Box Packing

Input: n boxes of capacities $c_1, c_2, \cdots, c_n$

m items of sizes $s_1, s_2, \cdots, s_m$

Can put at most 1 item in a box

Item j can be put into box i if $s_j \leq c_i$

Output: A way to put as many items as possible in the boxes.

Box Packing

Input: n boxes of capacities $c_1, c_2, \cdots, c_n$

m items of sizes $s_1, s_2, \cdots, s_m$

Can put at most 1 item in a box

Item j can be put into box i if $s_j \leq c_i$

Output: A way to put as many items as possible in the boxes.

Example:

- Box capacities: 60, 40, 25, 15, 12
- Item sizes: 45, 42, 20, 19, 16
- Can put 3 items in boxes: $45 \to 60, 20 \to 40, 19 \to 25$

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

Designing a Reasonable Strategy for Box Packing

- Q: Take box 1. Which item should we put in box 1?

Greedy Algorithm

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

Designing a Reasonable Strategy for Box Packing

- Q: Take box 1. Which item should we put in box 1?
- A: The item of the largest size that can be put into the box.

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

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

Lemma The strategy that put into box 1 the largest item it can hold is "safe": There is an optimum solution in which box 1 contains the largest item it can hold.

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

Lemma The strategy that put into box 1 the largest item it can hold is "safe": There is an optimum solution in which box 1 contains the largest item it can hold.

- Intuition: putting the item gives us the easiest residual problem.

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

Lemma The strategy that put into box 1 the largest item it can hold is "safe": There is an optimum solution in which box 1 contains the largest item it can hold.

- Intuition: putting the item gives us the easiest residual problem.
- formal proof via exchanging argument:

Lemma There is an optimum solution in which box 1 contains the largest item it can hold.

Lemma There is an optimum solution in which box 1 contains the largest item it can hold.

Proof.

- Let $j =$ largest item that box 1 can hold.

Lemma There is an optimum solution in which box 1 contains the largest item it can hold.

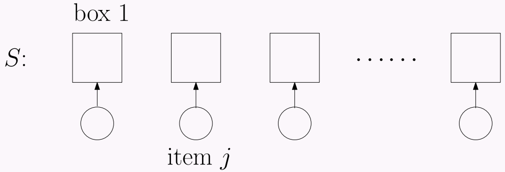
Proof.

- Let $j =$ largest item that box 1 can hold.
- Take any optimum solution S. If j is put into Box 1 in S, done.

Lemma There is an optimum solution in which box 1 contains the largest item it can hold.

Proof.

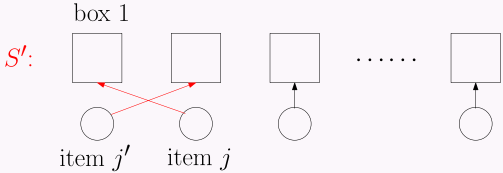
- Let $j =$ largest item that box 1 can hold.
- Take any optimum solution S. If j is put into Box 1 in S, done.
- Otherwise, assume this is what happens in S:



Lemma There is an optimum solution in which box 1 contains the largest item it can hold.

Proof.

- Let j = largest item that box 1 can hold.
- Take any optimum solution S. If j is put into Box 1 in S, done.
- Otherwise, assume this is what happens in S:

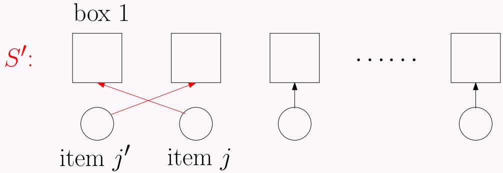


- $s_{j'} \leq s_j$, and swapping gives another solution S'

Lemma There is an optimum solution in which box 1 contains the largest item it can hold.

Proof.

- Let $j =$ largest item that box 1 can hold.
- Take any optimum solution S. If j is put into Box 1 in S, done.
- Otherwise, assume this is what happens in S:



- $s_{j'} \leq s_j$, and swapping gives another solution S'
- S' is also an optimum solution. In S', j is put into Box 1. □

- Notice that the exchanging operation is only for the sake of analysis; it is not a part of the algorithm.

- Notice that the exchanging operation is only for the sake of analysis; it is not a part of the algorithm.

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

- Notice that the exchanging operation is only for the sake of analysis; it is not a part of the algorithm.

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

- Trivial: we decided to put Item j into Box 1, and the remaining instance is obtained by removing Item j and Box 1.

Generic Greedy Algorithm

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

Greedy Algorithm for Box Packing

$T \leftarrow \{1, 2, 3, \cdots, m\}$
for $i \leftarrow 1$ to n **do**
    if some item in T can be put into box i **then**
        $j \leftarrow$ the largest item in T that can be put into box i
        print("put item j in box i")
        $T \leftarrow T \setminus \{j\}$

Why "Safety" + "Self-reduce" $\Longrightarrow$ Optimality?

- Let BP(B, T) denote a box-packing instance.
- $\phi(1, 2, ..., m) \mapsto \{1, 2, ..., n, \text{NULL}\}$ denote packing strategy. e.g., $\phi(2) = 3$ means item 2 is put into box 3.
- val(ϕ) := the number of items packed by ϕ.
- ϕ_g: the packing strategy obtained by greedy algorithm.

Proof.

- Base case: When $|B| = 1$ or $|T| = 1$.
- Inductive case: (Hypothesis) Assume Greedy alg solves BP(B', T') optimally for $|B'| = n - 1$ and $|T'| = m - 1$.

□

Why "Safety" + "Self-reduce" $\implies$ Optimality?

Proof.

- (Induction) Wlog, let π be the optimal solution matches our greedy sol on BP(B, T), saying $\pi(j) = 1$.
- By self-reduce: BP$(B \setminus \{1\}, T \setminus \{j\})$ is a smaller BP instance.
- π and ϕ_g onto BP$(B \setminus \{1\}, T \setminus \{j\})$, denoted as π' and ϕ'_g.
- By Inductive hypothesis, ϕ'_g is the optimal sol for BP$(B \setminus \{1\}, T \setminus \{j\})$.
- val$(\pi) \geq$ val$(\phi_g) = 1+$ val$(\phi'_g) \geq 1+$val(π') =val(π).

□

Running time

Generic Greedy Algorithm

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

Greedy Algorithm for Box Packing

$T \leftarrow \{1, 2, 3, \cdots, m\}$
for $i \leftarrow 1$ to n **do**
    if some item in T can be put into box i **then**
        $j \leftarrow$ the largest item in T that can be put into box i
        print("put item j in box i")
        $T \leftarrow T \setminus \{j\}$

Running time

Generic Greedy Algorithm

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

Greedy Algorithm for Box Packing

$T \leftarrow \{1, 2, 3, \cdots, m\}$
for $i \leftarrow 1$ to n **do**
    if some item in T can be put into box i **then**
        $j \leftarrow$ the largest item in T that can be put into box i
        print("put item j in box i")
        $T \leftarrow T \setminus \{j\}$

Running time

Generic Greedy Algorithm

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

Greedy Algorithm for Box Packing

```
T ← {1, 2, 3, ··· , m}
for i ← 1 to n do
    if some item in T can be put into box i then
        j ← the largest item in T that can be put into box i
        print("put item j in box i")
        T ← T \ {j}
```

- With sorted item-sizes and box-capacities, running time is $O(\max\{n, m\})$.

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

```
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.

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

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.

- 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

1 Toy Example: Box Packing

2 Interval Scheduling

3 Offline Caching
- Heap: Concrete Data Structure for Priority Queue

4 Data Compression and Huffman Code

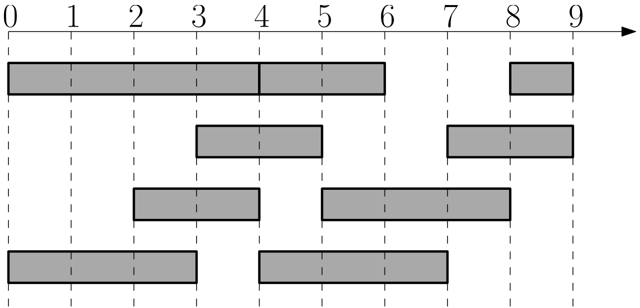
5 Summary

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

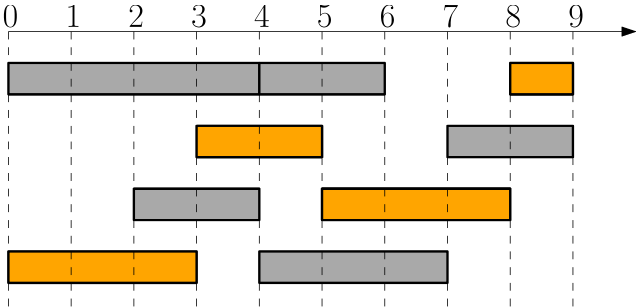


Interval Scheduling

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

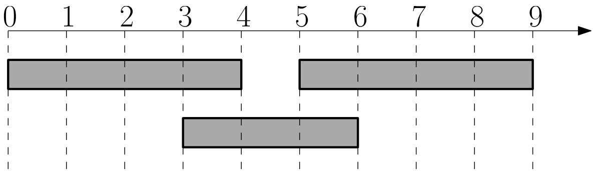
- Which of the following strategies are safe?
- Schedule the job with the smallest size?

Greedy Algorithm for Interval Scheduling

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

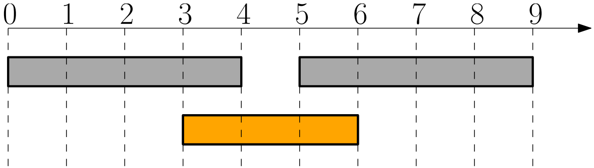
Greedy Algorithm for Interval Scheduling

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



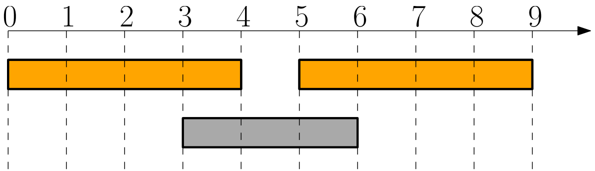
Greedy Algorithm for Interval Scheduling

- 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

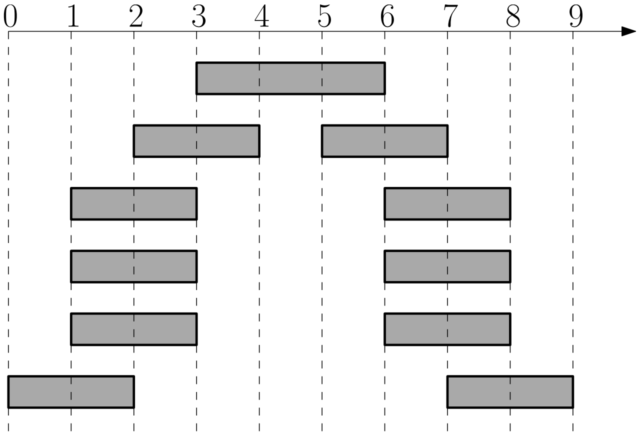
- Which of the following strategies are safe?
- Schedule the job with the smallest size? No!
- Schedule the job conflicting with smallest number of other jobs?

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!

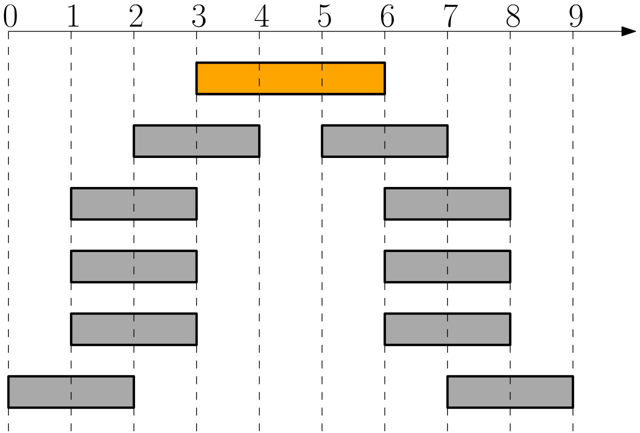
Greedy Algorithm for Interval Scheduling

- 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!



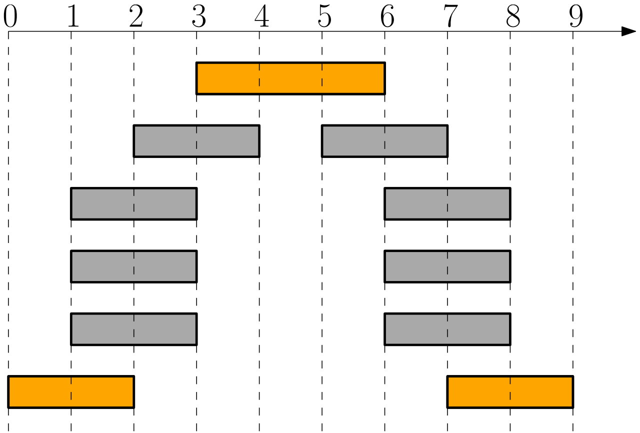
Greedy Algorithm for Interval Scheduling

- 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!



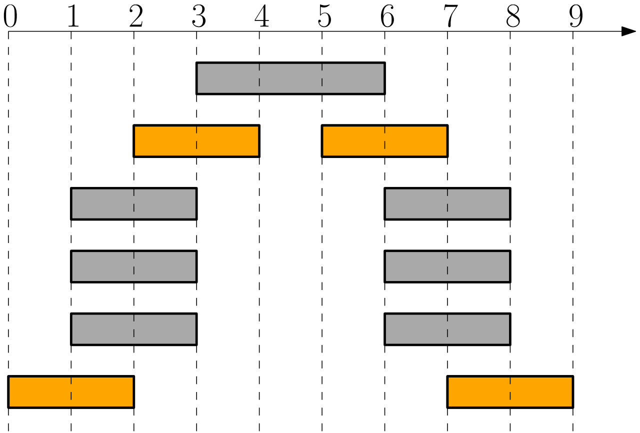
Greedy Algorithm for Interval Scheduling

- 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

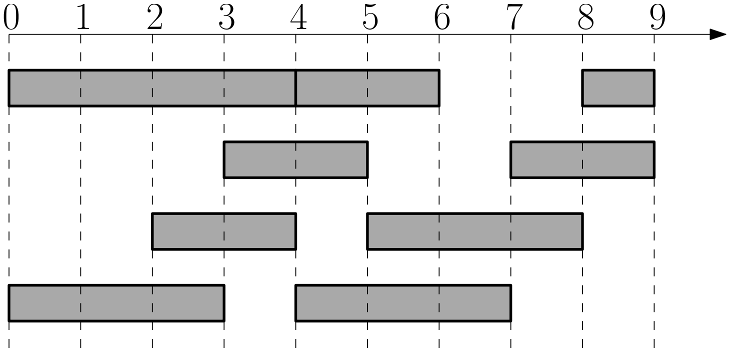
- Which of the following strategies are safe?
- Schedule the job with the smallest size? No!
- Schedule the job conflicting with smallest number of other jobs? No!
- Schedule the job with the earliest finish time?

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!

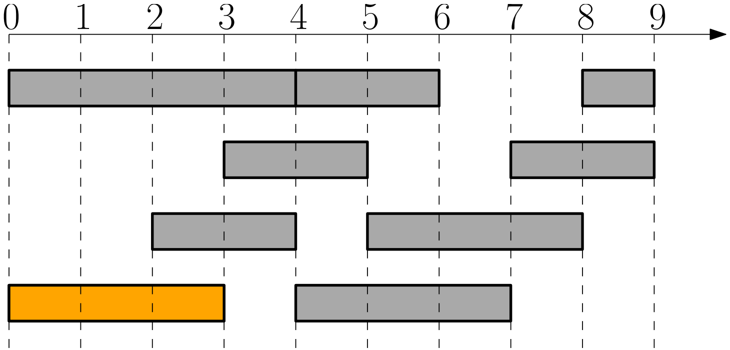
Greedy Algorithm for Interval Scheduling

- 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.

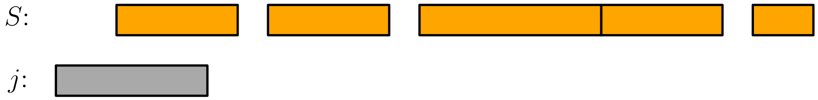
- Take an arbitrary optimum solution S
- If it contains j, done

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

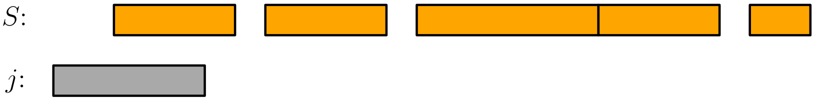


Greedy Algorithm for Interval Scheduling

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'. □

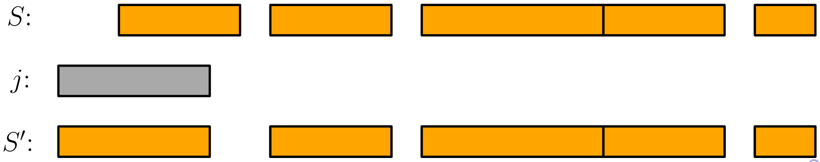


Greedy Algorithm for Interval Scheduling

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.

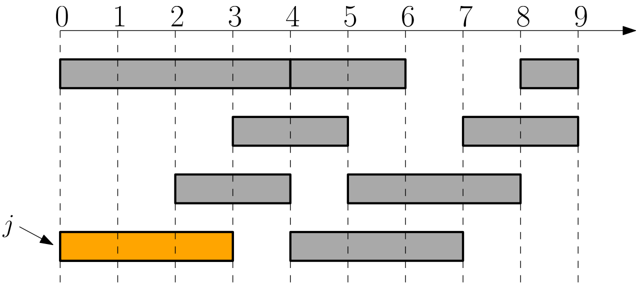
- Take an arbitrary optimum solution S
- If it contains j, done
- Otherwise, replace the first job in S with j to obtain another optimum schedule S'. □



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.

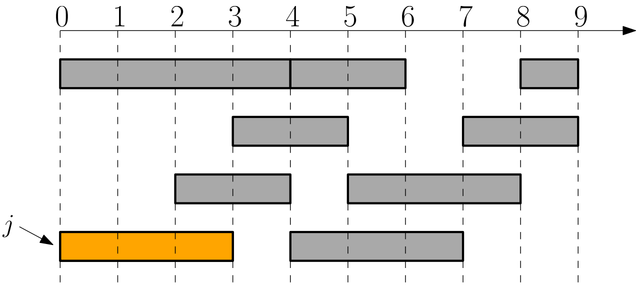
- What is the remaining task after we decided to schedule j?
- Is it another instance of interval scheduling problem?



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.

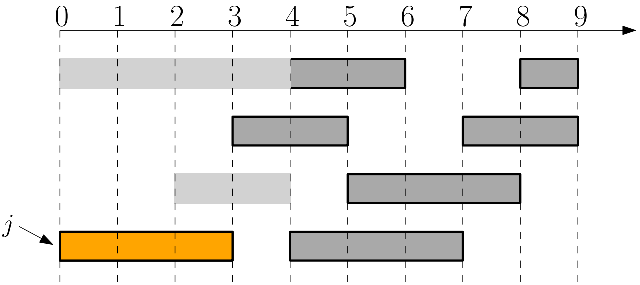
- What is the remaining task after we decided to schedule j?
- Is it another instance of interval scheduling problem? Yes!



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.

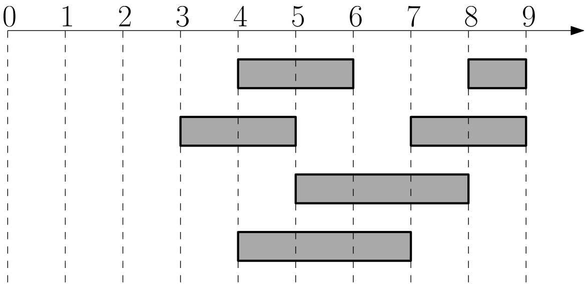
- What is the remaining task after we decided to schedule j?
- Is it another instance of interval scheduling problem? Yes!



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

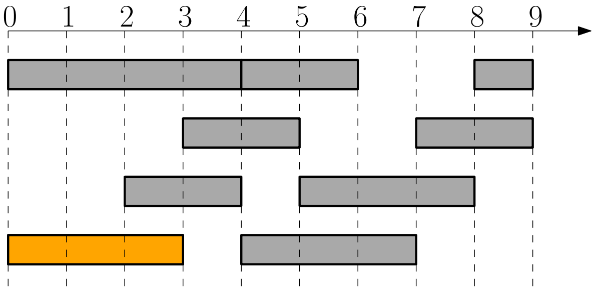
Schedule(s, f, n)

$A \leftarrow \{1, 2, \cdots, n\}, S \leftarrow \emptyset$
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)

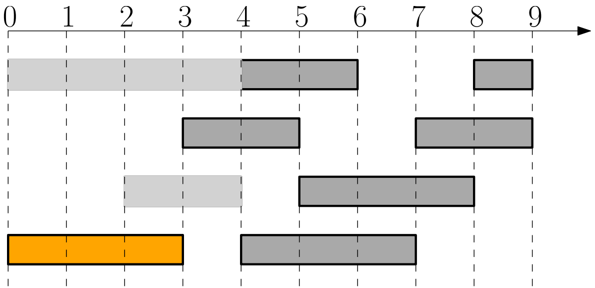
$A \leftarrow \{1, 2, \cdots, n\}, S \leftarrow \emptyset$
while $A \neq \emptyset$ **do**
    $j \leftarrow \arg\min_{j' \in A} f_{j'}$
    $S \leftarrow S \cup \{j\}$; $A \leftarrow \{j' \in A : s_{j'} \geq f_j\}$
return S



Greedy Algorithm for Interval Scheduling

Schedule(s, f, n)

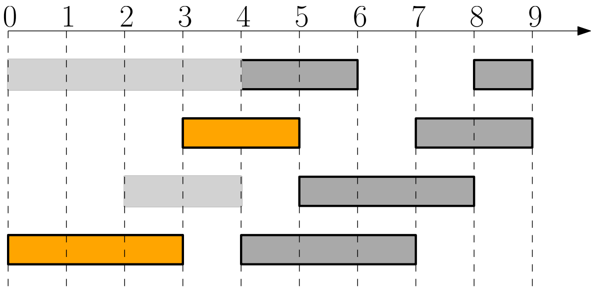
$A \leftarrow \{1, 2, \cdots, n\}, S \leftarrow \emptyset$
while $A \neq \emptyset$ **do**
    $j \leftarrow \arg\min_{j' \in A} f_{j'}$
    $S \leftarrow S \cup \{j\}$; $A \leftarrow \{j' \in A : s_{j'} \geq f_j\}$
return S



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)

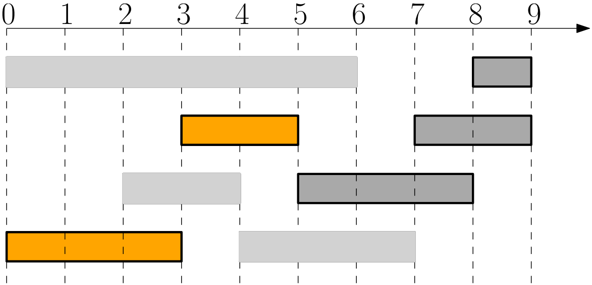
$A \leftarrow \{1, 2, \cdots, n\}, S \leftarrow \emptyset$

while $A \neq \emptyset$ **do**

  $j \leftarrow \arg\min_{j' \in A} f_{j'}$

  $S \leftarrow S \cup \{j\}$; $A \leftarrow \{j' \in A : s_{j'} \geq f_j\}$

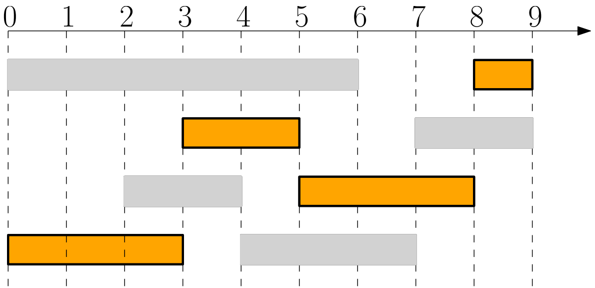
return S



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 ← {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
```

Running time of algorithm?

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

Running time of algorithm?

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