \geq t$ **then**
    $S \leftarrow S \cup \{j\}$
    $t \leftarrow f_j$
return S



Clever Implementation of Greedy Algorithm

Schedule(s, f, n)

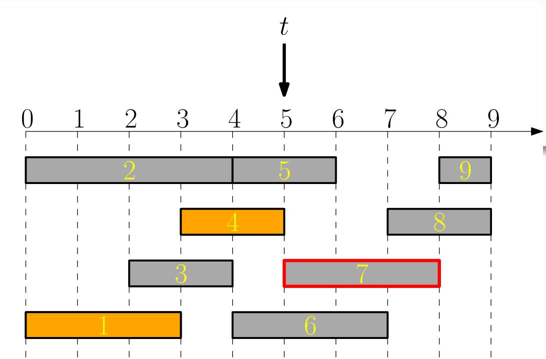
sort jobs according to f values
$t \leftarrow 0$, $S \leftarrow \emptyset$
for every $j \in [n]$ according to non-decreasing order of f_j **do**
    if $s_j \geq t$ **then**
        $S \leftarrow S \cup \{j\}$
        $t \leftarrow f_j$
return S



Clever Implementation of Greedy Algorithm

Schedule(s, f, n)

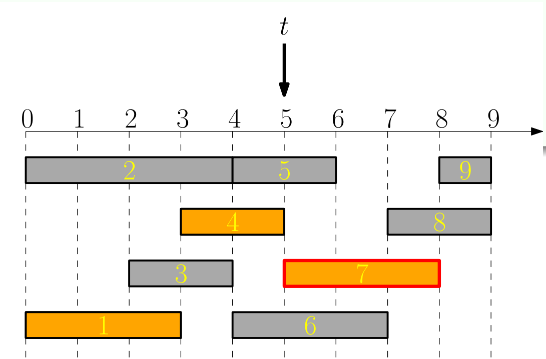
sort jobs according to f values
$t \leftarrow 0$, $S \leftarrow \emptyset$
for every $j \in [n]$ according to non-decreasing order of f_j **do**
    if $s_j \geq t$ **then**
        $S \leftarrow S \cup \{j\}$
        $t \leftarrow f_j$
return S



Clever Implementation of Greedy Algorithm

Schedule(s, f, n)

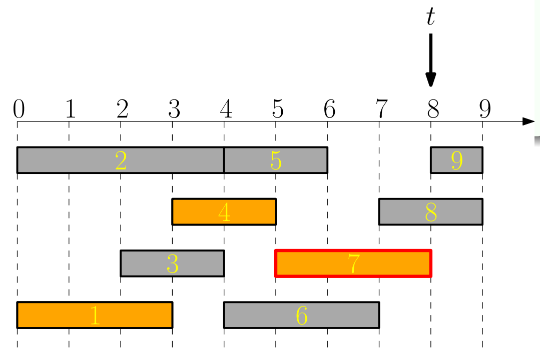
```
sort jobs according to f values
t ← 0, S ← ∅
for every j ∈ [n] according to non-decreasing order of f_j do
    if s_j ≥ t then
        S ← S ∪ {j}
        t ← f_j
return S
```



Clever Implementation of Greedy Algorithm

Schedule(s, f, n)

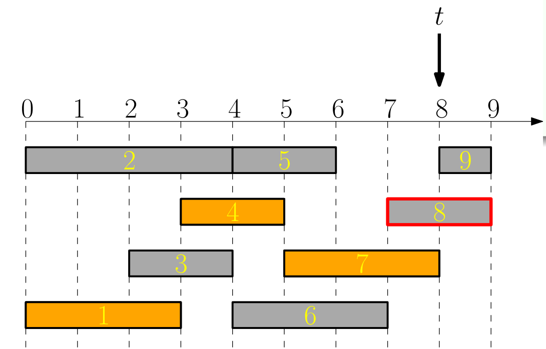
sort jobs according to f values
$t \leftarrow 0$, $S \leftarrow \emptyset$
for every $j \in [n]$ according to non-decreasing order of f_j **do**
    if $s_j \geq t$ **then**
        $S \leftarrow S \cup \{j\}$
        $t \leftarrow f_j$
return S



Clever Implementation of Greedy Algorithm

Schedule(s, f, n)

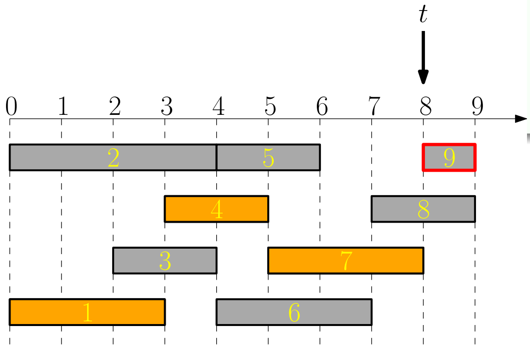
sort jobs according to f values
$t \leftarrow 0$, $S \leftarrow \emptyset$
for every $j \in [n]$ according to non-decreasing order of f_j **do**
  if $s_j \geq t$ **then**
    $S \leftarrow S \cup \{j\}$
    $t \leftarrow f_j$
return S



Clever Implementation of Greedy Algorithm

Schedule(s, f, n)

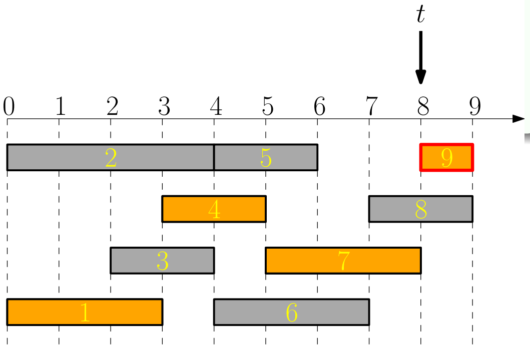
```
sort jobs according to f values
t ← 0, S ← ∅
for every j ∈ [n] according to non-decreasing order of f_j do
    if s_j ≥ t then
        S ← S ∪ {j}
        t ← f_j
return S
```



Clever Implementation of Greedy Algorithm

Schedule(s, f, n)

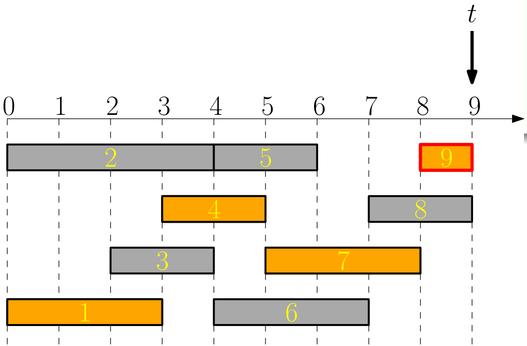
1: sort jobs according to f values
2: $t \leftarrow 0$, $S \leftarrow \emptyset$
3: **for** every $j \in [n]$ according to non-decreasing order of f_j **do**
4: **if** $s_j \geq t$ **then**
5: $S \leftarrow S \cup \{j\}$
6: $t \leftarrow f_j$
7: **return** S



Clever Implementation of Greedy Algorithm

Schedule(s, f, n)

1: sort jobs according to f values
2: $t \leftarrow 0$, $S \leftarrow \emptyset$
3: **for** every $j \in [n]$ according to non-decreasing order of f_j **do**
4:     **if** $s_j \geq t$ **then**
5:         $S \leftarrow S \cup \{j\}$
6:         $t \leftarrow f_j$
7: **return** S



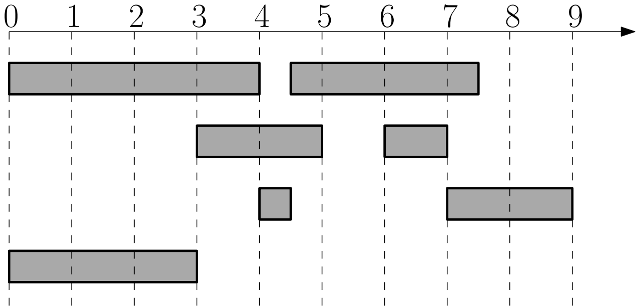
Outline

Interval Partitioning

Input: n jobs, job i with start time s_i and finish time f_i

i and j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ are disjoint

Output: A minimum number of machines to schedule all jobs so that all jobs on a single machine are compatible.

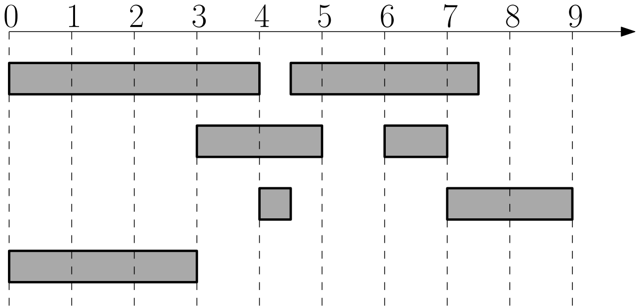


Interval Partitioning

Input: n jobs, job i with start time s_i and finish time f_i

i and j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ are disjoint

Output: A minimum number of machines to schedule all jobs so that all jobs on a single machine are compatible.

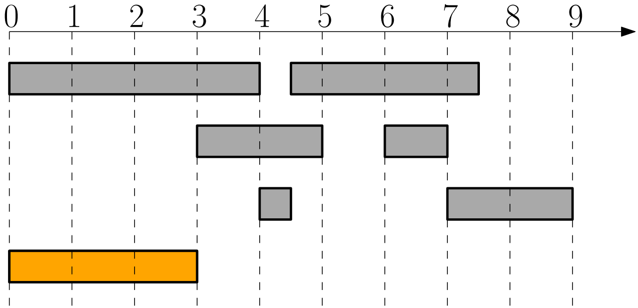


Interval Partitioning

Input: n jobs, job i with start time s_i and finish time f_i

i and j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ are disjoint

Output: A minimum number of machines to schedule all jobs so that all jobs on a single machine are compatible.

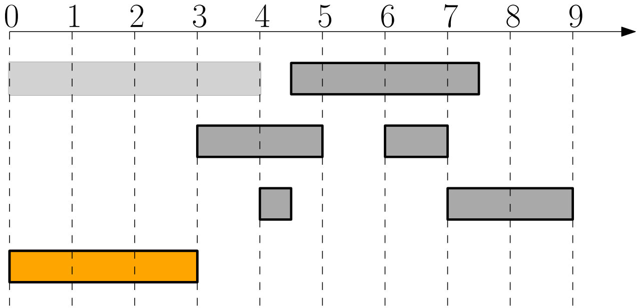


Interval Partitioning

Input: n jobs, job i with start time s_i and finish time f_i

i and j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ are disjoint

Output: A minimum number of machines to schedule all jobs so that all jobs on a single machine are compatible.

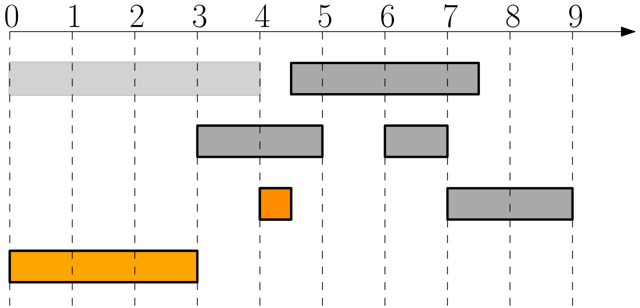


Interval Partitioning

Input: n jobs, job i with start time s_i and finish time f_i

i and j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ are disjoint

Output: A minimum number of machines to schedule all jobs so that all jobs on a single machine are compatible.

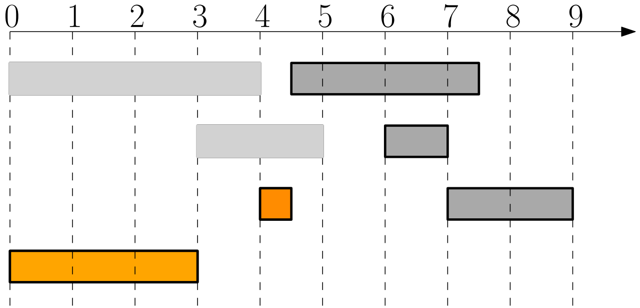


Interval Partitioning

Input: n jobs, job i with start time s_i and finish time f_i

i and j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ are disjoint

Output: A minimum number of machines to schedule all jobs so that all jobs on a single machine are compatible.

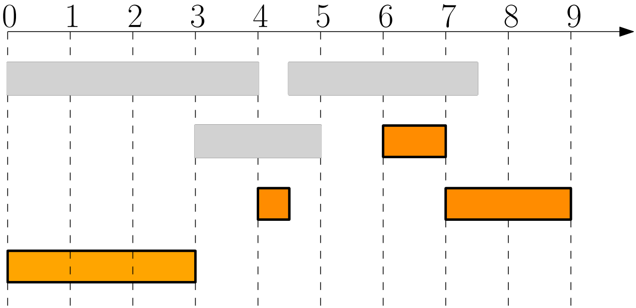


Interval Partitioning

Input: n jobs, job i with start time s_i and finish time f_i

i and j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ are disjoint

Output: A minimum number of machines to schedule all jobs so that all jobs on a single machine are compatible.

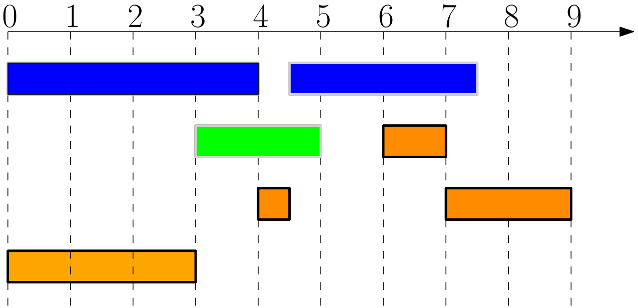


Interval Partitioning

Input: n jobs, job i with start time s_i and finish time f_i

i and j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ are disjoint

Output: A minimum number of machines to schedule all jobs so that all jobs on a single machine are compatible.



Interval Partitioning

Input: n jobs, job i with start time s_i and finish time f_i

i and j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ are disjoint

Output: A minimum number of machines to schedule all jobs so that all jobs on a single machine are compatible.