Dynamic Programming

Reading: Section 6.1, 6.2 and 6.4
Weighted Interval Scheduling

- Set of $n$ jobs to be executed
- Each job has a start time and a finish time: $s_1, \ldots, s_n, f_1, \ldots, f_n$
- Each job has a weight or value $v_i$
- Jobs cannot run in parallel
  - Jobs are compatible if their time does not overlap

- **Goal:** Find maximum-weight set of compatible jobs
Complexity of Solution

**BRUTE-FORCE** \((n, s_1, \ldots, s_n, f_1, \ldots, f_n, w_1, \ldots, w_n)\)

Sort jobs by finish time and renumber so that \(f_1 \leq f_2 \leq \ldots \leq f_n\).
Compute \(p[1], p[2], \ldots, p[n]\) via binary search.

**RETURN** \(\text{COMPUTE-OPT}(n)\).

**COMPUTE-OPT** \((j)\)

**IF** \((j = 0)\)

**RETURN** \(0\).

**ELSE**

**RETURN** \(\max\ \{\text{COMPUTE-OPT}(j-1), w_j + \text{COMPUTE-OPT}(p[j])\}\).
Memoization

- Top-down approach
  - Cache result of each subproblem

\[\text{TOP-DOWN}(n, s_1, \ldots, s_n, f_1, \ldots, f_n, w_1, \ldots, w_n)\]

Sort jobs by finish time and renumber so that \(f_1 \leq f_2 \leq \ldots \leq f_n\).

Compute \(p[1], p[2], \ldots, p[n]\) via binary search.

\(M[0] \leftarrow 0.\) \hspace{1cm} \text{(global array)}

\text{RETURN} \ M\text{-COMPUTE-OPT}(n).

\text{M-COMPUTE-OPT}(j)

\text{IF (}M[j]\text{ is uninitialized)}

\[M[j] \leftarrow \max \{ \text{M-COMPUTE-OPT}(j-1), w_j + \text{M-COMPUTE-OPT}(p[j]) \}.\]

\text{RETURN} \ M[j].\]
Iteration over Subproblems

- Bottom-up approach

\[ \text{BOTTOM-UP}(n, s_1, \ldots, s_n, f_1, \ldots, f_n, w_1, \ldots, w_n) \]

Sort jobs by finish time and renumber so that \( f_1 \leq f_2 \leq \ldots \leq f_n \).
Compute \( p[1], p[2], \ldots, p[n] \).
\( M[0] \leftarrow 0. \)

For \( j = 1 \) to \( n \)
\[ M[j] \leftarrow \max \{ M[j-1], w_j + M[p[j]] \} \]
Subset Sum

**Problem.** Given \( n \) jobs where job \( i \) requires \( \mathcal{W}_i \) minutes of time and a budget \( W \).

- Find subset \( S \) that maximizes \( \sum_{i \in S} w_i \) and has \( \sum_{i \in S} w_i \leq W \).
Subset Sum

**Problem.** Given $n$ jobs where job $i$ requires $w_i$ minutes of time and a budget $W$.

- Find subset $S$ that maximizes $\sum_{i \in S} w_i$ and has $\sum_{i \in S} w_i \leq W$.

- Ideas?
- What should the recurrence look like?
**Knapsack Problem**

**Problem.** Given $n$ items where item $i$ has value $v_i$ and weight $w_i$, and limit is $W$.

- Find subset $S$ that maximizes $\sum_{i \in S} v_i$ and has $\sum_{i \in S} w_i \leq W$.

- How does this problem relate to Subset Sum?