Dynamic Programming

Reading: Section 6.4 and 6.8
Weighted Interval Scheduling

- Iteration over subproblems

```markdown
**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 \(n\)

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

previously computed values
## Finding Solution

<table>
<thead>
<tr>
<th>Index</th>
<th>$w_1 = 2$</th>
<th>$w_2 = 4$</th>
<th>$w_3 = 4$</th>
<th>$w_4 = 7$</th>
<th>$w_5 = 2$</th>
<th>$w_6 = 1$</th>
<th>$p(1) = 0$</th>
<th>$p(2) = 0$</th>
<th>$p(3) = 1$</th>
<th>$p(4) = 0$</th>
<th>$p(5) = 3$</th>
<th>$p(6) = 3$</th>
</tr>
</thead>
</table>

- Compute M
- Getting the solution back
Finding Solution

Find-Solution(j)
    If $j = 0$ then
        Output nothing
    Else
        If $v_j + M[p(j)] \geq M[j - 1]$ then
            Output $j$ together with the result of Find-Solution($p(j)$)
        Else
            Output the result of Find-Solution($j - 1$)
        Endif
    Endif
Endif
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 $\sum_{i \in S} w_i \leq W$. 
Subset Sum

- Given the recurrence if $w < w_i$ then $OPT(i, w) = OPT(i - 1, w)$, otherwise

$$OPT(i, w) = \max\{OPT(i - 1, w), OPT(i - 1, w - w_i)\}$$
Subset Sum

- Given the recurrence if $w < w_i$ then $OPT(i, w) = OPT(i - 1, w)$, otherwise

$$OPT(i, w) = \max\{OPT(i - 1, w), OPT(i - 1, w - w_i)\}$$

- Three jobs with weights 2, 2 and 1 and W=4.
- How to compute the solution given M?

<table>
<thead>
<tr>
<th></th>
<th>$w = 0$</th>
<th>$w = 1$</th>
<th>$w = 2$</th>
<th>$w = 3$</th>
<th>$w = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$j = 3$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>$j = 2$</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>$j = 1$</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$j = 0$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Shortest Path

- We have seen Dijkstra’s algorithm for solving this problem
- Are there any conditions for it to work?
Shortest Path

- We have seen Dijkstra’s algorithm for solving this problem
- Are there any conditions for it to work?
- Negative weights
- Negative cycles
Shortest Path

- We have seen Dijkstra’s algorithm for solving this problem.
- Are there any conditions for it to work?

- **Negative weights but no negative cycles**
- Negative cycles
Dynamic Programming approach

- If no negative cycles, then there is a shortest path that is a simple path.
- What are the subproblems we have to look into?
Dynamic Programming approach

- If no negative cycles, then there is a shortest path that is a simple path.

- What are the subproblems we have to look into?
Dynamic Programming approach

- If no negative cycles, then there is a shortest path that is a simple path.

- What are the subproblems we have to look into?

- If $i$ is positive

\[
OPT(i, v) = \min \{ OPT(i - 1, v), \min_{w \in V} \{ OPT(i - 1, w) + c_{vw} \} \}\]
Bellman-Ford Algorithm

Shortest-Path\((G, s, t)\)

\[ n = \text{number of nodes in } G \]

Array \[ M[0...n-1, V] \]

Define \[ M[0, t] = 0 \] and \[ M[0, v] = \infty \] for all other \( v \in V \)

For \( i = 1, \ldots, n - 1 \)

For \( v \in V \) in any order

Compute \( M[i, v] \) using the recurrence

Endfor

Endfor

Return \( M[n - 1, s] \)
Exercise

- Find cost of shortest s-z path
Exercise (solution)

- Find cost shortest s-z path

```
\begin{array}{c|ccccc}
  i/v & s & t & y & x & z \\
  \hline
  0 & \infty & \infty & \infty & \infty & 0 \\
  1 & \infty & \infty & 9 & \infty & 0 \\
  2 & 2 & -4 & 9 & -6 & 0 \\
  3 & 2 & -4 & -9 & -6 & 0 \\
  4 & -2 & -4 & -9 & -6 & 0 \\
\end{array}
```