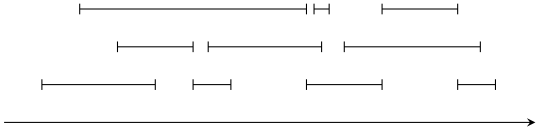
19.6.2
Interval Scheduling: Earliest finish time

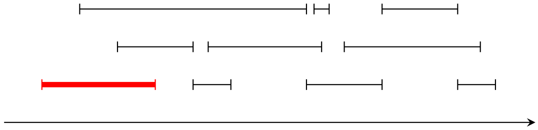
Earliest Finish Time

Process jobs in the order of their finishing times, beginning with those that finish earliest.



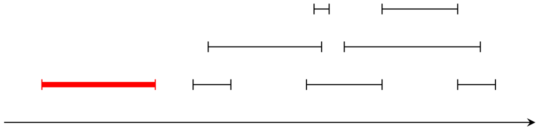
Earliest Finish Time

Process jobs in the order of their finishing times, beginning with those that finish earliest.



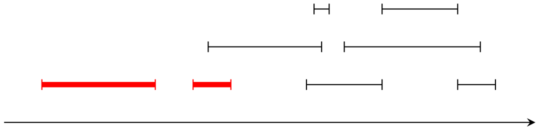
Earliest Finish Time

Process jobs in the order of their finishing times, beginning with those that finish earliest.



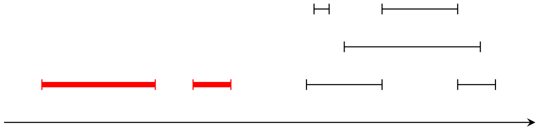
Earliest Finish Time

Process jobs in the order of their finishing times, beginning with those that finish earliest.



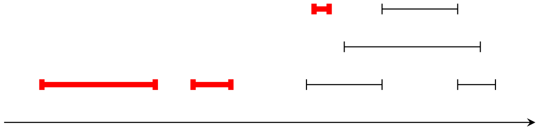
Earliest Finish Time

Process jobs in the order of their finishing times, beginning with those that finish earliest.



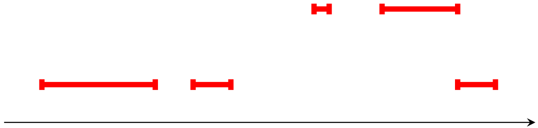
Earliest Finish Time

Process jobs in the order of their finishing times, beginning with those that finish earliest.



Earliest Finish Time

Process jobs in the order of their finishing times, beginning with those that finish earliest.



Optimal Greedy Algorithm

```
R is the set of all requests
X ← ∅ (* X stores the jobs that will be scheduled *)
while R is not empty
    choose i ∈ R such that finishing time of i is smallest
    add i to X
    remove from R all requests that overlap with i
return X
```

Theorem 19.2.

The greedy algorithm that picks jobs in the order of their finishing times is optimal.

Implementation and Running Time

```
Initially R is the set of all requests
X ← ∅ (* X stores the jobs that will be scheduled *)
while R is not empty
    choose i ∈ R such that finishing time of i is least
    if i does not overlap with requests in X
        add i to X
    remove i from R
return the set X
```

- Presort all requests based on finishing time. $O(n \log n)$ time
- Now choosing least finishing time is $O(1)$
- Keep track of the finishing time of the last request added to A. Then check if starting time of i later than that
- Thus, checking non-overlapping is $O(1)$
- Total time $O(n \log n + n) = O(n \log n)$

Comments

1. Interesting Exercise: smallest interval first picks at least half the optimum number of intervals.
2. All requests need not be known at the beginning. Such online algorithms are a subject of research

Weighted Interval Scheduling

Suppose we are given n jobs. Each job i has a start time s_i, a finish time f_i, and a weight w_i. We would like to find a set S of compatible jobs whose total weight is maximized. Which of the following greedy algorithms finds the optimum schedule?

- Earliest start time first.
- Earliest finish time fist.
- Highest weight first.
- None of the above.
- IDK.

Weighted problem can be solved via dynamic programming. See notes.

Weighted Interval Scheduling

Suppose we are given n jobs. Each job i has a start time s_i, a finish time f_i, and a weight w_i. We would like to find a set S of compatible jobs whose total weight is maximized. Which of the following greedy algorithms finds the optimum schedule?

- Earliest start time first.
- Earliest finish time fist.
- Highest weight first.
- None of the above.
- IDK.

Weighted problem can be solved via dynamic programming. See notes.

THE END

…

(for now)