Tutorial (Even More) Greedy

TA: Anna Lindeberg

September 30, 2024

A scheduling problem

This week, the department finalizes the booking request for the fall term 2025. Let's help out!

A scheduling problem

This week, the department finalizes the booking request for the fall term 2025. Let's help out!

Input: An array S containing n tuples (s_i, e_i) specifying a start time s_i and an end time e_i of n different classes.

Question: What is the minimum number of rooms needed so that there is no overlapping classes in any room?

Example

Say

$$S = [\underbrace{(10^{00}, 11^{45})}_{c_1}, \underbrace{(14^{00}, 15^{30})}_{c_2}, \underbrace{(8^{30}, 14^{00})}_{c_3}, \underbrace{(13^{00}, 15^{00})}_{c_4}, \underbrace{(8^{00}, 9^{45})}_{c_5}, \underbrace{(8^{00}, 16^{00})}_{c_6}, \underbrace{(15^{30}, 17^{00})}_{c_7}]$$

where $c_i = (s_i, e_i)$

Greedy algorithm

minRooms(S)

1: Sort S w.r.t. first element of each tuples
2: R $\leftarrow$ [0] ▷ R[i] stores current end time for last class in room i

Greedy algorithm

minRooms(S)

1: Sort S w.r.t. first element of each tuples
2: R $\leftarrow$ [0] ▷ R[i] stores current end time for last class in room i
3: **for** each (s_i, e_i) in S in order **do**

14: **end for**

Greedy algorithm

minRooms(S)

1: Sort S w.r.t. first element of each tuples
2: R $\leftarrow$ [0] ▷ R[i] stores current end time for last class in room i
3: **for** each (s_i, e_i) in S in order **do**
4: placed $\leftarrow$ `false`

14: **end for**

Greedy algorithm

minRooms(S)

```
1:  Sort S w.r.t. first element of each tuples
2:  R ← [0]                          ▷ R[i] stores current end time for last class in room i
3:  for each (s_i, e_i) in S in order do
4:      placed ← false
5:      for j ← 1, 2, ..., |R| do
6:          if R[j] ≤ s_i and not placed then
7:              R[j] ← e_i
8:              placed ← true
9:          end if
10:     end for

14: end for
```

Greedy algorithm

minRooms(S)

```
1:  Sort S w.r.t. first element of each tuples
2:  R ← [0]                          ▷ R[i] stores current end time for last class in room i
3:  for each (s_i, e_i) in S in order do
4:      placed ← false
5:      for j ← 1, 2, ..., |R| do
6:          if R[j] ≤ s_i and not placed then
7:              R[j] ← e_i
8:              placed ← true
9:          end if
10:     end for
11:     if not placed then
12:         Append e_i to R
13:     end if
14: end for
```

Let us prove:

Proposition

The algorithm **minRooms(S)** computes the minimum number of rooms needed.