Design a Dynamic Programming Algorithm

- Consider the instance: $i, W', (w_1, w_2, \cdots, w_i)$;
- $opt[i, W']$: the optimum value of the instance

Q: The value of the optimum solution that does not contain i?

A: $opt[i-1, W']$

Q: The value of the optimum solution that contains i?

A: $opt[i-1, W'-w_i] + w_i$

Dynamic Programming

- Consider the instance: $i, W', (w_1, w_2, \cdots, w_i)$;
- $opt[i, W']$: the optimum value of the instance

$$opt[i, W'] = \begin{cases} & i = 0 \\ & i > 0, w_i > W' \\ & i > 0, w_i \leq W' \end{cases}$$

Dynamic Programming

- Consider the instance: $i, W', (w_1, w_2, \cdots, w_i)$;
- $opt[i, W']$: the optimum value of the instance

$$opt[i, W'] = \begin{cases} 0 & i = 0 \\ & i > 0, w_i > W' \\ & i > 0, w_i \le W' \end{cases}$$

Dynamic Programming

- Consider the instance: $i, W', (w_1, w_2, \cdots, w_i)$;
- $opt[i, W']$: the optimum value of the instance

$$opt[i, W'] = \begin{cases} 0 & i = 0 \\ opt[i-1, W'] & i > 0, w_i > W' \\ \\ & i > 0, w_i \leq W' \end{cases}$$

Dynamic Programming

- Consider the instance: $i, W', (w_1, w_2, \cdots, w_i)$;
- $opt[i, W']$: the optimum value of the instance

$$opt[i, W'] = \begin{cases} 0 & i = 0 \\ opt[i-1, W'] & i > 0, w_i > W' \\ \max \left\{ \begin{array}{c} opt[i-1, W'] \\ opt[i-1, W'-w_i] + w_i \end{array} \right\} & i > 0, w_i \leq W' \end{cases}$$

Dynamic Programming

```
 1: for W' ← 0 to W do
 2:     opt[0, W'] ← 0
 3: for i ← 1 to n do
 4:     for W' ← 0 to W do
 5:         opt[i, W'] ← opt[i − 1, W']
 6:         if w_i ≤ W' and opt[i − 1, W' − w_i] + w_i ≥ opt[i, W'] then
 7:             opt[i, W'] ← opt[i − 1, W' − w_i] + w_i
 8: return opt[n, W]
```

Recover the Optimum Set

```
 1: for W' ← 0 to W do
 2:     opt[0, W'] ← 0
 3: for i ← 1 to n do
 4:     for W' ← 0 to W do
 5:         opt[i, W'] ← opt[i − 1, W']
 6:         b[i, W'] ← N
 7:         if wᵢ ≤ W' and opt[i − 1, W' − wᵢ] + wᵢ ≥ opt[i, W']
    then
 8:             opt[i, W'] ← opt[i − 1, W' − wᵢ] + wᵢ
 9:             b[i, W'] ← Y
10: return opt[n, W]
```

Recover the Optimum Set

```
1: i ← n, W' ← W, S ← ∅
2: while i > 0 do
3:     if b[i, W'] = Y then
4:         W' ← W' − w_i
5:         S ← S ∪ {i}
6:     i ← i − 1
7: return S
```

Running Time of Algorithm

```
1: for W' ← 0 to W do
2:      opt[0, W'] ← 0
3: for i ← 1 to n do
4:      for W' ← 0 to W do
5:          opt[i, W'] ← opt[i − 1, W']
6:          if w_i ≤ W' and opt[i − 1, W' − w_i] + w_i ≥ opt[i, W'] then
7:              opt[i, W'] ← opt[i − 1, W' − w_i] + w_i
8: return opt[n, W]
```

Running Time of Algorithm

1: **for** $W' \leftarrow 0$ to W **do**
2:     $opt[0, W'] \leftarrow 0$
3: **for** $i \leftarrow 1$ to n **do**
4:     **for** $W' \leftarrow 0$ to W **do**
5:         $opt[i, W'] \leftarrow opt[i - 1, W']$
6:         **if** $w_i \leq W'$ and $opt[i - 1, W' - w_i] + w_i \geq opt[i, W']$ **then**
7:             $opt[i, W'] \leftarrow opt[i - 1, W' - w_i] + w_i$
8: **return** $opt[n, W]$

- Running time is $O(nW)$

Running Time of Algorithm

```
1: for W' ← 0 to W do
2:     opt[0, W'] ← 0
3: for i ← 1 to n do
4:     for W' ← 0 to W do
5:         opt[i, W'] ← opt[i − 1, W']
6:         if wᵢ ≤ W' and opt[i − 1, W' − wᵢ] + wᵢ ≥ opt[i, W'] then
7:             opt[i, W'] ← opt[i − 1, W' − wᵢ] + wᵢ
8: return opt[n, W]
```

$$
\begin{aligned}
&1:\ \textbf{for } W' \leftarrow 0 \text{ to } W \textbf{ do}\\
&2:\quad opt[0, W'] \leftarrow 0\\
&3:\ \textbf{for } i \leftarrow 1 \text{ to } n \textbf{ do}\\
&4:\quad \textbf{for } W' \leftarrow 0 \text{ to } W \textbf{ do}\\
&5:\qquad opt[i, W'] \leftarrow opt[i-1, W']\\
&6:\qquad \textbf{if } w_i \leq W' \text{ and } opt[i-1, W'-w_i] + w_i \geq opt[i, W'] \textbf{ then}\\
&7:\qquad\quad opt[i, W'] \leftarrow opt[i-1, W'-w_i] + w_i\\
&8:\ \textbf{return } opt[n, W]
\end{aligned}
$$

- Running time is $O(nW)$
- Running time is pseudo-polynomial because it depends on value of the input integers.

Running Time of Algorithm

```
1: for W' ← 0 to W do
2:     opt[0, W'] ← 0
3: for i ← 1 to n do
4:     for W' ← 0 to W do
5:         opt[i, W'] ← opt[i − 1, W']
6:         if w_i ≤ W' and opt[i − 1, W' − w_i] + w_i ≥ opt[i, W'] then
7:             opt[i, W'] ← opt[i − 1, W' − w_i] + w_i
8: return opt[n, W]
```

- Running time is $O(nW)$
- Running time is pseudo-polynomial because it depends on value of the input integers.
- Game's running time:
  https://courses.csail.mit.edu/6.5440/fall23/

Avoiding Unncessary Computation and Memory Using Memoized Algorithm and Hash Map

compute-opt(i, W')

1: **if** $opt[i, W'] \neq \bot$ **then return** $opt[i, W']$
2: **if** $i = 0$ **then** $r \leftarrow 0$
3: **else**
4:     $r \leftarrow$ compute-opt$(i - 1, W')$
5:     **if** $w_i \leq W'$ **then**
6:         $r' \leftarrow$ compute-opt$(i - 1, W' - w_i) + w_i$
7:         **if** $r' > r$ **then** $r \leftarrow r'$
8: $opt[i, W'] \leftarrow r$
9: **return** r

- Use hash map for opt

Outline