Dynamic Programming

- Weighted Interval Scheduling
- Dynamic Programming Formulation

Imdad ullah Khan

Weighted Interval Scheduling

- You have a mono-task resource

  ▷ e.g. a lecture room or a research equipment

- and multiple requests to use the resource

  Each request specifies a start time and finish time

  Each request has a value (weight)

- Problem is to schedule (accept/reject) the requests

  Selected requests must not overlap in time

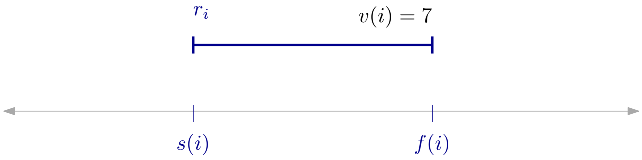
- ~~The goal is to accept maximum number of requests~~
- Goal is to accept requests with maximum total value

Weighted Interval Scheduling

- $\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$ (set of requests)
- Starting and finishing time of r_i: $s(i)$ and $f(i)$

$$\text{for } 1 \leq i \leq n \quad s(i) < f(i)$$

- Duration of request r_i is $d_i = f(i)$ - $s(i)$
- Value of request r_i is $v(i) \geq 0$

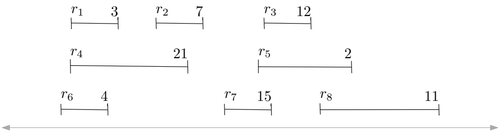


Weighted Interval Scheduling: Compatible Requests

Requests r_i and r_j are **compatible** if they do not overlap in time

▷ Otherwise r_i and r_j are **conflicting**

$$\underbrace{s(i) < f(i) \quad < \quad s(j) < f(j)}_{r_i \text{ is to the left of } r_j} \qquad \text{OR} \qquad \underbrace{s(j) < f(j) \quad < \quad s(i) < f(i)}_{r_i \text{ is to the right of } r_j}$$



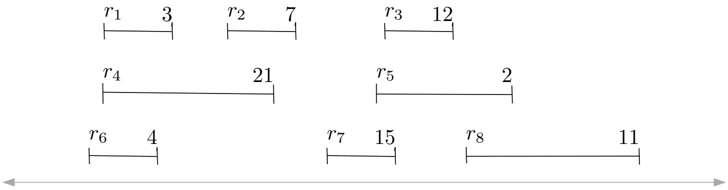
- r_1 and r_2 are compatible, r_4 and r_8 are compatible
- r_1 and r_4 are conflicting, r_5 and r_7 are conflicting

A set is compatible if all pairs in it are compatible

- $\{r_1, r_2, r_8\}$ is compatible
- $\{r_1, r_2, r_5, r_8\}$ is not compatible

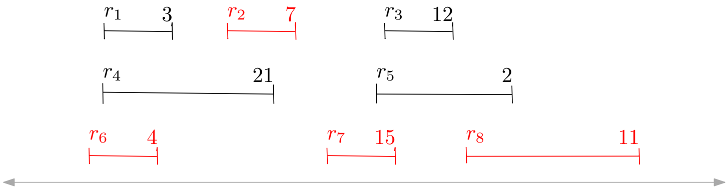
Weighted Interval Scheduling: Problem Formulation

Input: A set $\mathcal{R}$ of requests
Output: A compatible subset $S \subset \mathcal{R}$ of max total weight



Weighted Interval Scheduling: Problem

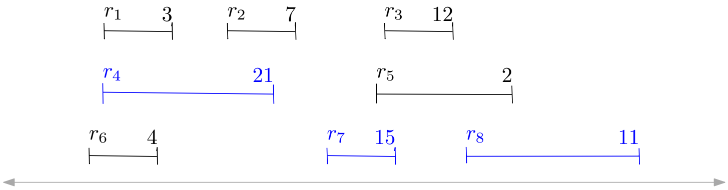
Input: A set $\mathcal{R}$ of requests
Output: A compatible subset $S \subset \mathcal{R}$ of max total weight



A subset with 4 requests and weight 37

Weighted Interval Scheduling: Problem

Input: A set $\mathcal{R}$ of requests
Output: A compatible subset $S \subset \mathcal{R}$ of max total weight



A subset with 3 requests and weight 47

Weighted Interval Scheduling: Problem

Input: A set $\mathcal{R}$ of requests
Output: A compatible subset $S \subset \mathcal{R}$ of max total weight

This is a general version of the (unweighted) interval scheduling

- That is a special case of it

  $\triangleright$ All weights (or values) are equal (to 1)

- Algorithms not working for unweighted case will not work here
- Algorithms working for weighted case will work for unweighted case

Weighted Interval Scheduling: Greedy Algorithms

Input: A set $\mathcal{R}$ of requests
Output: A compatible subset $S \subset \mathcal{R}$ of max total weight

General version of the (unweighted) interval scheduling

The following algorithms will not work

- Earliest starting request first
- Latest finishing request first
- Shortest duration request first
- Least conflicting request first

Why?

The following algorithm may work

- Earliest finishing request first

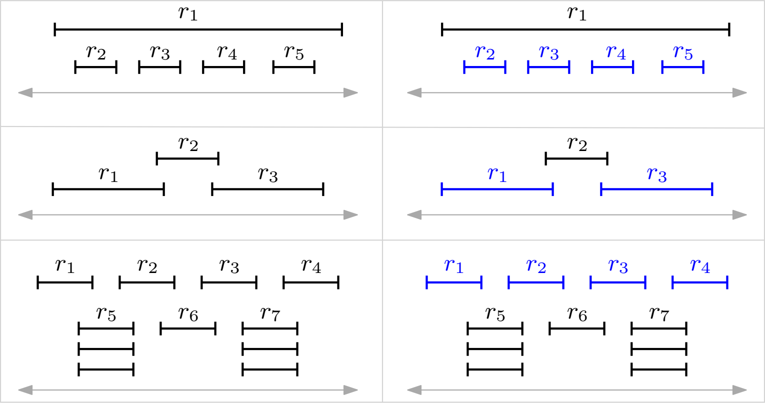
Why?

Weighted Interval Scheduling: Greedy Algorithms

Greedy Algorithm: Earliest finishing request first

▷ Idea is to make resource free as soon as possible

It was optimal for unweighted interval scheduling problem

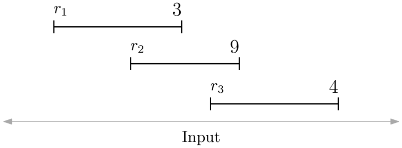


Weighted Interval Scheduling: Greedy Algorithms

Greedy Algorithm: Earliest finishing request first for weighted intervals

▷ Idea is to make resource free as soon as possible

Optimal?



Weighted Interval Scheduling: Greedy Algorithms

Greedy Algorithm: Earliest finishing request first for weighted intervals

▷ Idea is to make resource free as soon as possible

Optimal?