Workshop #1
CSCI 432

Single Room Scheduling

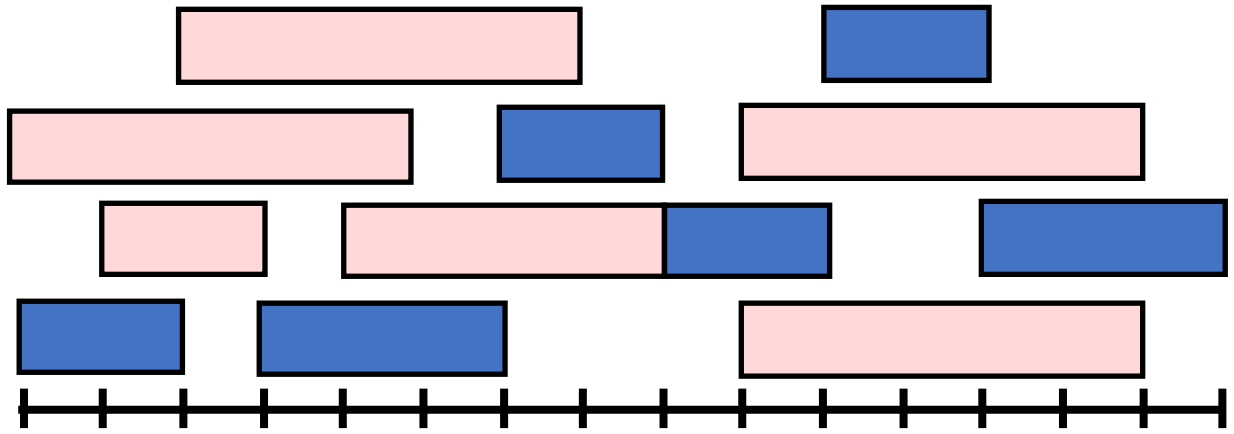
Input:

- $C = \{c_1, c_2, \ldots, c_n\}$ – 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

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 $S_{ALG}[k] = c_i \neq c_j = S_{OPT}[k]$.

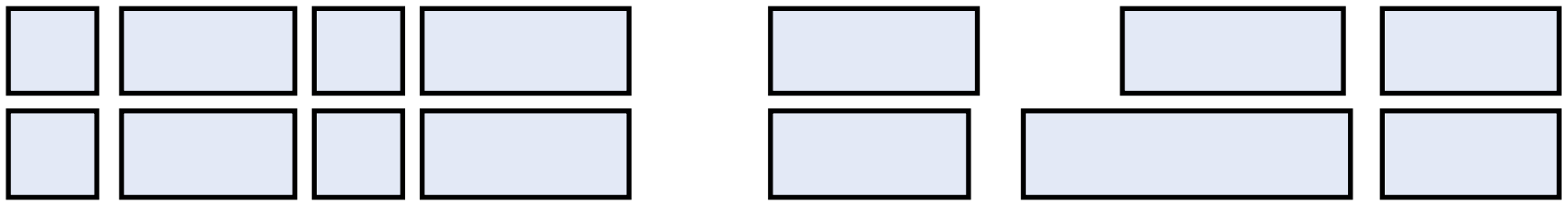
Create the revised schedule $S'_{OPT} = S_{OPT} \setminus \{c_j\} \cup \{c_i\}$. (I.e., Swap $S_{ALG}[k]$ for $S_{OPT}[k]$)

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.

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

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



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.

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 many as possible to room 1,
then as many as possible to room 2,…

Optimal?

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:

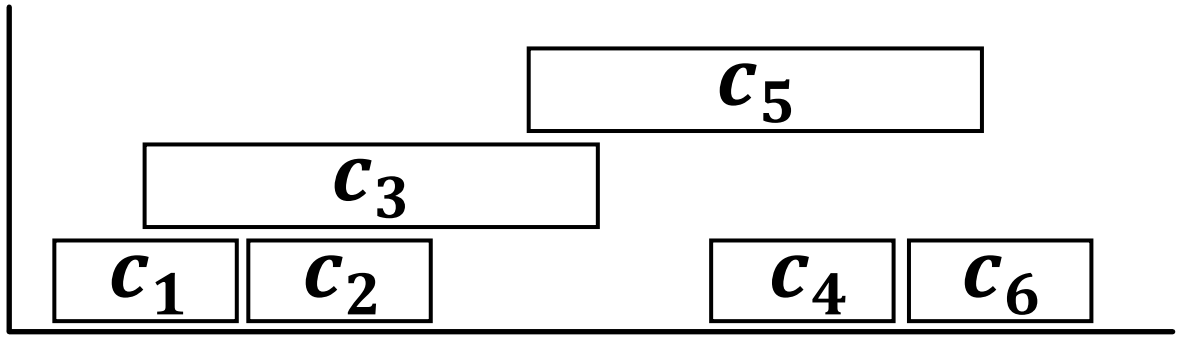
- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

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

Algorithm Idea?

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

Optimal? No!



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?

Occupied Rooms

Unoccupied Rooms ← But previously occupied

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.

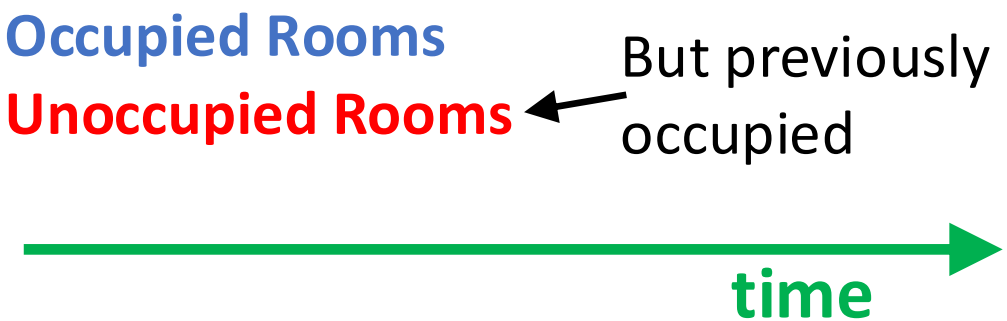
Rules:

- 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?

At each time:



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:

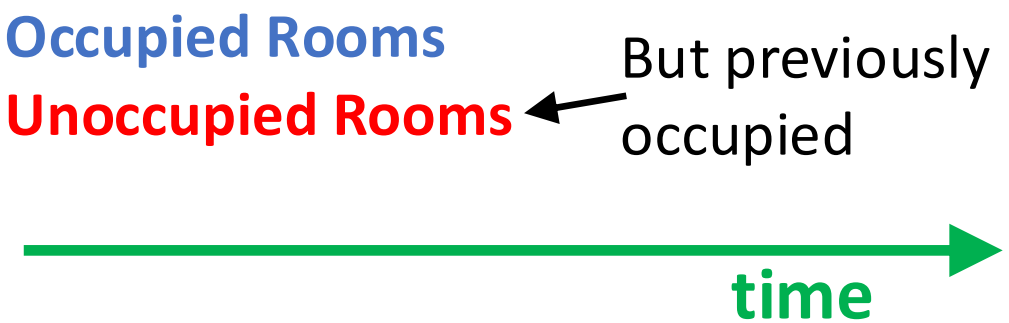
- c_i and c_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

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

Algorithm Idea?

At each time:

1. When course ends, move its **occupied room** to **unoccupied rooms**.



Room Minimization

Input:

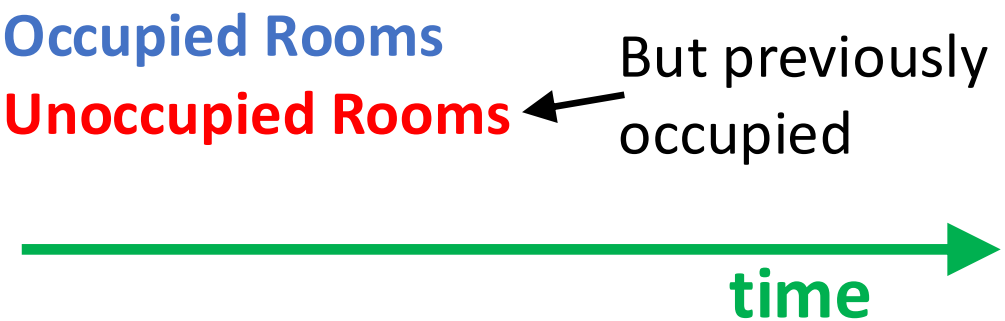
- $C = \{c_1, c_2, \ldots, c_n\}$ – set of courses that need rooms.
- $c_i = [s_i, f_i)$ – start and finish times for each course.

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?



At each time:

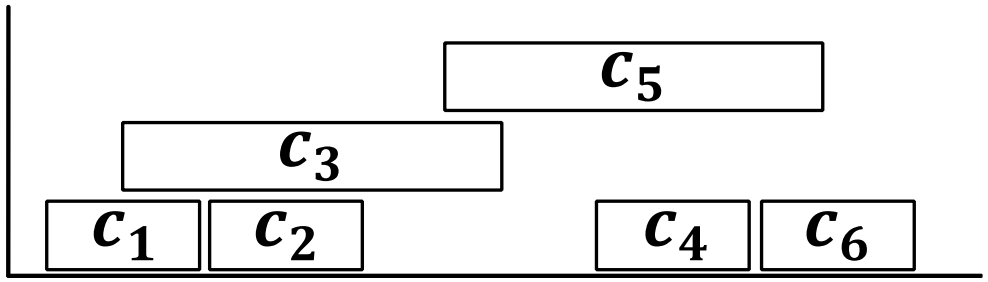
1. When course ends, move its **occupied room** to **unoccupied rooms**.
2. When course starts, **select unoccupied room**. If none exists, select a new (never used) room.

```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```

F = Unoccupied Rooms

B = Occupied Rooms

S = course -> room Map

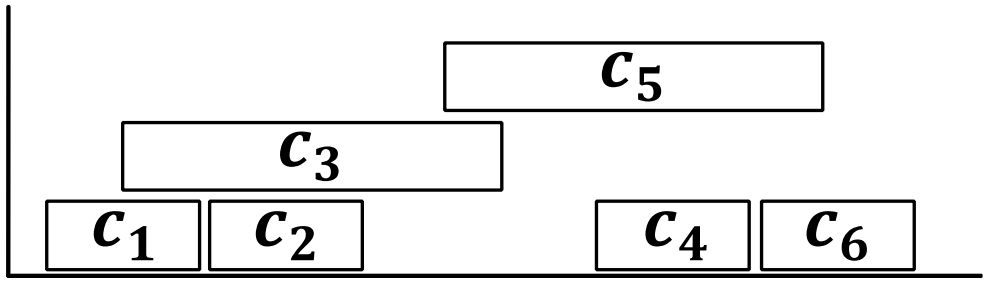


```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```

F = Unoccupied Rooms

B = Occupied Rooms

S = course -> room Map



$$F = \{\}$$
$$B = \{\}$$
$$S = \{\}$$

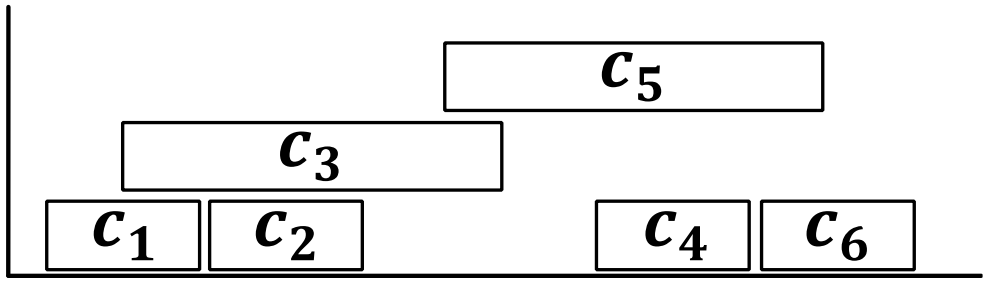
```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```

Go through one timeslot at a time

F = Unoccupied Rooms

B = Occupied Rooms

S = course -> room Map



$F = \{\}$

$B = \{\}$

$S = \{\}$

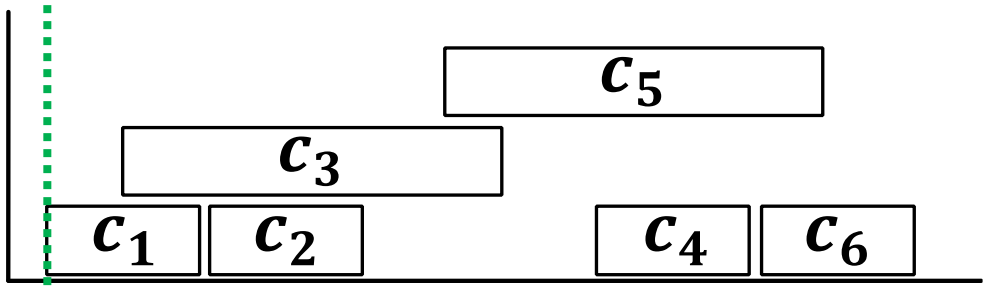
```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```

Go through one timeslot at a time

F = Unoccupied Rooms

B = Occupied Rooms

S = course -> room Map



$F = \{\}$

$B = \{\}$

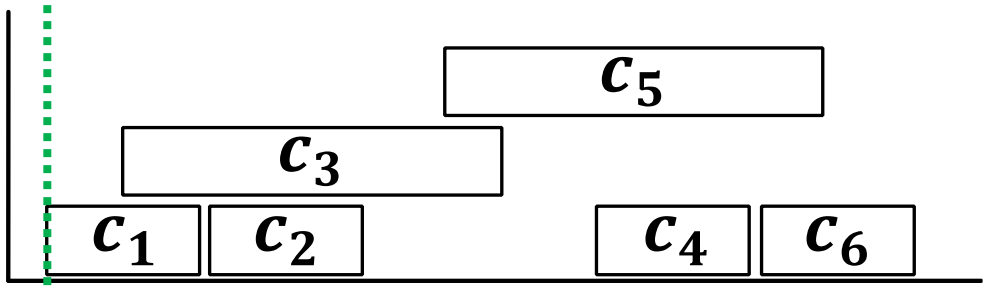
$S = \{\}$

```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```

F = Unoccupied Rooms

B = Occupied Rooms

S = course -> room Map



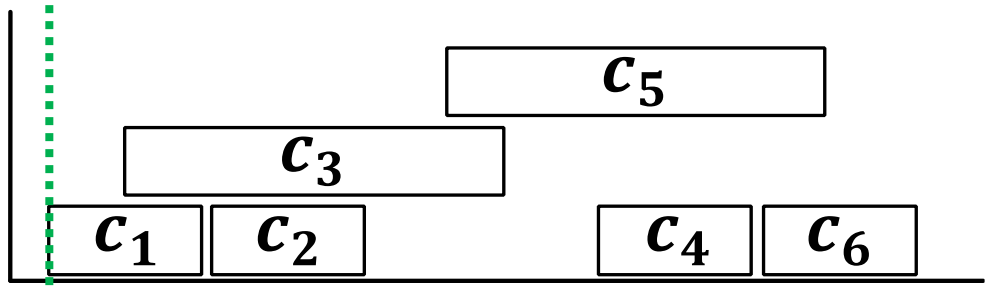
$$F = \{\}$$
$$B = \{\}$$
$$S = \{\}$$

```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```

F = Unoccupied Rooms

B = Occupied Rooms

S = course -> room Map



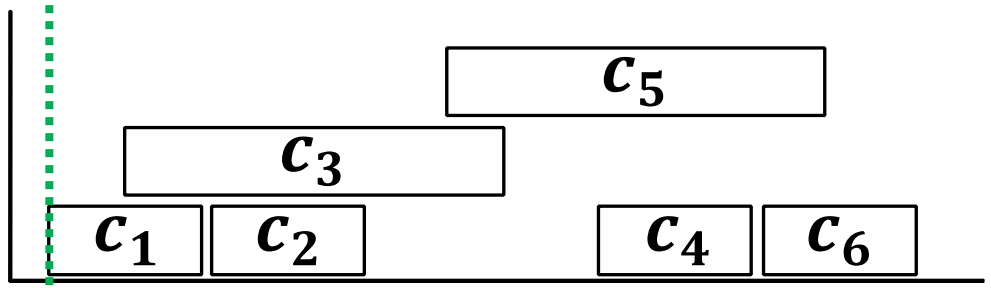
$$F = \{\}$$
$$B = \{\}$$
$$S = \{\}$$

```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```

F = Unoccupied Rooms

B = Occupied Rooms

S = course -> room Map



$F = \{1\}$

$B = \{\}$

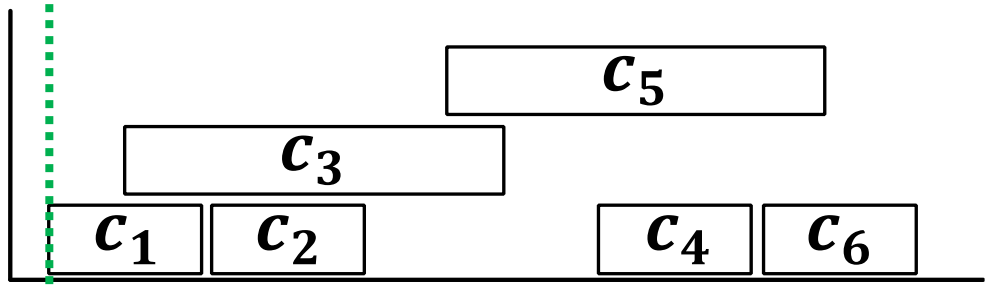
$S = \{\}$

```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```

F = Unoccupied Rooms

B = Occupied Rooms

S = course -> room Map



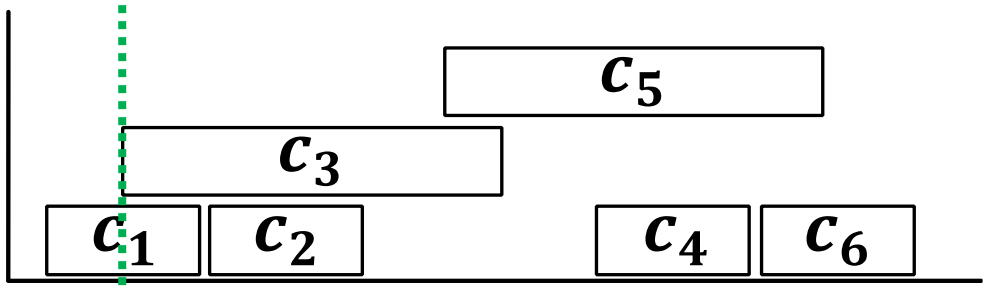
$$F = \{\}$$
$$B = \{1\}$$
$$S = \{c_1 \rightarrow 1\}$$

```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```

F = Unoccupied Rooms

B = Occupied Rooms

S = course -> room Map



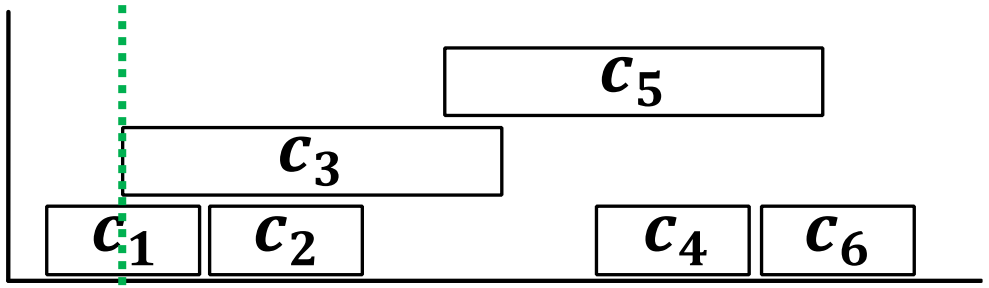
$$F = \{\}$$
$$B = \{1\}$$
$$S = \{c_1 \rightarrow 1\}$$

```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```

F = Unoccupied Rooms

B = Occupied Rooms

S = course -> room Map



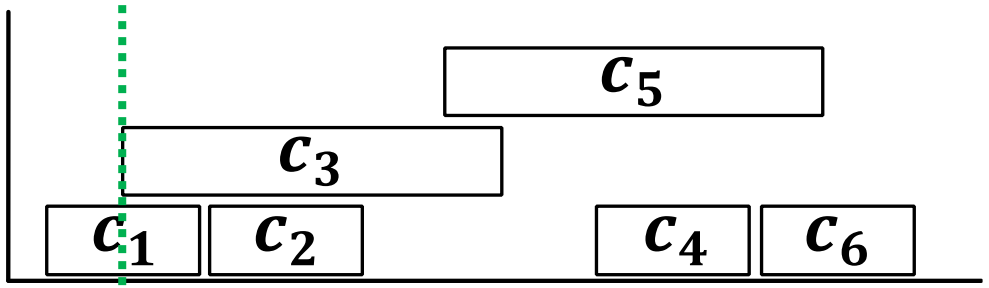
$$F = \{\}$$
$$B = \{1\}$$
$$S = \{c_1 \rightarrow 1\}$$

```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```

F = Unoccupied Rooms

B = Occupied Rooms

S = course -> room Map



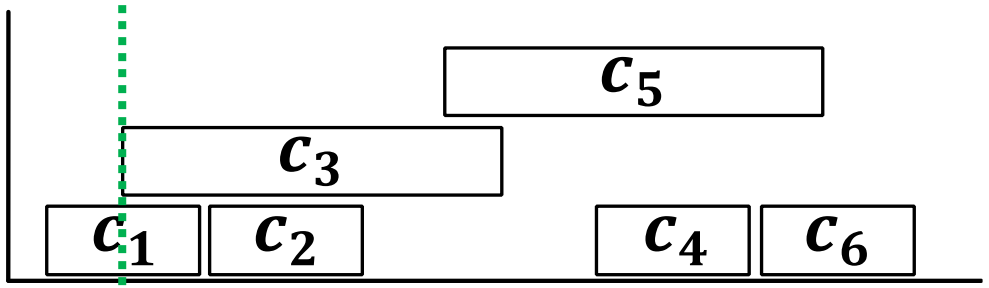
$$F = \{\}$$
$$B = \{1\}$$
$$S = \{c_1 \rightarrow 1\}$$

```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```

F = Unoccupied Rooms

B = Occupied Rooms

S = course -> room Map



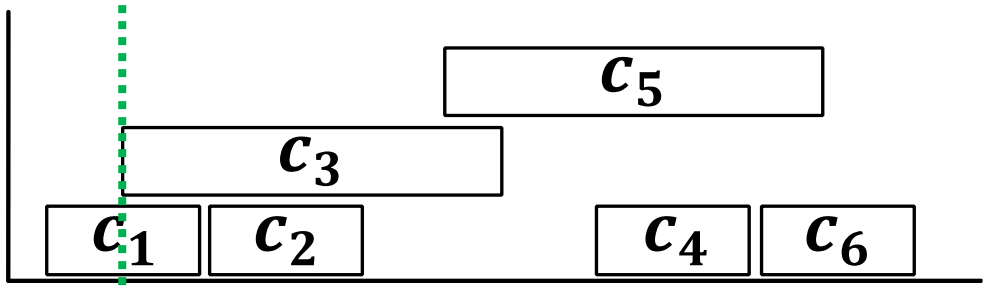
$$F = \{2\}$$
$$B = \{1\}$$
$$S = \{c_1 \rightarrow 1\}$$

```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```

F = Unoccupied Rooms

B = Occupied Rooms

S = course -> room Map



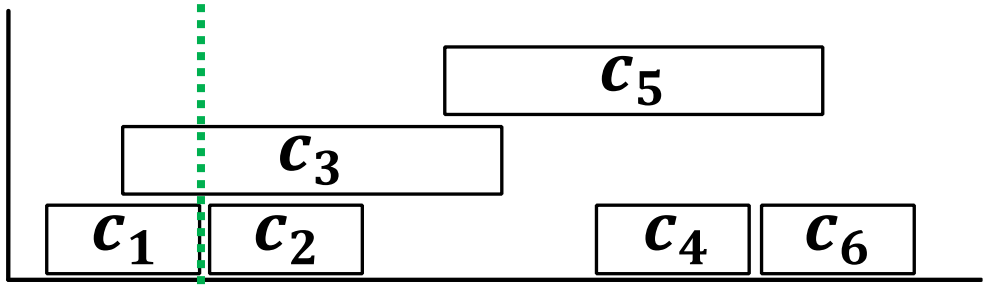
$$F = \{\}$$
$$B = \{1,2\}$$
$$S = \{c_1 \rightarrow 1, c_3 \rightarrow 2\}$$

```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```

F = Unoccupied Rooms

B = Occupied Rooms

S = course -> room Map



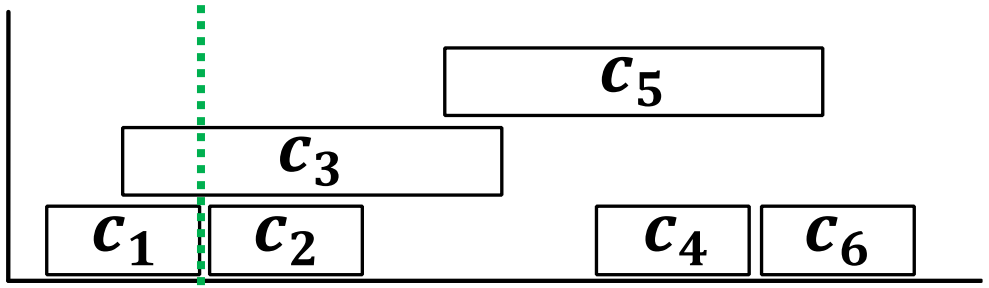
$$F = \{\}$$
$$B = \{1,2\}$$
$$S = \{c_1 \rightarrow 1, c_3 \rightarrow 2\}$$

```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```

F = Unoccupied Rooms

B = Occupied Rooms

S = course -> room Map



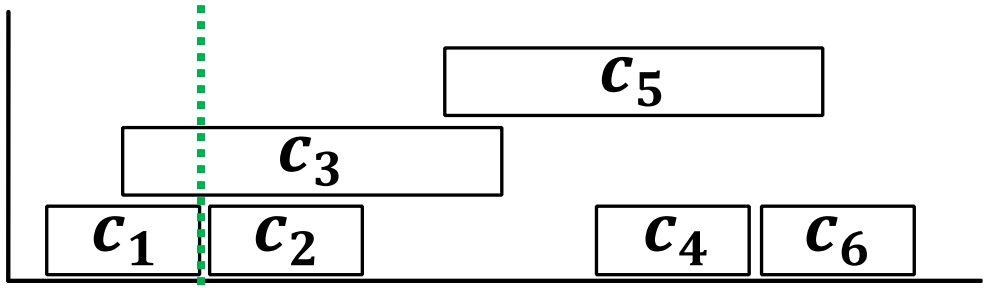
$$F = \{\}$$
$$B = \{1,2\}$$
$$S = \{c_1 \rightarrow 1, c_3 \rightarrow 2\}$$

```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```

F = Unoccupied Rooms

B = Occupied Rooms

S = course -> room Map



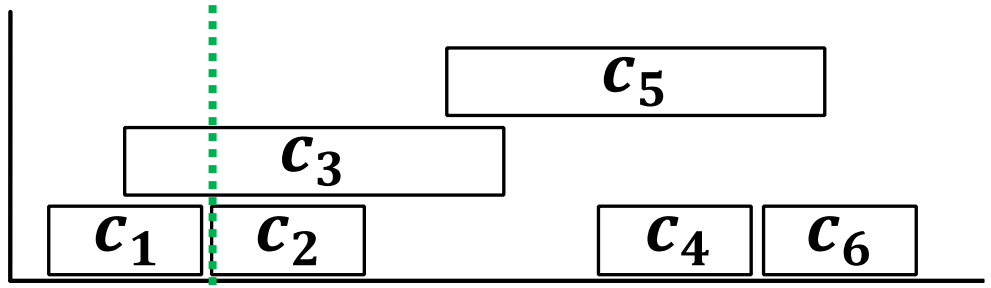
$$F = \{1\}$$
$$B = \{2\}$$
$$S = \{c_1 \rightarrow 1, c_3 \rightarrow 2\}$$

```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```

F = Unoccupied Rooms

B = Occupied Rooms

S = course -> room Map



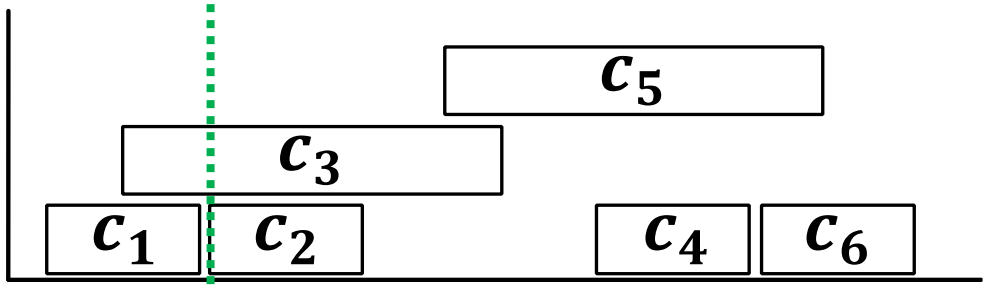
$$F = \{1\}$$
$$B = \{2\}$$
$$S = \{c_1 \rightarrow 1, c_3 \rightarrow 2\}$$

```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```

F = Unoccupied Rooms

B = Occupied Rooms

S = course -> room Map



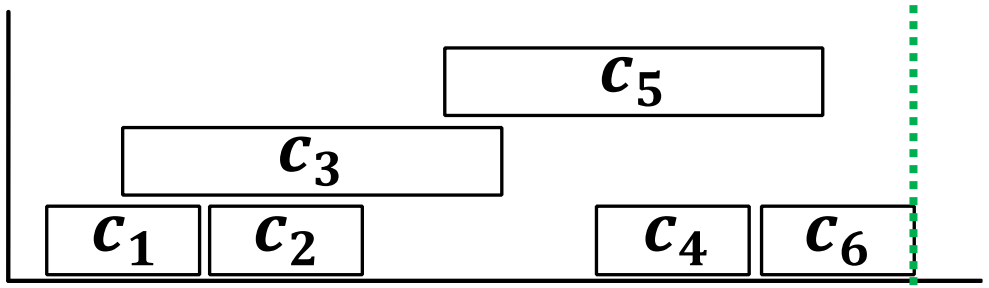
$$F = \{\}$$
$$B = \{2,1\}$$
$$S = \{c_1 \to 1, c_3 \to 2, c_2 \to 1\}$$

```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```

F = Unoccupied Rooms

B = Occupied Rooms

S = course -> room Map



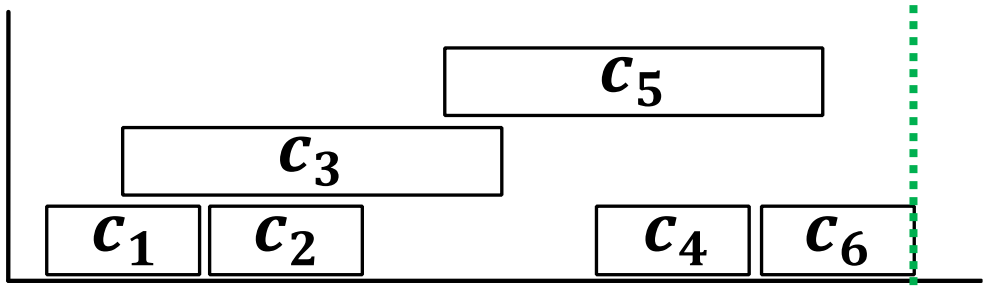
$F = \{1,2\}$

$B = \{\}$

$S = \{c_1 \to 1, c_3 \to 2, c_2 \to 1, \dots\}$

Optimal?

```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```



$F = \{1,2\}$

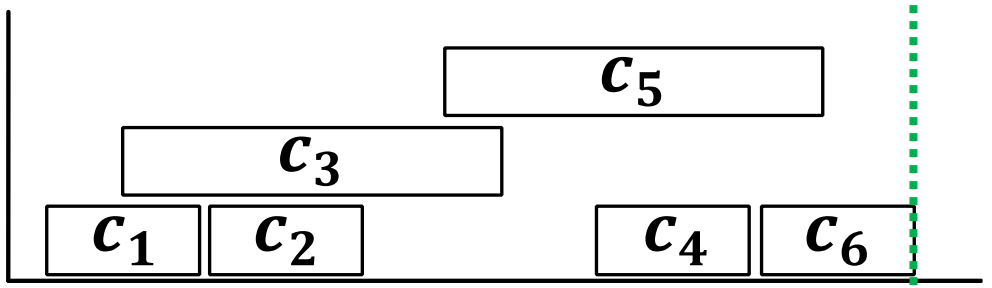
$B = \{\}$

$S = \{c_1 \to 1, c_3 \to 2, c_2 \to 1, \ldots\}$

Optimal?

```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```

If we identify a lower bound intrinsic to the problem (i.e., can't do is with fewer than X rooms), and our algorithm does it with X rooms, our algorithm is optimal.



$F = \{1,2\}$

$B = \{\}$

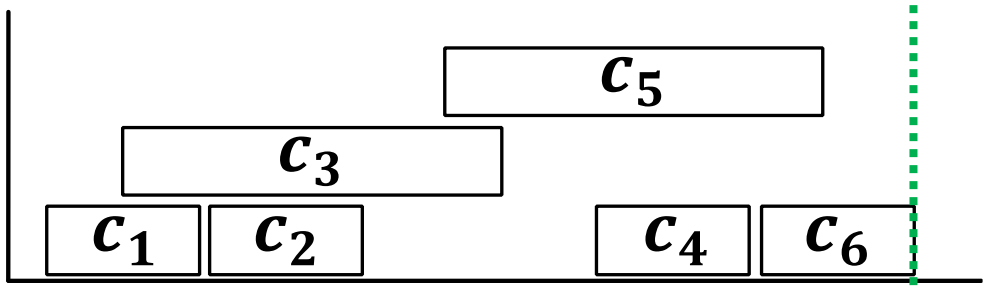
$S = \{c_1 \to 1, c_3 \to 2, c_2 \to 1, \ldots\}$

```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```

Optimal?

What is the smallest possible number of rooms?

If we identify a lower bound intrinsic to the problem (i.e., can't do is with fewer than X rooms), and our algorithm does it with X rooms, our algorithm is optimal.



$F = \{1,2\}$

$B = \{\}$

$S = \{c_1 \to 1, c_3 \to 2, c_2 \to 1, \ldots\}$

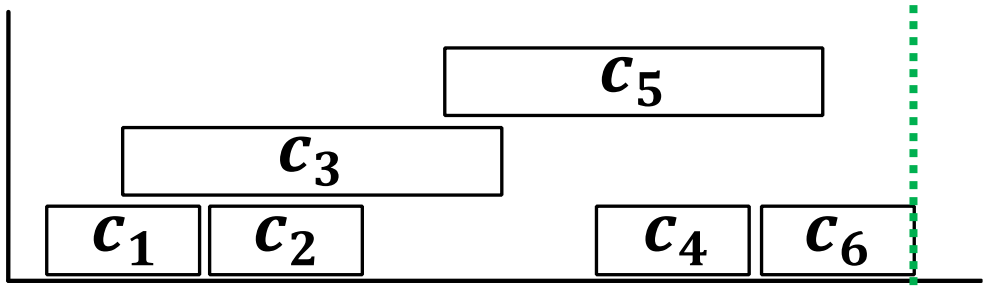
```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```

Optimal?

What is the smallest possible number of rooms?

The number of concurrent courses.

If we identify a lower bound intrinsic to the problem (i.e., can't do is with fewer than X rooms), and our algorithm does it with X rooms, our algorithm is optimal.



$F = \{1,2\}$

$B = \{\}$

$S = \{c_1 \to 1, c_3 \to 2, c_2 \to 1, \ldots\}$

```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```

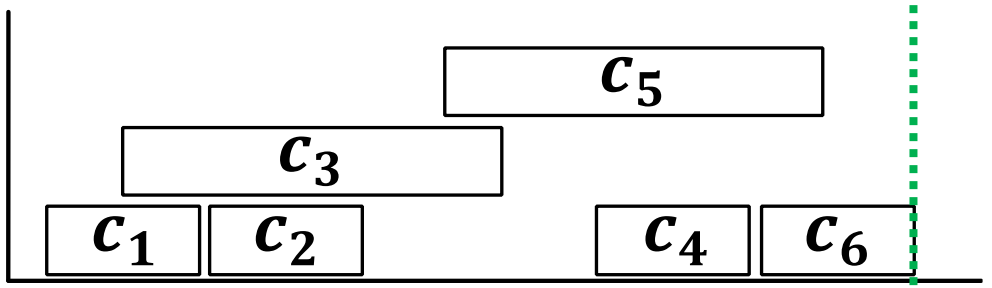
Optimal?

rooms needed ≥ # concurrent courses.

New room added ⇔ all other rooms in use

(i.e., # rooms used = max[# concurrent courses])

So, yes.



$F = \{1,2\}$

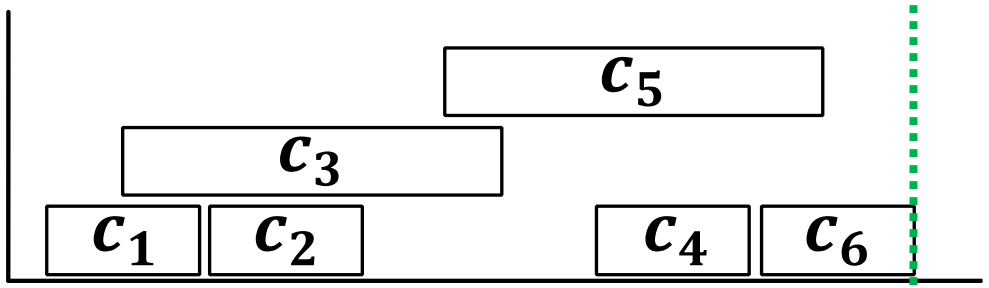
$B = \{\}$

$S = \{c_1 \to 1, c_3 \to 2, c_2 \to 1, \dots\}$

```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```

Running Time?

Valid?



$F = \{1,2\}$

$B = \{\}$

$S = \{c_1 \to 1, c_3 \to 2, c_2 \to 1, \ldots\}$

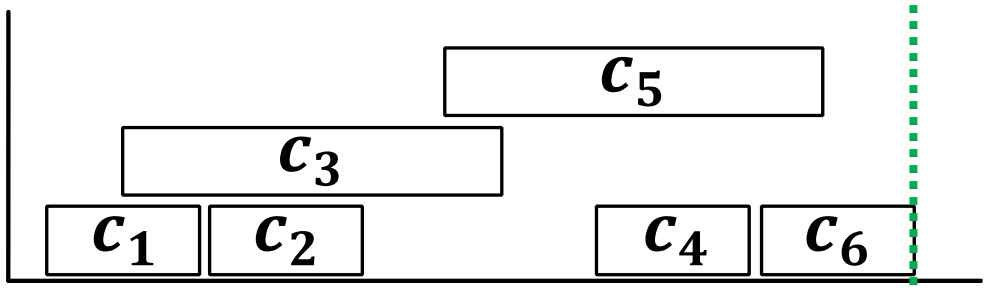
```
room_minimization(courses C)
    F = B = S = ∅
    room_num = 0
    foreach timeslot t
        foreach c in C
            if c.finish == t
                F.add(B.getBookedRoom(S.getroom(c)))
        foreach c in C
            if c.start == t
                if F.isEmpty()
                    room_num += 1
                    F.add(room_num)
                room = F.getFreeRoom()
                S.schedule(c, room)
                B.add(room)
    return S
```

Running Time?

$O(|C|^2)$ – Don't need to go through all timeslots. Just start/finish for each activity.

Valid?

Yes. Each course is given a room and no rooms are concurrently occupied.



$F = \{1,2\}$

$B = \{\}$

$S = \{c_1 \to 1, c_3 \to 2, c_2 \to 1, \dots\}$