Greedy Algorithms
CSCI 532

Single Room Scheduling

Goal: Assign courses to a single classroom.

Single Room Scheduling

Input:

- $C = \{c_1, c_2, \ldots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Single Room Scheduling

Input:

- $C = \{c_1, c_2, \dots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Single Room Scheduling

Input:

- $C = \{c_1, c_2, \dots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of compatible courses.

I.e., Fill a single room up with the most possible courses.

Single Room Scheduling

Input:

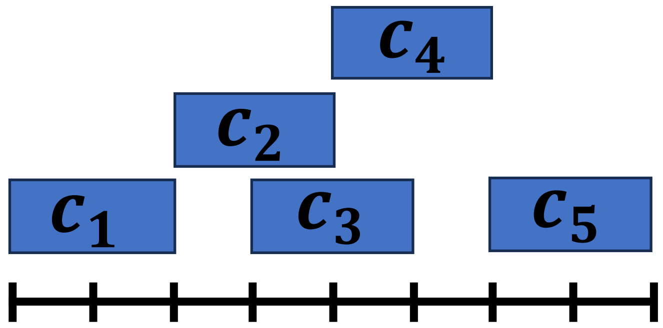
- $C = \{c_1, c_2, \dots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of compatible courses.

i	1	2	3	4	5
s_i	1	3	4	5	7
f_i	3	5	6	7	9



Single Room Scheduling

Input:

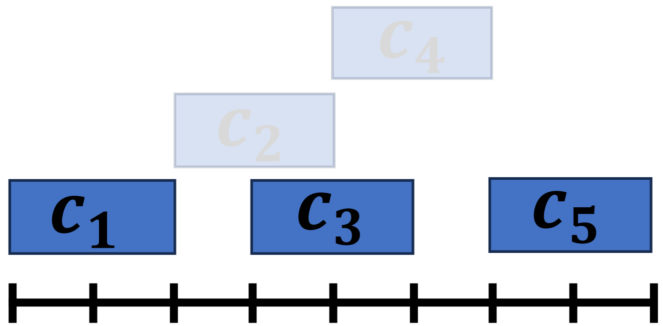
- $C = \{c_1, c_2, \ldots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of compatible courses.

i	1	2	3	4	5
s_i	1	3	4	5	7
f_i	3	5	6	7	9



Single Room Scheduling

Input:

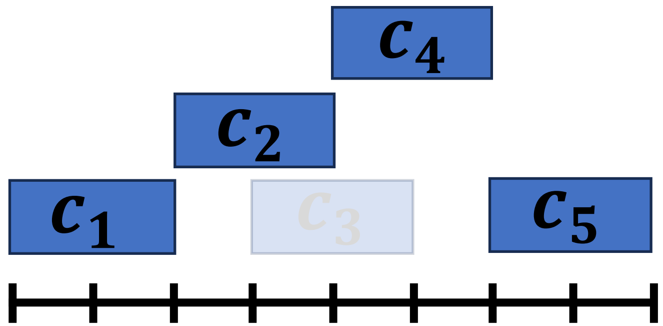
- $C = \{c_1, c_2, \dots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of compatible courses.

i	1	2	3	4	5
s_i	1	3	4	5	7
f_i	3	5	6	7	9



Single Room Scheduling

Input:

- $C = \{c_1, c_2, \ldots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of compatible courses.

Greedy selection criteria?

i	1	2	3	4	5
s_i	1	3	4	5	7
f_i	3	5	6	7	9

Single Room Scheduling

Input:

- $C = \{c_1, c_2, \dots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of compatible courses.

Greedy selection criteria?

Smallest conflict first.

In each iteration, pick the course that overlaps with the smallest number of other courses.

Single Room Scheduling

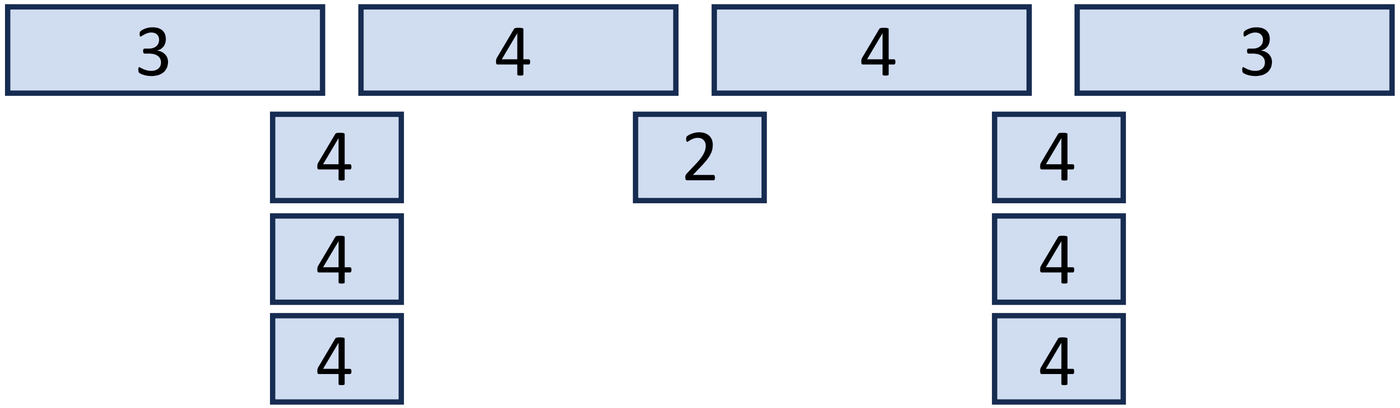
Input:

- $C = \{c_1, c_2, \ldots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of compatible courses.



Greedy selection criteria?

Smallest conflict first.

In each iteration, pick the course that overlaps with the smallest number of other courses.

Single Room Scheduling

Input:

- $C = \{c_1, c_2, \dots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of compatible courses.

Greedy selection criteria?

Smallest duration first.

In each iteration, pick the course that takes the least amount of time.

Single Room Scheduling

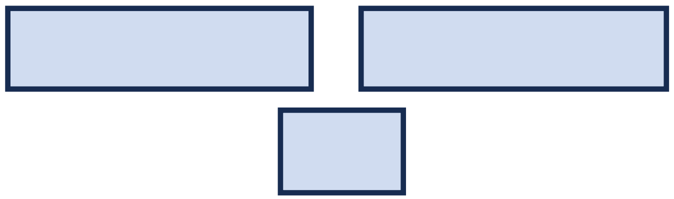
Input:

- $C = \{c_1, c_2, \dots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of compatible courses.



Greedy selection criteria?

Smallest duration first.

In each iteration, pick the course that takes the least amount of time.

Single Room Scheduling

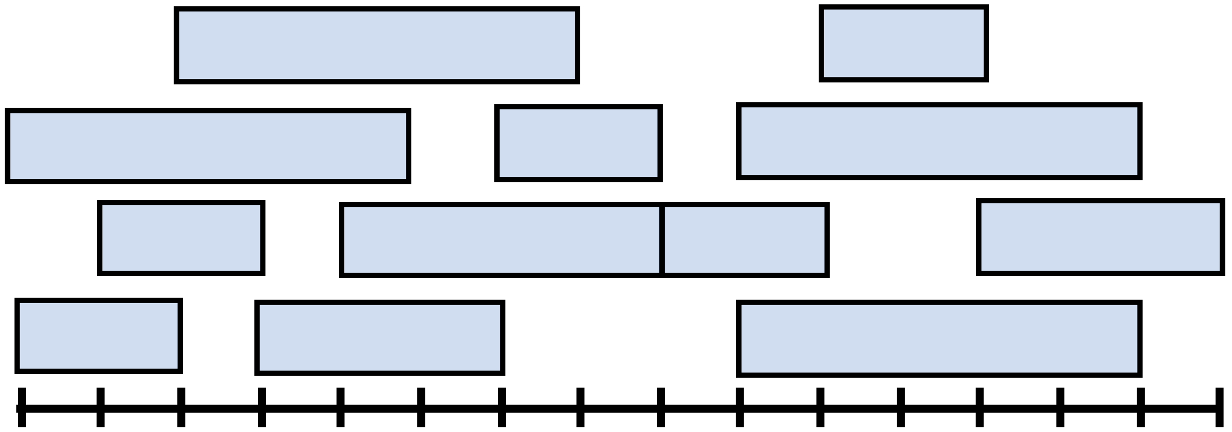
Input:

- $C = \{c_1, c_2, \dots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of compatible courses.



Greedy selection criteria?

Earliest compatible finish.

In each iteration, pick the course that ends earliest and is compatible with existing schedule.

Single Room Scheduling

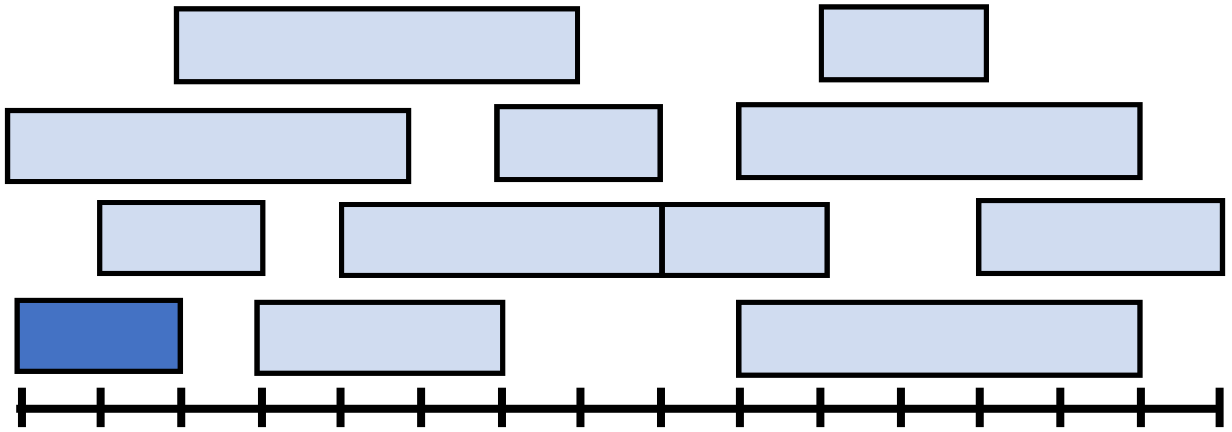
Input:

- $C = \{c_1, c_2, \dots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of compatible courses.



Greedy selection criteria?

Earliest compatible finish.

In each iteration, pick the course that ends earliest and is compatible with existing schedule.

Single Room Scheduling

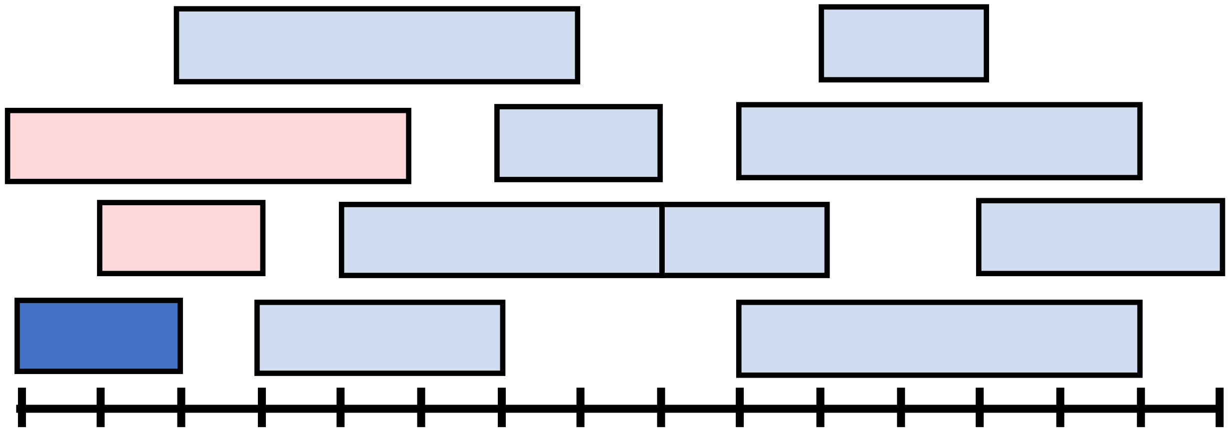
Input:

- $C = \{c_1, c_2, \dots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of compatible courses.



Greedy selection criteria?

Earliest compatible finish.

In each iteration, pick the course that ends earliest and is compatible with existing schedule.

Single Room Scheduling

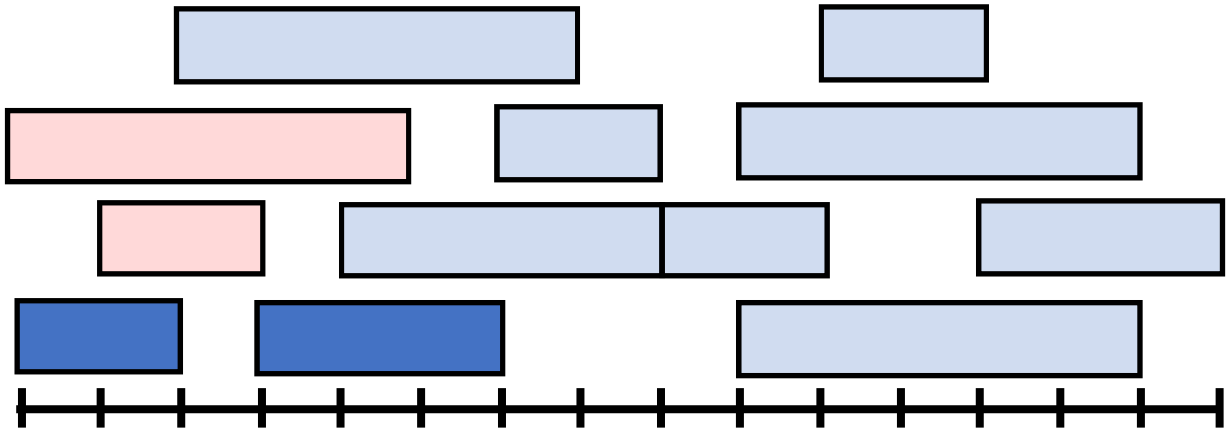
Input:

- $C = \{c_1, c_2, \ldots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of compatible courses.



Greedy selection criteria?

Earliest compatible finish.

In each iteration, pick the course that ends earliest and is compatible with existing schedule.

Single Room Scheduling

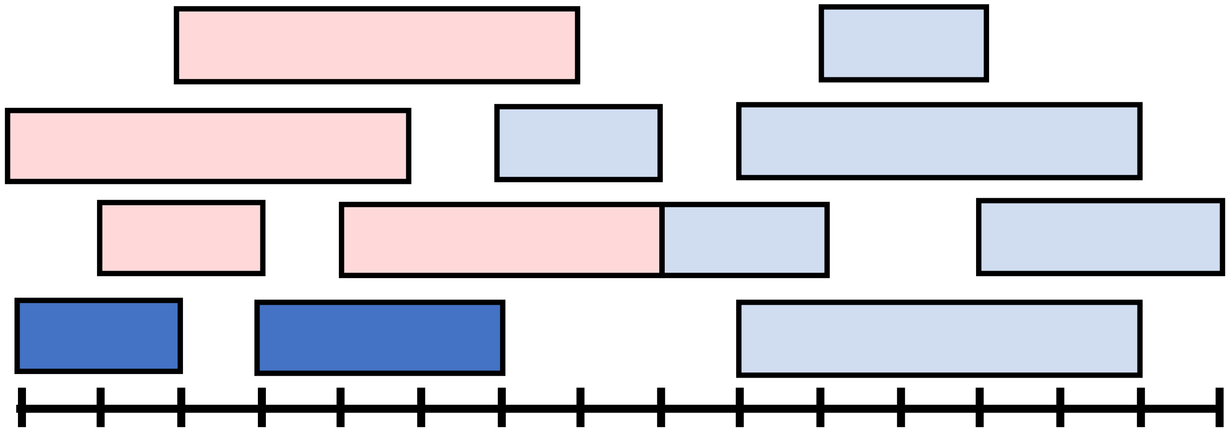
Input:

- $C = \{c_1, c_2, \ldots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of compatible courses.



Greedy selection criteria?

Earliest compatible finish.

In each iteration, pick the course that ends earliest and is compatible with existing schedule.

Single Room Scheduling

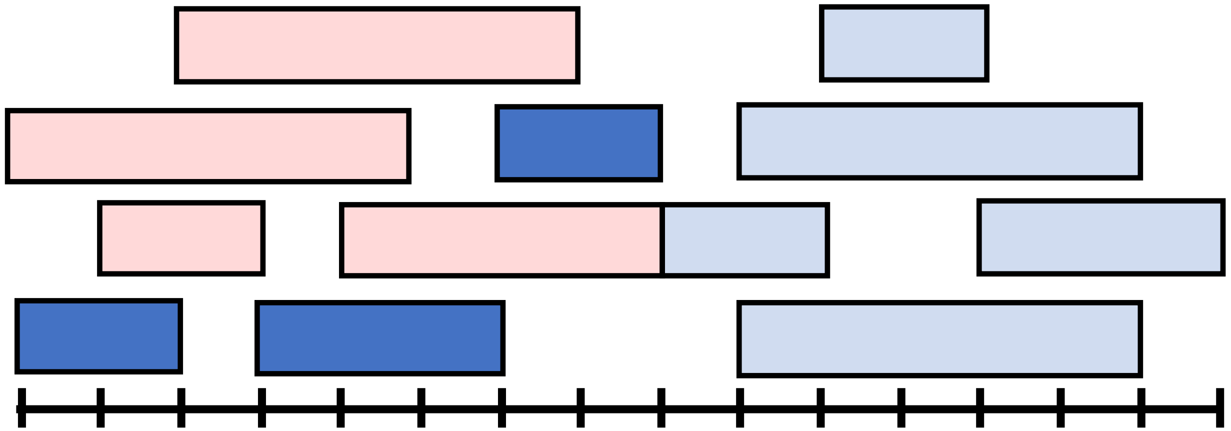
Input:

- $C = \{c_1, c_2, \ldots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of compatible courses.



Greedy selection criteria?

Earliest compatible finish.

In each iteration, pick the course that ends earliest and is compatible with existing schedule.

Single Room Scheduling

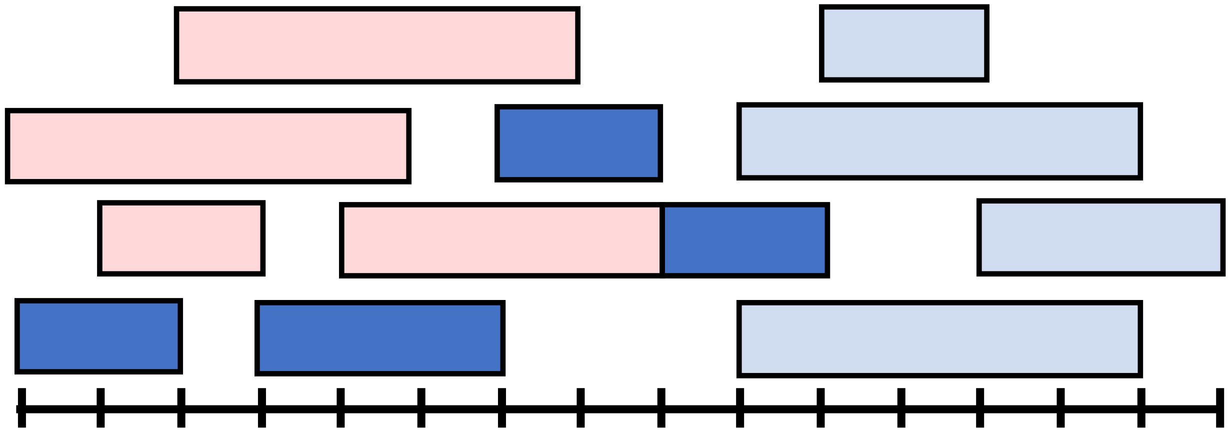
Input:

- $C = \{c_1, c_2, \ldots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of compatible courses.



Greedy selection criteria?

Earliest compatible finish.

In each iteration, pick the course that ends earliest and is compatible with existing schedule.

Single Room Scheduling

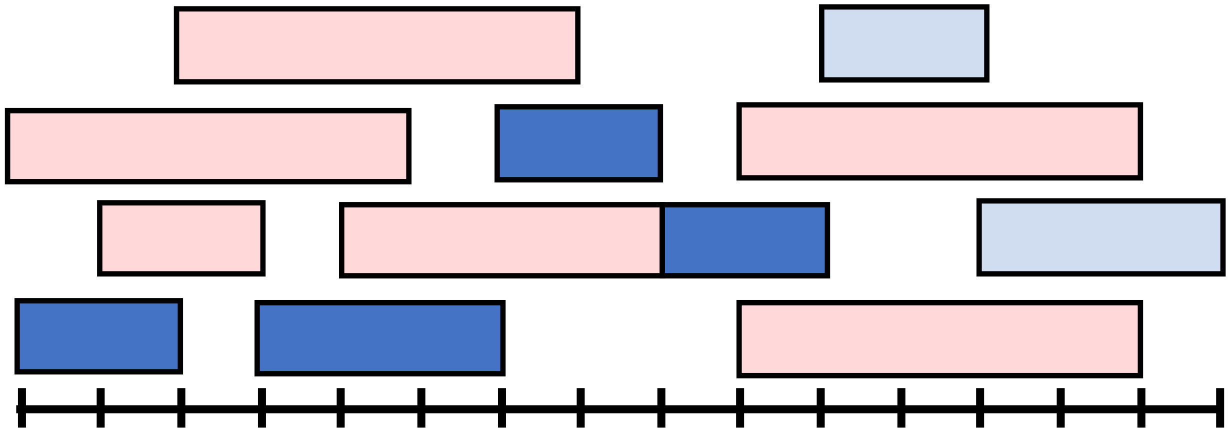
Input:

- $C = \{c_1, c_2, \ldots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of compatible courses.



Greedy selection criteria?

Earliest compatible finish.

In each iteration, pick the course that ends earliest and is compatible with existing schedule.

Single Room Scheduling

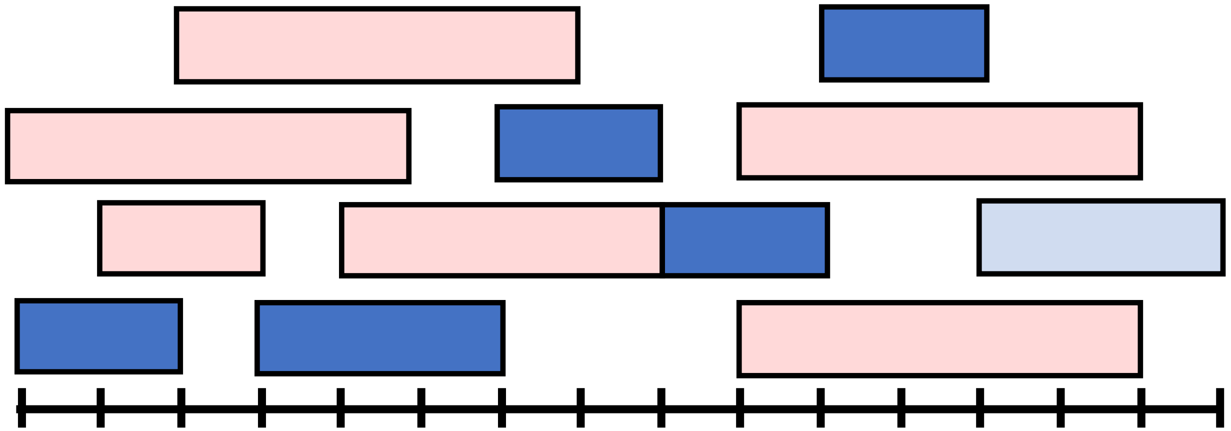
Input:

- $C = \{c_1, c_2, \ldots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of compatible courses.



Greedy selection criteria?

Earliest compatible finish.

In each iteration, pick the course that ends earliest and is compatible with existing schedule.

Single Room Scheduling

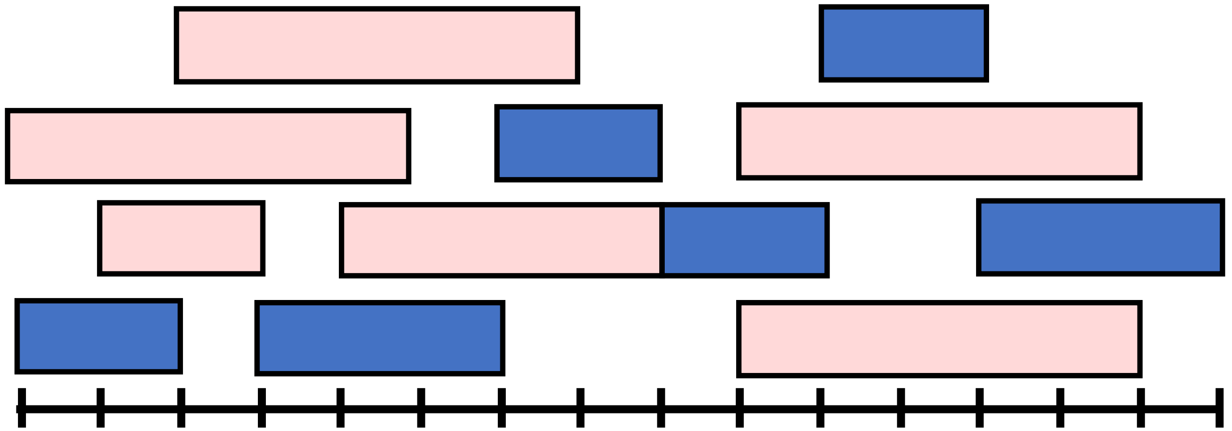
Input:

- $C = \{c_1, c_2, \ldots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of compatible courses.



Greedy selection criteria?

Earliest compatible finish.

In each iteration, pick the course that ends earliest and is compatible with existing schedule.

Single Room Scheduling

Valid?

Running Time?

Performance?

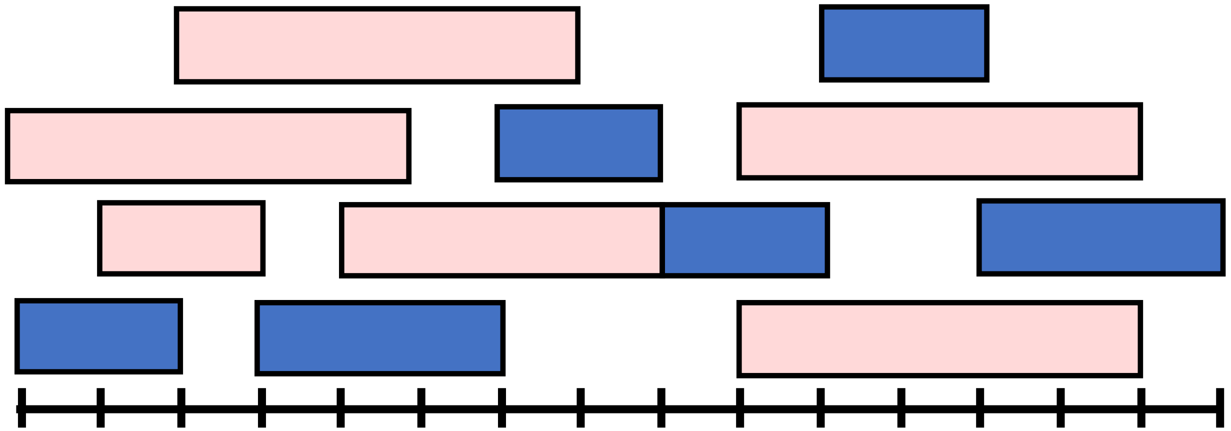
Input:

- $C = \{c_1, c_2, \dots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of compatible courses.



Greedy selection criteria?

Earliest compatible finish.

In each iteration, pick the course that ends earliest and is compatible with existing schedule.

Single Room Scheduling

Valid? Selected compatible courses.

Running Time?

Performance?

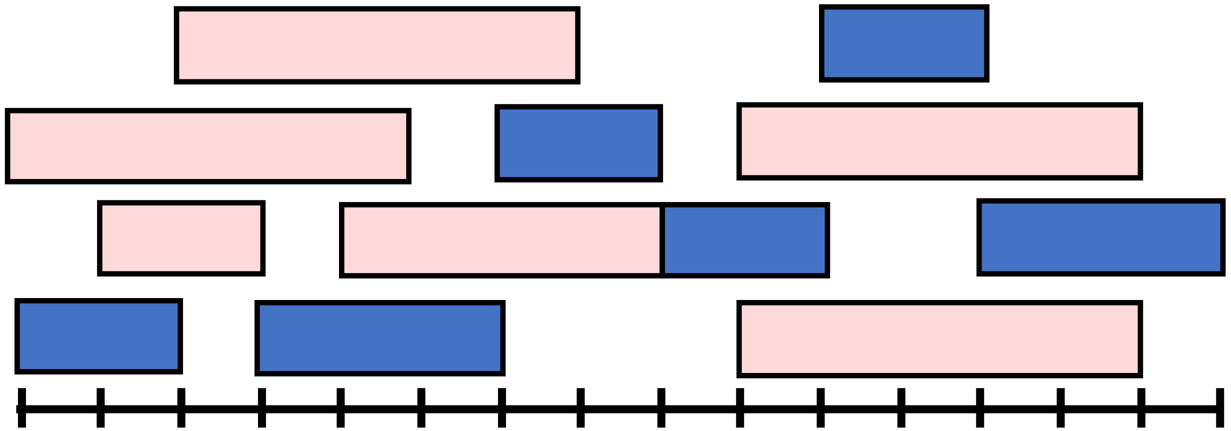
Input:

- $C = \{c_1, c_2, \ldots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of compatible courses.



Greedy selection criteria?

Earliest compatible finish.

In each iteration, pick the course that ends earliest and is compatible with existing schedule.

Single Room Scheduling

Valid? Selected compatible courses.

Running Time?

Performance?

Input:

- $C = \{c_1, c_2, \ldots, c_n\}$ – set of courses
- $c_i = [s_i, f_i)$ – start and finish times

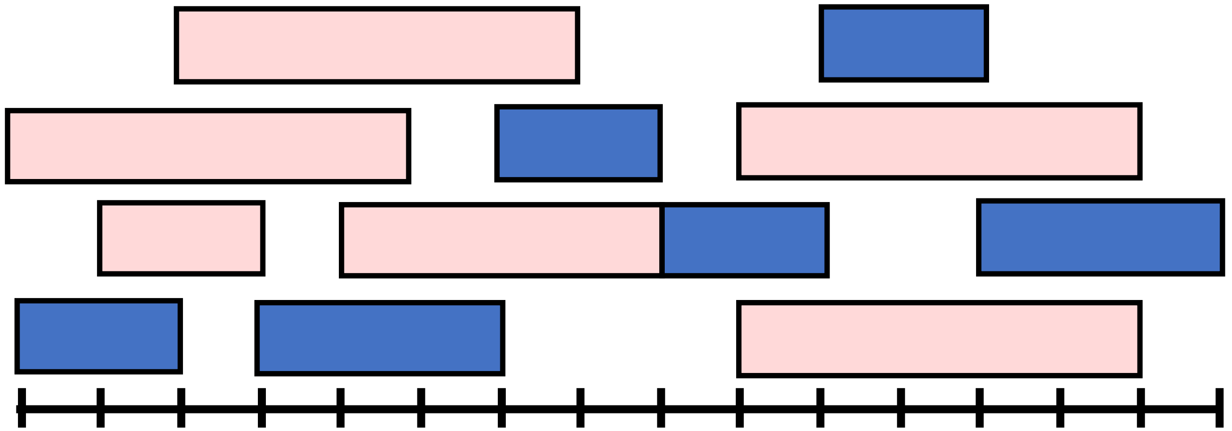
Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ a

Goal: Select a maximum sized subset of

Implementation Plan:

1. Sort by increasing finish times.
2. Select first course.
3. Iterate through list looking for first compatible course.
4. Repeat.



Greedy selection criteria?

Earliest compatible finish.

In each iteration, pick the course that ends earliest and is compatible with existing schedule.

Single Room Scheduling

Valid? Selected compatible courses.

Running Time? $O(n \log n)$

Performance?

Input:

- $C = \{c_1, c_2, \ldots, c_n\}$ – set of courses
- $c_i = [s_i, f_i)$ – start and finish times

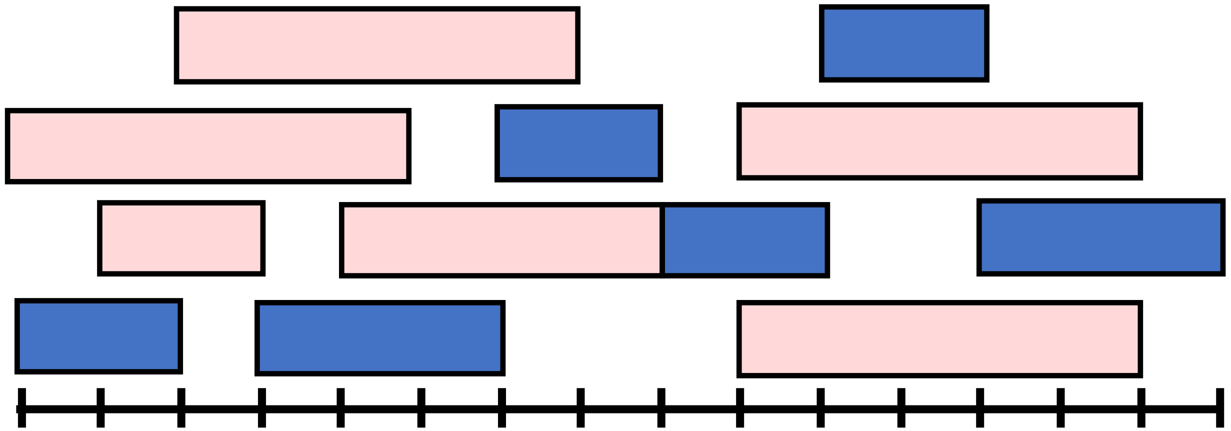
Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ a

Goal: Select a maximum sized subset of

Implementation Plan:

1. Sort by increasing finish times.
2. Select first course.
3. Iterate through list looking for first compatible course.
4. Repeat.



Greedy selection criteria?

Earliest compatible finish.

In each iteration, pick the course that ends earliest and is compatible with existing schedule.

Single Room Scheduling

Greedy decision: Select the next course with the earliest compatible finish time.

Proof of optimality:

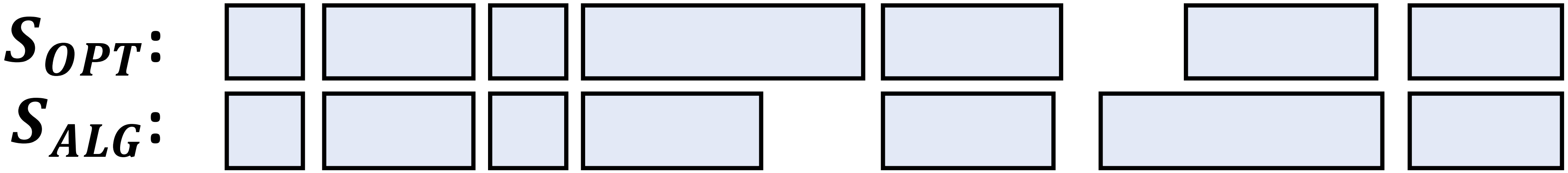
Plan: Turn a hypothetical optimal solution into the algorithm's solution without changing the cost (i.e., number of courses) and without violating course compatibility.

Single Room Scheduling

<u>Greedy decision:</u> Select the next course with the earliest compatible finish time.

<u>Proof of optimality:</u> Let C be the set of courses, $S_{ALG} \subseteq C$ be the greedy algorithm's selection, and $S_{OPT} \subseteq C$ be an optimal selection, all sorted by increasing finish time.

<u>Plan:</u> Turn a hypothetical optimal solution into the algorithm's solution without changing the cost (i.e., number of courses) and without violating course compatibility.



Single Room Scheduling

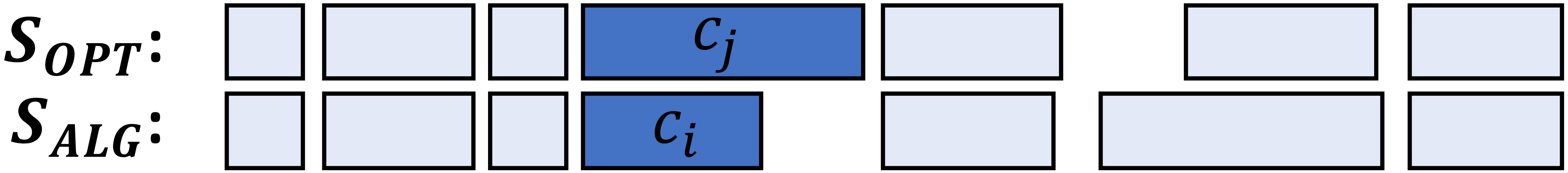
Greedy decision: Select the next course with the earliest compatible finish time.

Proof of optimality: Let C be the set of courses, $S_{ALG} \subseteq C$ be the greedy algorithm's selection, and $S_{OPT} \subseteq C$ be an optimal selection, all sorted by increasing finish time.

Suppose $S_{ALG}[i] = S_{OPT}[i]$, for all $i < k$ and $S_{ALG}[k] = c_i \neq c_j = S_{OPT}[k]$.

Suppose S_{ALG} and S_{OPT} schedule the same courses up until course k.

Plan: Turn a hypothetical optimal solution into the algorithm's solution without changing the cost (i.e., number of courses) and without violating course compatibility.



Single Room Scheduling

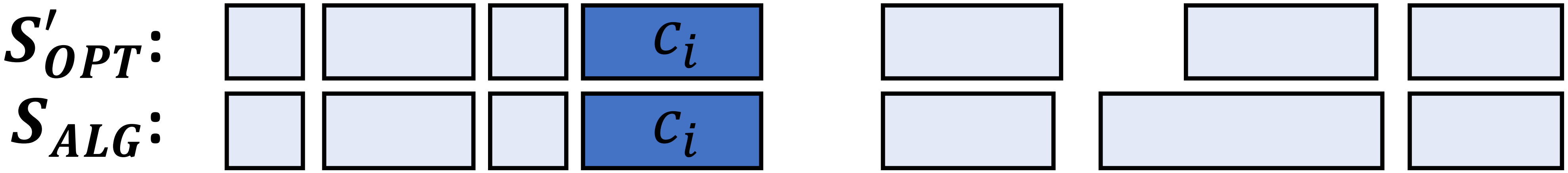
<u>Greedy decision:</u> Select the next course with the earliest compatible finish time.

<u>Proof of optimality:</u> Let C be the set of courses, $S_{ALG} \subseteq C$ be the greedy algorithm's selection, and $S_{OPT} \subseteq C$ be an optimal selection, all sorted by increasing finish time.

Suppose $S_{ALG}[i] = S_{OPT}[i]$, for all $i < k$ and $S_{ALG}[k] = c_i \neq c_j = S_{OPT}[k]$.

Create the revised schedule $S'_{OPT} = S_{OPT} \setminus \{c_j\} \cup \{c_i\}$. (I.e., Swap $S_{ALG}[k]$ for $S_{OPT}[k]$)

<u>Plan:</u> Turn a hypothetical optimal solution into the algorithm's solution without changing the cost (i.e., number of courses) and without violating course compatibility.



Single Room Scheduling

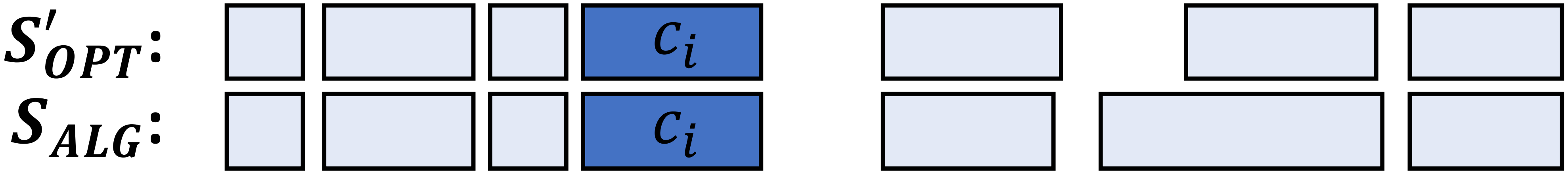
<u>Greedy decision:</u> Select the next course with the earliest compatible finish time.

<u>Proof of optimality:</u> Let C be the set of courses, $S_{ALG} \subseteq C$ be the greedy algorithm's selection, and $S_{OPT} \subseteq C$ be an optimal selection, all sorted by increasing finish time.

Suppose $S_{ALG}[i] = S_{OPT}[i]$, for all $i < k$ and $S_{ALG}[k] = c_i \neq c_j = S_{OPT}[k]$.

Create the revised schedule $S'_{OPT} = S_{OPT} \setminus \{c_j\} \cup \{c_i\}$. (I.e., Swap $S_{ALG}[k]$ for $S_{OPT}[k]$)

Will S'_{OPT} be valid?



Single Room Scheduling

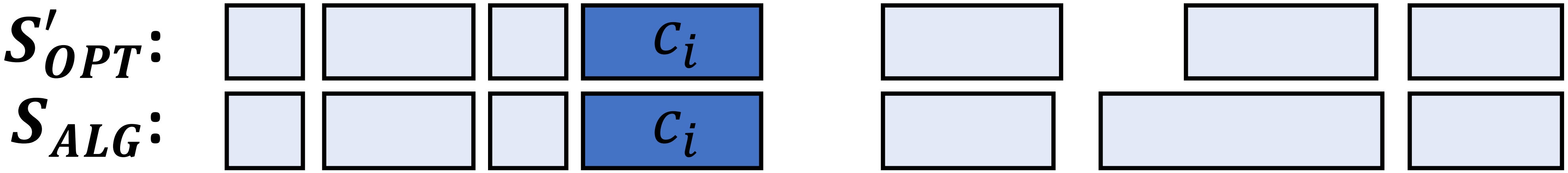
<u>Greedy decision:</u> Select the next course with the earliest compatible finish time.

<u>Proof of optimality:</u> Let C be the set of courses, $S_{ALG} \subseteq C$ be the greedy algorithm's selection, and $S_{OPT} \subseteq C$ be an optimal selection, all sorted by increasing finish time.

Suppose $S_{ALG}[i] = S_{OPT}[i]$, for all $i < k$ and $S_{ALG}[k] = c_i \neq c_j = S_{OPT}[k]$.

Create the revised schedule $S'_{OPT} = S_{OPT} \setminus \{c_j\} \cup \{c_i\}$. (I.e., Swap $S_{ALG}[k]$ for $S_{OPT}[k]$)

Will S'_{OPT} be valid?
Need to check and see if c_j messed up any compatibilities.



Single Room Scheduling

<u>Greedy decision:</u> Select the next course with the earliest compatible finish time.

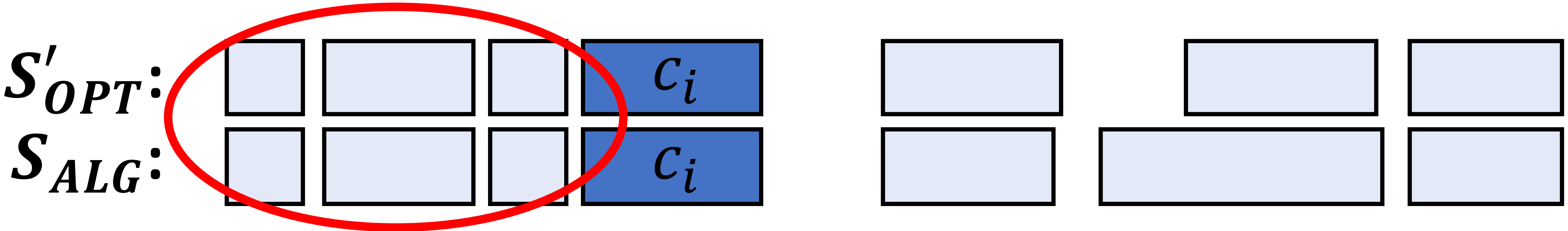
<u>Proof of optimality:</u> Let C be the set of courses, $S_{ALG} \subseteq C$ be the greedy algorithm's selection, and $S_{OPT} \subseteq C$ be an optimal selection, all sorted by increasing finish time.

Suppose $S_{ALG}[i] = S_{OPT}[i]$, for all $i < k$ and $S_{ALG}[k] = c_i \neq c_j = S_{OPT}[k]$.

Create the revised schedule $S'_{OPT} = S_{OPT} \setminus \{c_j\} \cup \{c_i\}$. (I.e., Swap $S_{ALG}[k]$ for $S_{OPT}[k]$)

c_i is compatible with previous courses in S'_{OPT} since $S_{ALG}[i] = S_{OPT}[i] = S'_{OPT}[i]$, for all $i < k$

c_i is compatible with S_{ALG}, and S_{OPT} and S'_{OPT} share the same courses before c_i.



Single Room Scheduling

Greedy decision: Select the next course with the earliest compatible finish time.

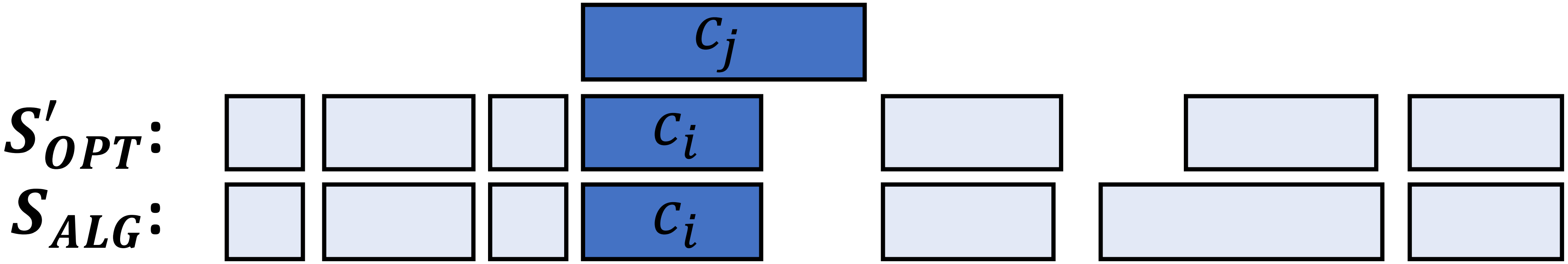
Proof of optimality: Let C be the set of courses, $S_{ALG} \subseteq C$ be the greedy algorithm's selection, and $S_{OPT} \subseteq C$ be an optimal selection, all sorted by increasing finish time.

Suppose $S_{ALG}[i] = S_{OPT}[i]$, for all $i < k$ and $S_{ALG}[k] = c_i \neq c_j = S_{OPT}[k]$.

Create the revised schedule $S'_{OPT} = S_{OPT} \setminus \{c_j\} \cup \{c_i\}$. (I.e., Swap $S_{ALG}[k]$ for $S_{OPT}[k]$)

c_i is compatible with previous courses in S'_{OPT} since $S_{ALG}[i] = S_{OPT}[i] = S'_{OPT}[i]$, for all $i < k$

c_i is compatible with subsequent courses in S'_{OPT} since $f_i \leq f_j$. Otherwise, the greedy algorithm would have selected c_j instead of c_i.



Single Room Scheduling

<u>Greedy decision:</u> Select the next course with the earliest compatible finish time.

<u>Proof of optimality:</u> Let C be the set of courses, $S_{ALG} \subseteq C$ be the greedy algorithm's selection, and $S_{OPT} \subseteq C$ be an optimal selection, all sorted by increasing finish time.

Suppose $S_{ALG}[i] = S_{OPT}[i]$, for all $i < k$ and $S_{ALG}[k] = c_i \neq c_j = S_{OPT}[k]$.

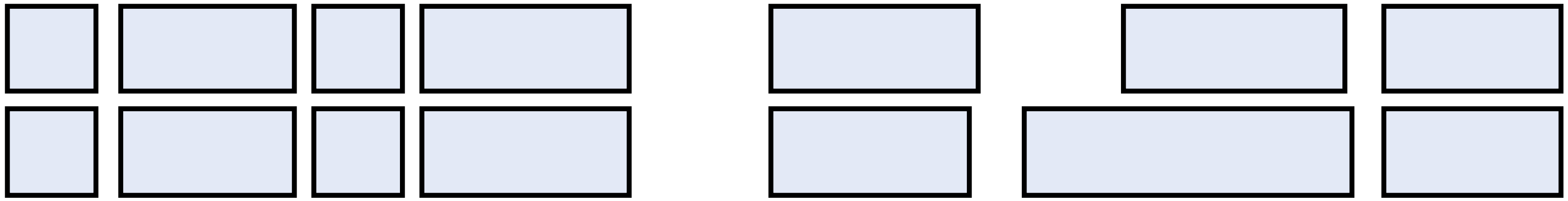
Create the revised schedule $S'_{OPT} = S_{OPT} \setminus \{c_j\} \cup \{c_i\}$. (I.e., Swap $S_{ALG}[k]$ for $S_{OPT}[k]$)

c_i is compatible with previous courses in S'_{OPT} since $S_{ALG}[i] = S_{OPT}[i] = S'_{OPT}[i]$, for all $i < k$

c_i is compatible with subsequent courses in S'_{OPT} since $f_i \leq f_j$. Otherwise, the greedy algorithm would have selected c_j instead of c_i.

So S'_{OPT} is a valid schedule with the same number of courses as S_{OPT}, so S'_{OPT} is also optimal.

$\boldsymbol{S'_{OPT}}$:
$\boldsymbol{S_{ALG}}$:



Single Room Scheduling

<u>Greedy decision:</u> Select the next course with the earliest compatible finish time.

<u>Proof of optimality:</u> Let C be the set of courses, $S_{ALG} \subseteq C$ be the greedy algorithm's selection, and $S_{OPT} \subseteq C$ be an optimal selection, all sorted by increasing finish time.

Suppose $S_{ALG}[i] = S_{OPT}[i]$, for all $i < k$ and $S_{ALG}[k] = c_i \neq c_j = S_{OPT}[k]$.

Create the revised schedule $S'_{OPT} = S_{OPT} \setminus \{c_j\} \cup \{c_i\}$. (I.e., Swap $S_{ALG}[k]$ for $S_{OPT}[k]$)

c_i is compatible with previous courses in S'_{OPT} since $S_{ALG}[i] = S_{OPT}[i] = S'_{OPT}[i]$, for all $i < k$

c_i is compatible with subsequent courses in S'_{OPT} since $f_i \leq f_j$. Otherwise, the greedy algorithm would have selected c_j instead of c_i.

So S'_{OPT} is a valid schedule with the same number of courses as S_{OPT}, so S'_{OPT} is also optimal.

We can then proceed inductively and show that each course in S_{OPT} can be replaced by the corresponding course in S_{ALG} without violating compatibility.

Single Room Scheduling

<u>Greedy decision:</u> Select the next course with the earliest compatible finish time.

<u>Proof of optimality:</u> Let C be the set of courses, $S_{ALG} \subseteq C$ be the greedy algorithm's selection, and $S_{OPT} \subseteq C$ be an optimal selection, all sorted by increasing finish time.

Suppose $S_{ALG}[i] = S_{OPT}[i]$, for all $i < k$ and $S_{ALG}[k] = c_i \neq c_j = S_{OPT}[k]$.

Create the revised schedule $S'_{OPT} = S_{OPT} \setminus \{c_j\} \cup \{c_i\}$. (I.e., Swap $S_{ALG}[k]$ for $S_{OPT}[k]$)

c_i is compatible with previous courses in S'_{OPT} since $S_{ALG}[i] = S_{OPT}[i] = S'_{OPT}[i]$, for all $i < k$

c_i is compatible with subsequent courses in S'_{OPT} since $f_i \leq f_j$. Otherwise, the greedy algorithm would have selected c_j instead of c_i.

So S'_{OPT} is a valid schedule with the same number of courses as S_{OPT}, so S'_{OPT} is also optimal.

We can then proceed inductively and show that each course in S_{OPT} can be replaced by the corresponding course in S_{ALG} without violating compatibility. Since replacing every course in S_{OPT} with the courses in S_{ALG} keeps the solution optimal, S_{ALG} must be optimal. (i.e., we translated S_{OPT} into S_{ALG} at no extra cost).

Single Room Scheduling

Input:

- $C = \{c_1, c_2, \dots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of compatible courses.

Room Minimization

Input:

- $C = \{c_1, c_2, \dots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Compatibly schedule all courses with the min number of rooms.

Room Minimization

Input:

- $C = \{c_1, c_2, \ldots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Compatibly schedule all courses with the min number of rooms.

Algorithm Idea?

Room Minimization

Input:

- $C = \{c_1, c_2, \ldots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

Rules:

- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Compatibly schedule all courses with the min number of rooms.

Algorithm Idea?

Assign as much as possible to room 1,
then as much as possible to room 2,…

Optimal?