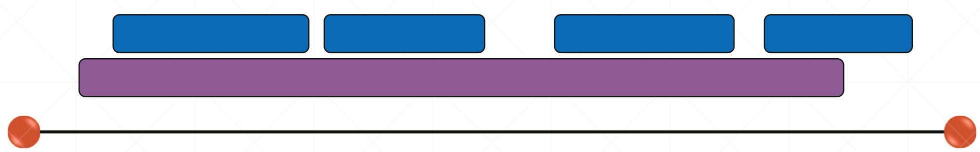
Dynamic Programming Algorithms

Max Weight Interval Scheduling

CS 336: Design and Analysis of Algorithms
Konstantin Makarychev

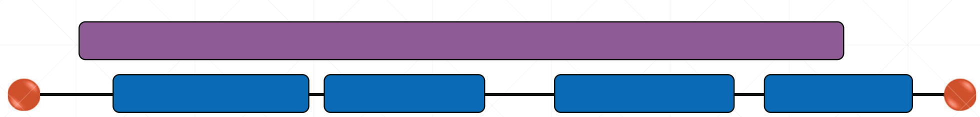
Weighted Scheduling

- Given: a set of jobs J. Each job has a start time s_j, finish time f_j, and weight w_j.
- Goal: schedule a subset $S \subset J$ of jobs on a single machine so as to maximize the total weight of scheduled jobs: $\sum_{j \in S} w_j$.



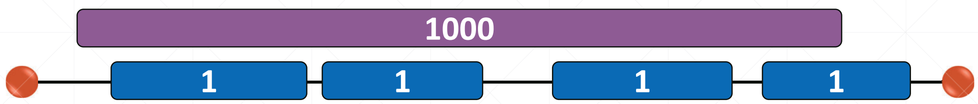
Weighted Scheduling

- Given: a set of jobs J. Each job has a start time s_j, finish time f_j, and weight w_j.
- Goal: schedule a subset $S \subset J$ of jobs on a single machine so as to maximize the total weight of scheduled jobs: $\sum_{j \in S} w_j$.



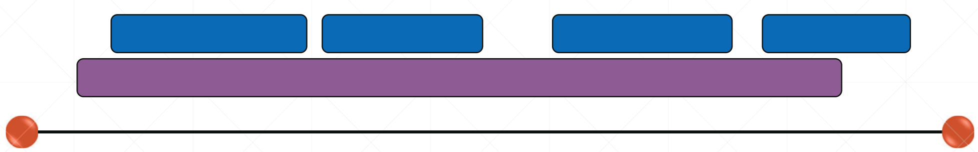
Weighted Scheduling

- Given: a set of jobs J. Each job has a start time s_j, finish time f_j, and weight w_j.
- Goal: schedule a subset $S \subset J$ of jobs on a single machine so as to maximize the total weight of scheduled jobs: $\sum_{j \in S} w_j$.



Weighted Scheduling

- Given: a set of jobs J. Each job has a start time s_j, finish time f_j, and weight w_j.
- Goal: schedule a subset $S \subset J$ of jobs on a single machine so as to maximize the total weight of scheduled jobs: $\sum_{j \in S} w_j$.



Dynamic Programming

- 1st step: Define a subproblem. Subproblem j: Maximum number of scheduled jobs in the set

$$\{1,2,\dots,j\}$$

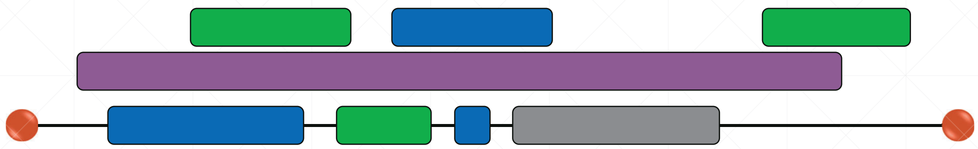
- 2nd step: Define the ordering. Let's schedule jobs from left to right. Sort all jobs by f_j as we did before.
- Assume:

$$f_1 \le \cdots \le f_n$$

Dynamic Programming for Interval Scheduling

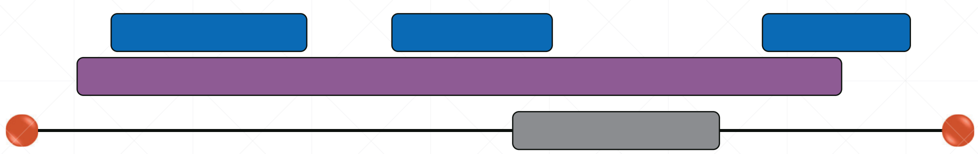
- Subproblem j: Maximum number of scheduled jobs in the set

$$\{1,2, \dots, j\}$$



Subproblems

- Let I_j be the optimal schedule for $\{1, \dots, j\}$.
- Let OPT_j be the value of I_j.
- Find the optimal solutions for the first $j-1$ subproblems.

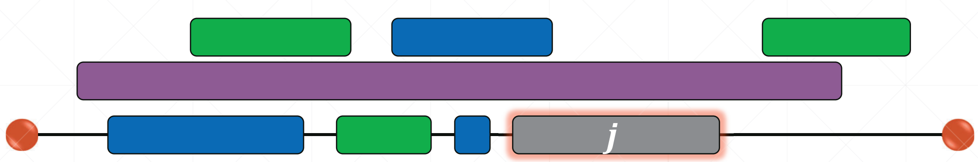


Solve subproblem j

Consider two options: $j \notin I_j$ and $j \in I_j$.

If j is ***not*** scheduled in the optimal solution i.e., $j \notin I_j$, then

➢ $I_j = ?$

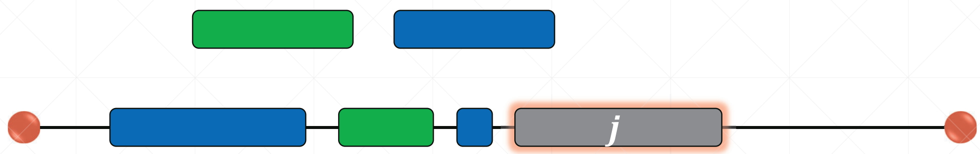


Solve subproblem j

Consider two options: $j \notin I_j$ and $j \in I_j$.

If j is ***not*** scheduled in the optimal solution i.e., $j \notin I_j$, then

- $I_j = ?$

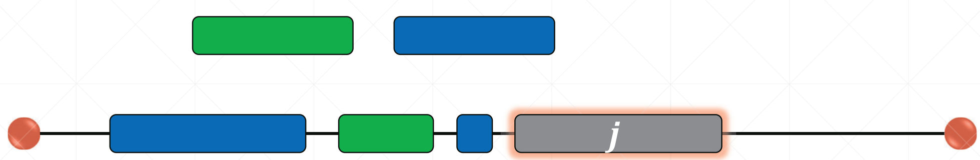


Solve subproblem j

Consider two options: $j \notin I_j$ and $j \in I_j$.

If j is ***not*** scheduled in the optimal solution i.e., $j \notin I_j$, then

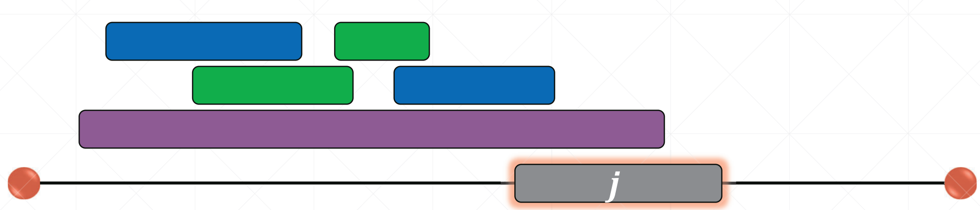
- $I_j = I_{j-1}$



Weighted Scheduling for first j jobs

If j is scheduled in the optimal solution i.e., $j \in I_j$, then

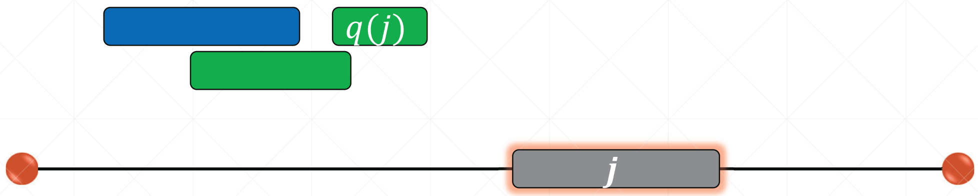
- $I_j = ?$
- Remove all jobs that overlap with j.



Weighted Scheduling for first j jobs

If j is scheduled in the optimal solution i.e., $j \in I_j$, then

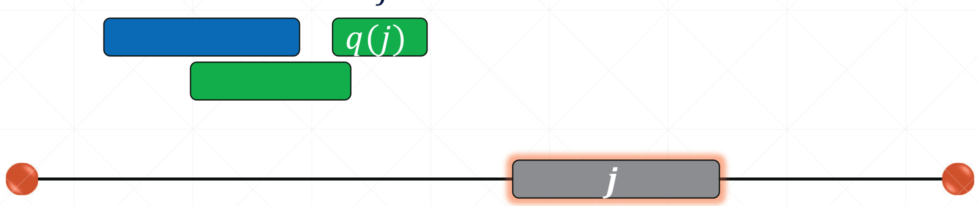
- $I_j = ?$
- Remove all jobs that overlap with j. Let $q(j)$ be the last job that finishes before j starts.



Weighted Scheduling for first j jobs

If j is scheduled in the optimal solution i.e., $j \in I_j$, then

- $I_j = I_{q(j)} \cup \{j\}$
- Remove all jobs that overlap with j. Let $q(j)$ be the last job that finishes before j starts.



DP for Max Weight Interval Scheduling

Sort jobs by f_j
For each j: let $q(j)$ be the last job that finishes before j starts.

$$q(j) = \max\{j' \in J: f_{j'} \leq s_j\}$$

For j = 1 **to** n:
  Let I_j be the best of two solutions: $\{j\} \cup I_{q\{j\}}$ and I_{j-1}.
  Let OPT_j be the value of that solution.
Return I_n.

DP for Max Weight Interval Scheduling

Sort jobs by f_j // $O(n \log_2 n)$ instructions
For each j: let $q(j)$ be the last job that finishes before j starts.

$$q(j) = \max\{j' \in J: f_{j'} \le s_j\} \text{ // O(?) operations}$$

For j = 1 **to** n: // $O(n)$ instructions for the entire loop.
Let I_j be the best of two solutions: $\{j\} \cup I_{q\{j\}}$ and I_{j-1}.
Let OPT_j be the value of that solution.
Return I_n.

Computing q_j

$$q(j) = \max\{j' \in J: f_{j'} \le s_j\}$$

$q(j)$ is the last job that finishes before job j starts.

Use binary search!

- **Running time:**

$O(\log n)$ instructions per each computation of $q(j)$.