Laboratório de Programação Avançada 2018/19

Week 6 – Greedy Algorithms

UNIVERSIDADE DE COIMBRA

Outline

1. Interval partitioning
2. Heuristics

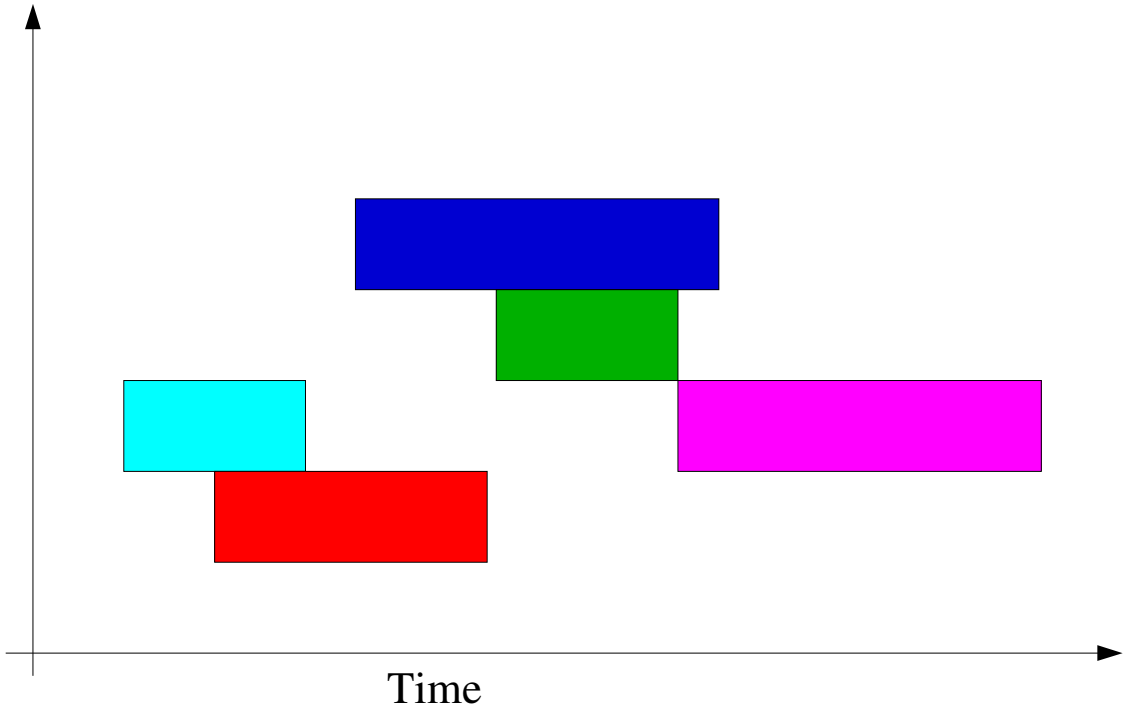
Interval partitioning problem

- A set $S = \{1, \ldots, n\}$ of activities needs a room. Each room can only be used by one activity at a time.
- Each activity i takes place during the interval $[s_i, f_i)$.
- Two activities are compatible if their intervals do not overlap.

Select the minimum number of rooms required to schedule all activities.

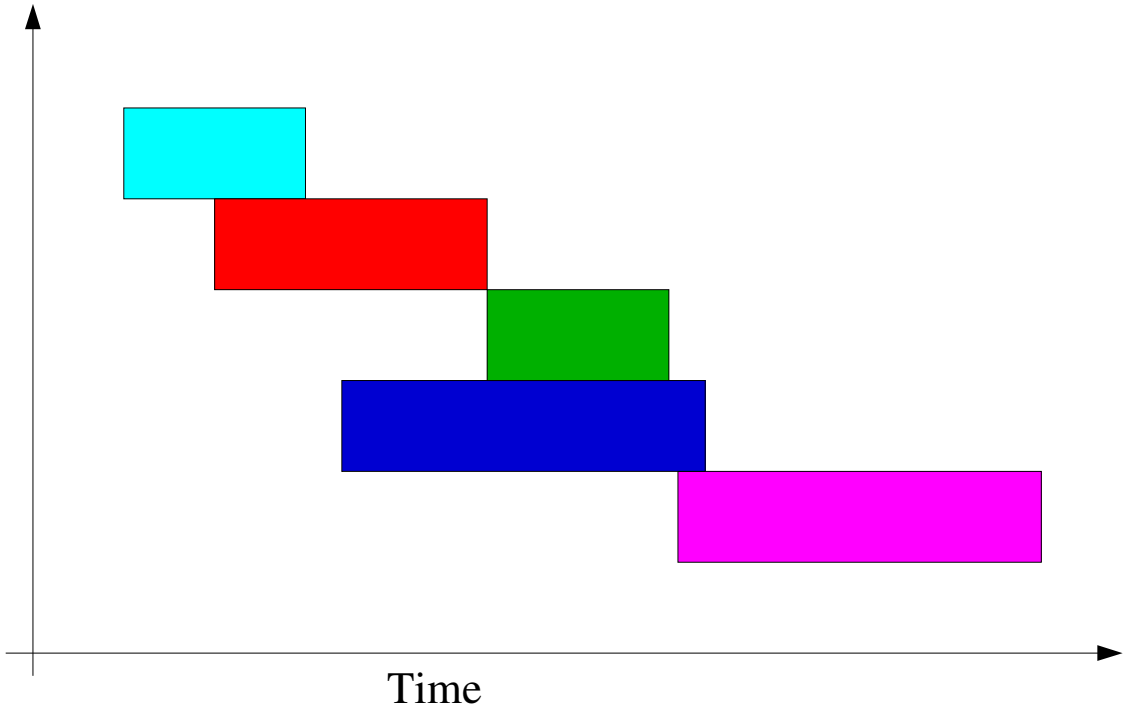
(see Chapter 4.1 of Kleinberg and Tardos, Algorithm Design)

Interval partitioning problem



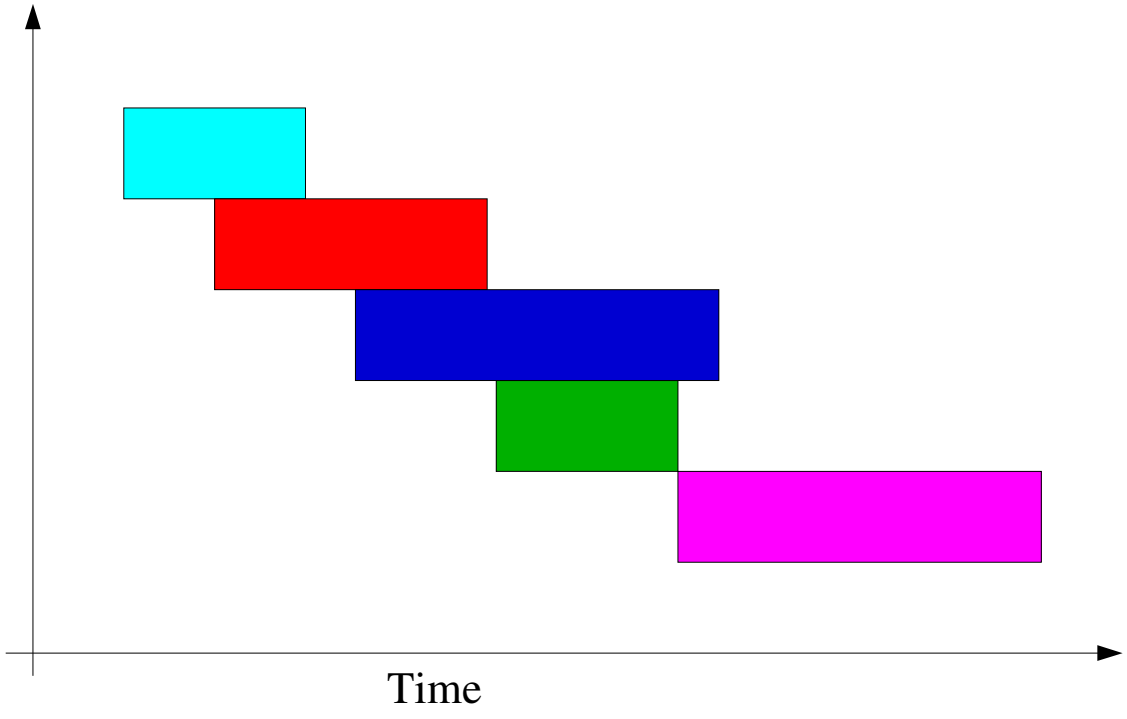
How to solve the problem?

Interval partitioning problem



Sorting in non-decreasing order by finishing times?

Interval partitioning problem



Sorting in non-decreasing order by starting times?

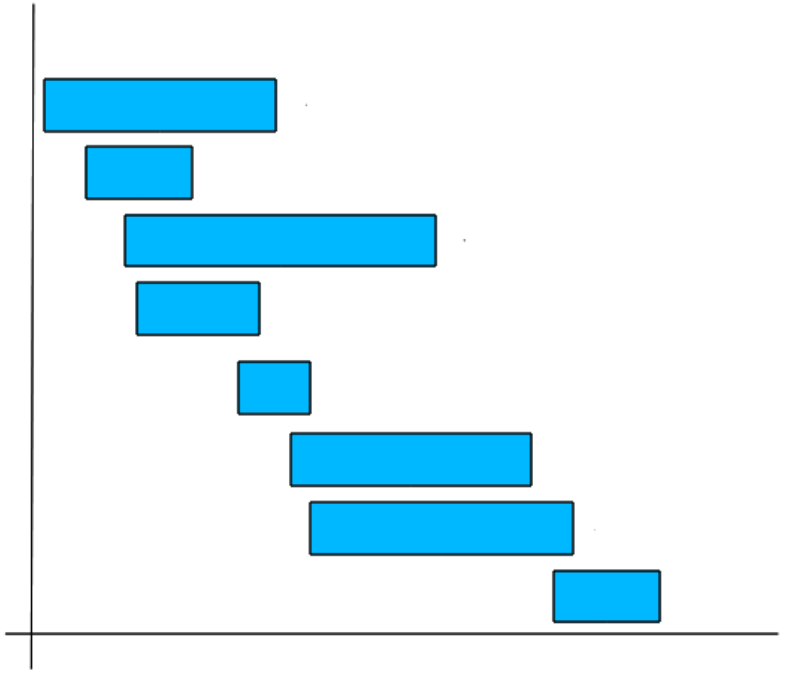
Interval partitioning problem

Function *partition*(S)

sort S into nondecreasing order by starting times s_i
$R_1 = R_2 = \ldots = R_n = \emptyset$ {n rooms available}
$d = 0$ {d is the number of rooms used}
for i in S **do**
  if i can be assigned to some room $k \leq d$ **then**
    $R_k = R_k \cup \{i\}$
  else
    $R_{d+1} = R_{d+1} \cup \{i\}$
    $d = d + 1$
return d

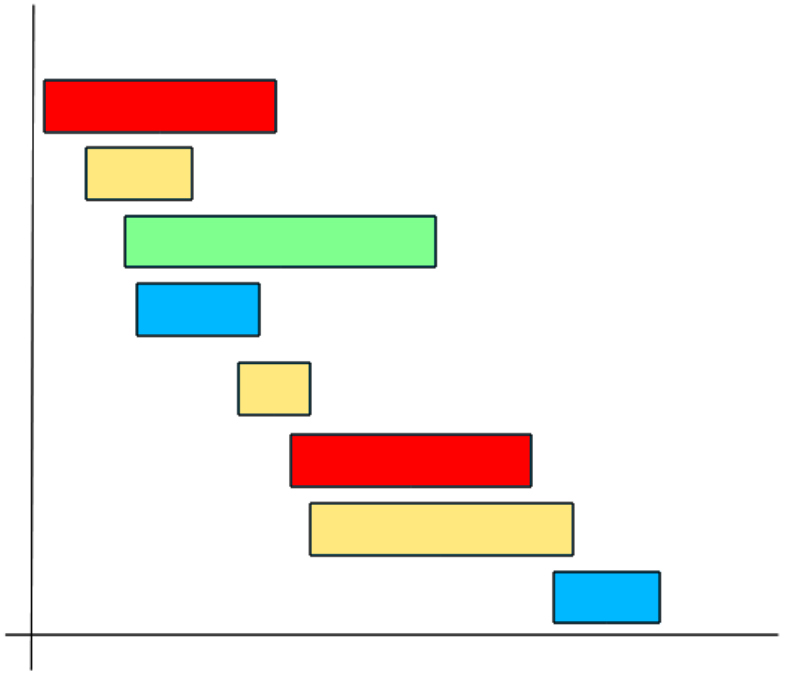
- Greedy choice: choose the next earliest starting time activity
- Why it works?

Interval partitioning problem



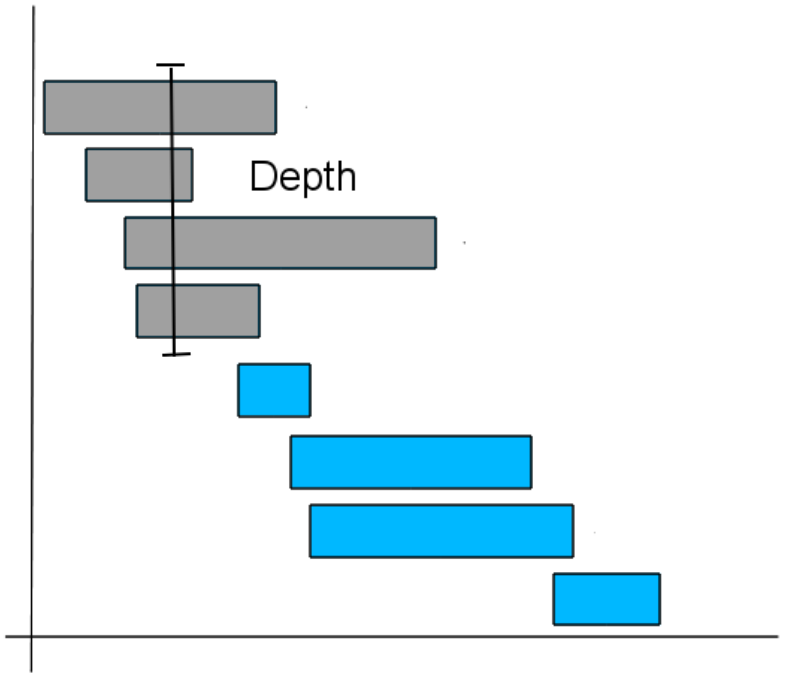
Sorting in non-decreasing order by starting times.

Interval partitioning problem



A solution to the problem.

Interval partitioning problem



Depth d is the maximum overlap of activities that can exist, which is the minimum number of rooms required.

Interval partitioning problem

Claim: The greedy algorithm finds d rooms, where d is the depth

- Assume that the greedy algorithm finds more than d rooms
- Then, at some point in the algorithm, an activity j had requested the $d + 1$-th room.
- This can only happen if there exists d activities that are not compatible among themselves (have requested d rooms) that started before activity j and overlap with its starting time. This contradicts the fact that depth is d.

This is easier to show than by using optimal substructure and greedy choice properties.

Interval partitioning problem

Function *partition*(S)

sort S into nondecreasing order by starting times s_i
$R_1 = R_2 = \ldots = R_n = \emptyset$ {n rooms available}
$d = 0$ {d is the number of rooms used}
for i in S **do**
  if i can be assigned to some room $k \leq d$ **then**
    $R_k = R_k \cup \{i\}$
  else
    $R_{d+1} = R_{d+1} \cup \{i\}$
    $d = d + 1$
return d

Naïve approach in $O(n^2)$ time: search for all available rooms at each step.

Interval partitioning problem

Function *partition*(S)

sort S into nondecreasing order by starting times s_i
$R_1 = R_2 = \ldots = R_n = \emptyset$ {n rooms available}
$d = 0$ {d is the number of rooms used}
for i in S **do**
  if i can be assigned to some room $k \leq d$ **then**
    $R_k = R_k \cup \{i\}$
  else
    $R_{d+1} = R_{d+1} \cup \{i\}$
    $d = d + 1$
return d

$O(n \log n)$ time with a min-priority queue that keeps track of the room that has the activity with the earliest finish time.

Heuristics

Greedy algorithms may not give the optimal solution for some problems, but they may give a good approximation:

- Nearest neighbor for the TSP: Choose the unvisited node that is closest to the last visited node.
- Greedy coloring for the Graph Coloring Problem: Choose a node N and color it with the next available color.
- Greedy algorithm for the Knapsack Problem: Solve the fractional problem and remove the last selected item.