# Design and Analysis of Algorithms

# L30: Knapsack problem
## Dynamic Programming

Dr. Ram P Rustagi
Sem IV (2020-Even)
Dept of CSE, KSIT
rprustagi@ksit.edu.in

# Resources

- Text book 1: Levitin
  - Sec `8.2, 8.3,` **<u>`8.4`</u>**
- Text book 2: Horowitz
  - Sec `5.1,5.2,5.4,5.8,5.9`

- R1: Introduction to Algorithms
    - Cormen et al.

# Knapsack problem

- Knapsack problem:
  - Given $n$ items of known weights $w_1, \ldots, w_n$, and
  - their values $v_1, \ldots, v_n$ and a capacity $W$
  - Find the most valuable subset of items that fit into the knapsack.
  - Note: All the weights $w_i$'s and knapsack capacity $W$ are integers, but values can be real numbers.
- Goal: solve the knapsack problem using dynamic programming.

# DP Approach: Knapsack

- To solve knapsack problem using DP,
  - need to design a recurrence relation, that
    - expresses a solution to an instance in terms of smaller instances.
- Consider an instance defined by first $i$ items with
  - weights $w_1, w_2, \ldots, w_i$; $1 \le i \le n$
  - values $v_1, v_2, \ldots, v_i$; $1 \le i \le n$
  - and knapsack capacity $j$, $1 \le j \le w$
- Let $V[i,j]$ be the optimal solution to this instance
  - i.e. the value of most valuable subsets of first $i$ items that fit knapsack of capacity $j$.
- Approach: divide first $i$ items into two categories:
  - those that don't include $i^{th}$ item, and those that do.

# DP Approach: Knapsack

| | 0 | | $j-w_i$ | | $j$ | | W |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | | | | | | |
| $i-1$ | 0 | | **V[i-1,j-$w_i$]** | | **V[i-1,j]** | | |
| $i$ | 0 | | | | **V[i,j]** | | |
| | 0 | | | | | | |
| n | 0 | | | | | | |

Table for solving knapsack problem using dynamic programming

- Category 1: subsets that do not include $i^{th}$ item.
  - Value of optimal subset is V[i-1,j]
- Category 2: subsets that do include $i^{th}$ item.
  - Thus $j>w_i$ i.e. $j-w_i \geq 0$.
  - Value of optimal subset is $v_i$+V[i-1,j-$w_i$]

# DP Approach: Knapsack

- Possible cases:
  - $j<w_i$ (i.e. $j-w_i<0$), i.e. weight of $i^{th}$ item is more than $j$ and thus can't be included
  - $j \geq w_i$ (i.e $j-w_i \geq 0$) weight of $i^{th}$ item is less than or equal to $j$, and thus $i^{th}$ item may included or execluded.
- Thus,

$$V[i,j] = \begin{cases} max\{V[i-1,j], v_i + V[i-1,\ j-w_i]\} & if\ j-w_i \geq 0 \\ V[i-1,\ j] & if\ j-w_i < 0 \end{cases} \qquad (1)$$

- The initial conditions can be defined as
  $V[i,0]=0$ for $i \geq 0$, and
  $V[0,j]=0$ for $j \geq 0$

# Example: Knapsack

$$V[i,j] = \begin{cases} max\{V[i-1,j], v_i + V[i-1,\ j-w_i]\} & if\ j-w_i \geq 0 \\ V[i-1,\ j] & if\ j-w_i < 0 \end{cases} \quad (1)$$

- Example: consider knapsack of size 5 (i.e. max weight it can hold is 5),
  - with weights as
    $w_1$=2, $w_2$=1, $w_3$=3, $w_4$=2
  - and values as
    $v_1$=\$12, $v_2$=\$10, $v_3$=\$20, $v_4$=\$15

# Example Knapsack

| Capacity→ wts, values↓ | | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
| | 0 | **0** | **0** | **0** | **0** | **0** | **0** |
| $w_1=2$ $v_1=12$ | 1 | **0** | **0** | **12** | | | |
| $w_2=1$ $v_2=10$ | 2 | **0** | | | | | |
| $w_3=3$ $v_3=20$ | 3 | **0** | | | | | |
| $w_4=2$ $v_4=15$ | 4 | **0** | | | | | |

$V[0,j]=0$ **for** $0 \leq j \leq 5$

$V[i,0]=0$ **for** $0 \leq i \leq 4$

$V[1,1]=V[1-1,1]$ **since** $j=1<w_1=2$

$\quad =0$

$V[1,2]=\max\{V[0,2],12+V[0,2-2];j=2 \geq w_1=2$

$\quad =12$

| Capacity→ wts, values↓ | | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
| | 0 | **0** | **0** | **0** | **0** | **0** | **0** |
| $w_1=2$ $v_1=12$ | 1 | **0** | **0** | **12** | **12** | **12** | **12** |
| $w_2=1$ $v_2=10$ | 2 | **0** | **10** | | | | |
| $w_3=3$ $v_3=20$ | 3 | **0** | | | | | |
| $w_4=2$ $v_4=15$ | 4 | **0** | | | | | |

$V[1,3]=\max\{V[0,3],12+V[0,3-2]\};j=3\geq w_1=2$
$=\max\{0,12+V[0,1]\}=12$

$V[1,4]=\max\{V[0,4],12+V[0,4-2]\};j=4\geq w_1=2$
$=\max\{0,12+V[0,2]\}=12$

$V[1,5]=\max\{V[0,5],12+V[0,5-2]\};j=5\geq w_1=2$
$=\max\{0, 12+V[0,3]\}=12$

$V[2,1]=\max\{V[1,1],10+V[1,1-1]\}; j=1\geq w_2=1$
$=\max\{0, 10+V[1,0]\} = 10$

| Capacity→ wts, values↓ | | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
| | 0 | **0** | **0** | **0** | **0** | **0** | **0** |
| $w_1=2$ $v_1=12$ | 1 | **0** | **0** | **12** | **12** | **12** | **12** |
| $w_2=1$ $v_2=10$ | 2 | **0** | **10** | **12** | **22** | **22** | **22** |
| $w_3=3$ $v_3=20$ | 3 | **0** | | | | | |
| $w_4=2$ $v_4=15$ | 4 | **0** | | | | | |

$V[2,2]=\max\{V[1,2],\ 10+V[1,2-1]\};\ j=2\geq w_2=1$
$\quad =\max\{12,\ 10+0\} = 12$

$V[2,3]=\max\{V[1,3],\ 10+V[1,3-1]\};\ j=3\geq w_2=1$
$\quad =\max\{12,\ 10+V[1,2]\}=\max\{12,22\}=22$

$V[2,4]=\max\{V[1,4],\ 10+V[1,4-1]\};\ j=4\geq w_2=1$
$\quad =\max\{12,\ 10+V[1,3]\}=\max\{12,22\}=22$

$V[2,5]=\max\{V[1,5],\ 10+V[1,5-1]\};\ j=5\geq w_2=1$
$\quad =\max\{12,\ 10+V[1,4]\}=\max\{12,22\}=22$

| Capacity→ wts, values↓ | | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
| | 0 | **0** | **0** | **0** | **0** | **0** | **0** |
| $w_1=2 \ v_1=12$ | 1 | **0** | **0** | **12** | **12** | **12** | **12** |
| $w_2=1 \ v_2=10$ | 2 | **0** | **10** | **12** | **22** | **22** | **22** |
| $w_3=3 \ v_3=20$ | 3 | **0** | **10** | **12** | **22** | **30** | **32** |
| $w_4=2 \ v_4=15$ | 4 | **0** | | | | | |

$V[3,1]=V[2,1] = 10;$ $(j=1<w_3=3)$

$V[3,2]=V[2,2] = 12;$ $(j=2<w_3=3)$

$V[3,3]=\max\{V[2,3], 20+V[2,3-3]\};$ $(j=3\geq w_3=3)$

$\quad =\max\{22, 20+0\} = 22$

$V[3,4]=\max\{V[2,4], 20+V[2,4-3]\};$ $(j=4\geq w_3=3)$

$\quad =\max\{22, 20+V[2,1]\}=\max\{22,30\}=30$

$V[3,5]=\max\{V[2,5], 20+V[2,5-3]\};$ $(j=5\geq w_3=3)$

$\quad =\max\{12, 20+V[2,2]\}=\max\{12,20+12\}=32$

12

| Capacity→ wts, values↓ | | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
| | 0 | **0** | **0** | **0** | **0** | **0** | **0** |
| $w_1=2$ $v_1=12$ | 1 | **0** | **0** | **12** | **12** | **12** | **12** |
| $w_2=1$ $v_2=10$ | 2 | **0** | **10** | **12** | **22** | **22** | **22** |
| $w_3=3$ $v_3=20$ | 3 | **0** | **10** | **12** | **22** | **30** | **32** |
| $w_4=2$ $v_4=15$ | 4 | **0** | **10** | **15** | **25** | **30** | **37** |

$V[4,1]=V[3,1] = 10$; $(j=1<w_4=2)$

$V[4,2]=\max\{V[3,2],15+V[3,2-2]\}$; $(j=2\geq w_4=2)$

$=\max\{12,15+V[3,0]\}=\max\{12,15+0\}=15$

$V[4,3]=\max\{V[3,3], 15+V[3,3-2]\}$; $(j=3\geq w_4=2)$

$=\max\{22,15+V[3,1]\}=\max\{22,15+10\}=25$

$V[4,4]=\max\{V[3,4], 15+V[3,4-2]\}$; $(j=4\geq w_4=2)$

$=\max\{30, 15+V[3,2]\}=\max\{30,15+12\}=30$

$V[4,5]=\max\{V[3,5], 15+V[3,5-2]\}$; $(j=5\geq w_4=2)$

$=\max\{32, 15+V[3,3]\}=\max\{32,15+22\}=37$

# Example Knapsack: Optimal Subset

| Capacity→<br>wts, values↓ | | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
| | 0 | **0** | **0** | **0** | **0** | **0** | **0** |
| $w_1=2$ $v_1=12$ | 1 | **0** | **0** | **12** | **12** | **12** | **12** |
| $w_2=1$ $v_2=10$ | 2 | **0** | **10** | **12** | **22** | **22** | **22** |
| $w_3=3$ $v_3=20$ | 3 | **0** | **10** | **12** | **22** | **30** | **32** |
| $w_4=2$ $v_4=15$ | 4 | **0** | **10** | **15** | **25** | **30** | **37** |

- Optimal subset
  - Backtrack from maximal value V[4,5] to prev. rows.
  - Thus, optimal subsets are

V[4,5]=37 (≠V[3,5]) implies $w_4=2$ is included

V[3,3]=22 (=V[2,3]) implies $w_3=3$ is not included

V[2,3]=22 (≠V[1,3] implies $w_2=1$ is included

V[1,2]=12 (≠V[0,2] implies $w_1=2$ is included

# Algorithm: Knapsack using DP

```
Algo DPKnapsack(w[1..n], v[1..n], W)
  int V[0..n,0..W],  P[1..n,1..W];
  for j=0 to W do
     V[0,j]=0
  for i=0 to n do
     V[i,0]= 0
  for i=1 to n do
     for j= 1 to W do
       if w[i]≤j and v[i]+V[i-1,j-w[i]]>V[i-1,j] then
          V[i,j]= v[i] + V[i-1,j-w[i]];
          P[i,j]= j-w[i]
       else
          V[i,j]= V[i-1,j]
          P[i,j] := j
  return V[n,W] and the optimal subset by backtracing
```

# Efficiency of Knapsack

- Time Efficiency: $\Theta(nW)$
- Space efficiency: $\Theta(nW)$

# Summary

- Knapsack algorithm using dynamic programming
- Efficiency
- optimal subsets using backtracking