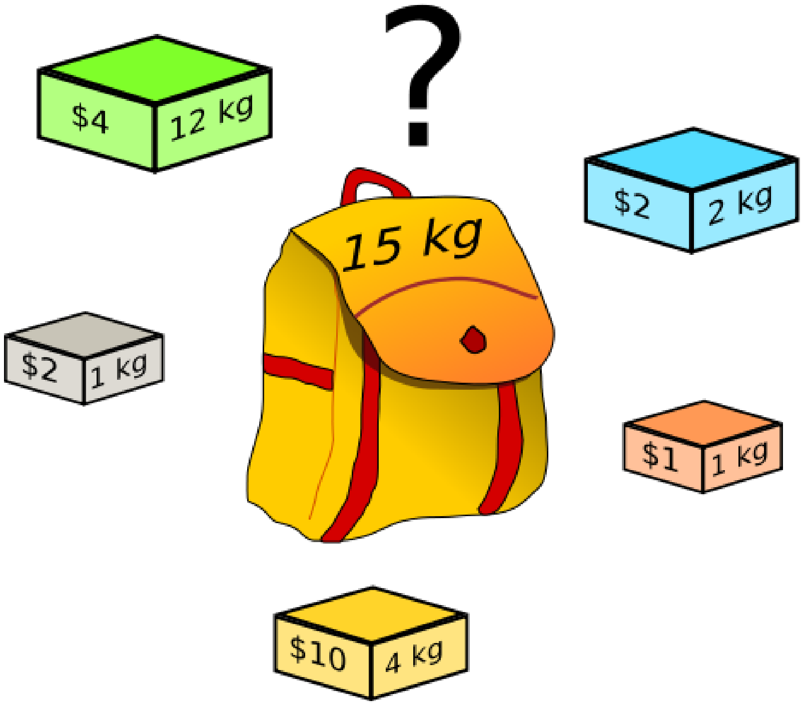
VE281

Data Structures and Algorithms

Dynamic Programming: Knapsack Problem

0/1 Knapsack Problem

- You have a set of n items, each with a weight w_i and a value p_i.
- You also have a knapsack with a weight capacity C.
- How will you choose items to put into the knapsack to maximize the total value?



- We assume w_i, C are **positive integers**.

0/1 Knapsack Problem

Mathematic Formulation

- Let $x_i = 1$ when item i is selected and $x_i = 0$ when item i is not selected.

- The problem is the following optimization problem:
  - Maximize $\sum_{i=1}^{n} p_i x_i$
  - Subject to $\sum_{i=1}^{n} w_i x_i \leq C$ and $x_i \in \{0,1\}$ for all i.

0/1 Knapsack Problem

A Brute-Force Solution

- Enumerate all the combinations of $(x_1, x_2, \ldots, x_n)$ with $x_i \in \{0,1\}$
- **Discard** combinations for which $\sum_{i=1}^{n} w_i x_i > C$
- For the remaining combinations, find the one that **maximizes** $\sum_{i=1}^{n} p_i x_i$.

- What is the time complexity?
  - Since there are 2^n combinations to consider, the time complexity is $\Omega(2^n)$.

0/1 Knapsack Problem

A Greedy Solution

- We select items one by one to put into the knapsack.
- Each item is selected using a **greedy** criterion.
- Once an item is selected, it will not be removed from the knapsack later on.

- Possible greedy criterion:
  - Choose items in decreasing order of values p_i.
  - Choose items in decreasing order of relative values p_i/w_i.

However, greedy solution may not be the optimal.

0/1 Knapsack Problem

A Greedy Solution

- Greedy attempt 1: Select items in decreasing order of values
- Example: $n = 3, C = 7$
  $(w_1, p_1) = (7, 10), (w_2, p_2) = (3, 8), (w_3, p_3) = (2, 6)$
- Only item 1 will be selected.
  - Total value is 10.
- However, this is not optimal.
  - We can put item 2 and 3 together into the knapsack, which gives a total value of 14.

0/1 Knapsack Problem

A Greedy Solution

- Greedy attempt 2: Select items in decreasing order of average values p_i/w_i.
- Example: $n = 2, C = 7$
  $(w_1, p_1) = (1, 10), (w_2, p_2) = (7, 20)$
- Only item 1 will be selected.
  - Total value is 10.
- However, this is not optimal.
  - If we choose item 2, this is a better choice.

0/1 Knapsack Problem

- Can we apply **dynamic programming** to solve the knapsack problem?

- Let us consider the following problem:
  - Given i items with weights as $w_1, \dots, w_i$ and values as $p_1, \dots, p_i$ and a knapsack with weight capacity j, choose items to put into the knapsack to maximize the total values.
  - Define the above problem as Q_{ij} and the maximal value as m_{ij}.
  - Our original problem is Q_{nC}.

0/1 Knapsack Problem

- A mathematic formulation of Q_{ij}:
  - Maximize $\sum_{k=1}^{i} p_k x_k$
  - Subject to $\sum_{k=1}^{i} w_k x_k \leq j$ and $x_k \in \{0,1\}$ for all $1 \leq k \leq i$.

- Does Q_{ij} exhibit optimal substructure, i.e., does the optimal solution to Q_{ij} contain within it optimal solutions to subproblems?

0/1 Knapsack Problem

Optimal Substructure

- Let $x_1 = a_1, \ldots, x_i = a_i$ be an optimal solution to the problem Q_{ij}.

- If $a_i = 0$, then $x_1 = a_1, \ldots, x_{i-1} = a_{i-1}$ is an optimal solution to the problem $Q_{(i-1)j}$.
    - Why?
- If $a_i = 1$, then $x_1 = a_1, \ldots, x_{i-1} = a_{i-1}$ is an optimal solution to the problem $Q_{(i-1)(j-w_i)}$.
    - Why?

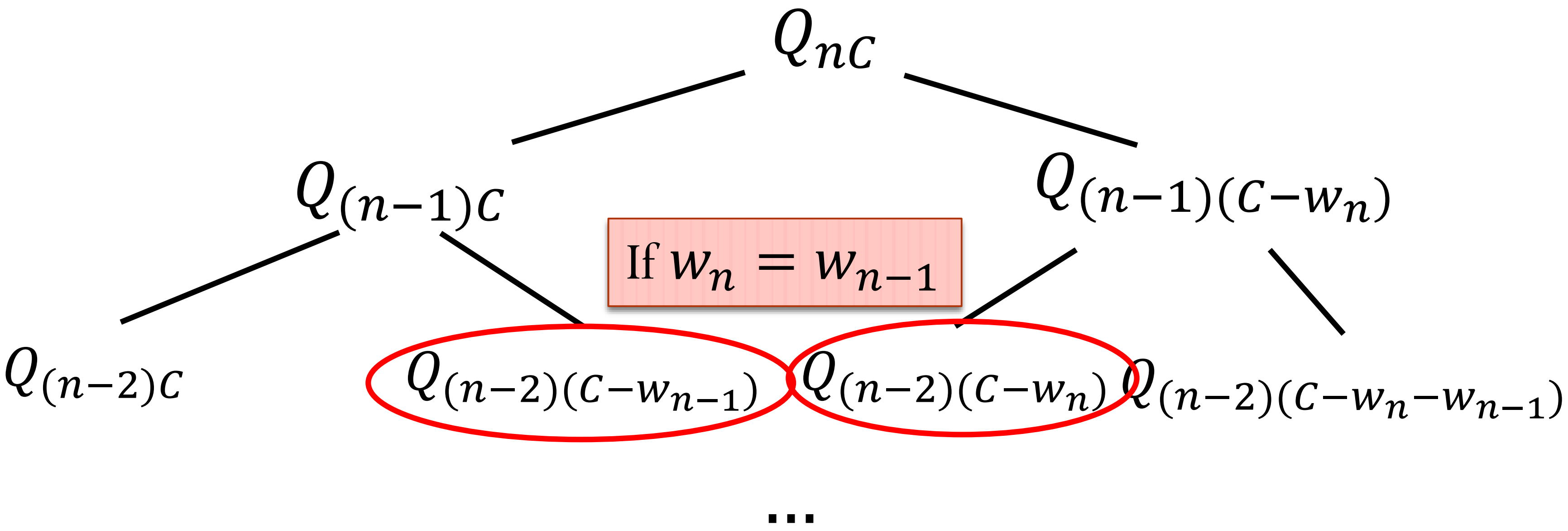
0/1 Knapsack Problem

- Since we don't know whether a_i should be 0 or 1, we need to solve both subproblems $Q_{(i-1)j}$ and $Q_{(i-1)(j-w_i)}$.

- If we have solved the subproblems $Q_{(i-1)j}$ and $Q_{(i-1)(j-w_i)}$ and obtained the maximal values as $m_{(i-1)j}$ and $m_{(i-1)(j-w_i)}$, we can obtain the maximal value for Q_{ij} as
  - $m_{ij} = \max\{m_{(i-1)j},\ m_{(i-1)(j-w_i)} + p_i\}$

0/1 Knapsack Problem

Recursion Tree



Q_{nC}

$Q_{(n-1)C}$ $Q_{(n-1)(C-w_n)}$

If $w_n = w_{n-1}$

$Q_{(n-2)C}$ $Q_{(n-2)(C-w_{n-1})}$ $Q_{(n-2)(C-w_n)}$ $Q_{(n-2)(C-w_n-w_{n-1})}$

…

- The recursion tree is a binary tree. It contains $O(2^n)$ subproblems.
- However, the subproblems may overlap depending on the values of $w_1, w_2, \ldots, w_n$.

0/1 Knapsack Problem

Dynamic Programming Algorithm

- We use a tabular, bottom-up approach.
- We will calculate all m_{ij} for $1 \leq i \leq n$ and $0 \leq j \leq C$ by applying the recursive relation:

$$m_{ij} = \max\{m_{(i-1)j}, m_{(i-1)(j-w_i)} + p_i\}$$

- In the initial round, we compute $m_{10}, m_{11}, \dots, m_{1C}$.
- In the second round, we compute $m_{20}, \dots, m_{2C}$.
  - They depend on $m_{10}, m_{11}, \dots, m_{1C}$.
- In the third round, we compute $m_{30}, \dots, m_{3C}$.
  - They depend on $m_{20}, \dots, m_{2C}$.
- So on and so forth.

0/1 Knapsack Problem

Dynamic Programming Algorithm

- In the initial round, we compute $m_{10}, m_{11}, \ldots, m_{1C}$.
  - If $j < w_1$, we cannot put item 1 into the knapsack. Thus, $m_{1j} = 0$.
  - Otherwise, $m_{1j} = p_1$.

- In the subsequent rounds for computing $m_{i0}, m_{i1}, \ldots, m_{iC}$.
  - If $j < w_i$, we cannot put item i into the knapsack. Thus, $m_{ij} = m_{(i-1)j}$.
  - Otherwise, $m_{ij} = \max\{m_{(i-1)j}, m_{(i-1)(j-w_i)} + p_i\}$.

0/1 Knapsack Problem

Example

- $n = 5, C = 8, (w_1, p_1) = (2,3), (w_2, p_2) = (6,9), (w_3, p_3) = (5,10), (w_4, p_4) = (3,8), (w_5, p_5) = (4,9)$

j

m_{ij}		0	1	2	3	4	5	6	7	8
	1									
	2									
i	3									
	4									
	5									

If $j < w_1, m_{1j} = 0$. Otherwise, $m_{1j} = p_1$

0/1 Knapsack Problem

Example

- $n = 5, C = 8, (w_1, p_1) = (2,3), (w_2, p_2) = (6,9), (w_3, p_3) = (5,10), (w_4, p_4) = (3,8), (w_5, p_5) = (4,9)$

m_{ij}						j				
		0	1	2	3	4	5	6	7	8
	1	0	0	3	3	3	3	3	3	3
	2									
i	3									
	4									
	5									

If $j < w_1, m_{1j} = 0$. Otherwise, $m_{1j} = p_1$

0/1 Knapsack Problem

Example

- $n = 5, C = 8, (w_1, p_1) = (2,3), (w_2, p_2) = (6,9),$ $(w_3, p_3) = (5,10), (w_4, p_4) = (3,8), (w_5, p_5) = (4,9)$

m_{ij}						j				
		0	1	2	3	4	5	6	7	8
	1	0	0	3	3	3	3	3	3	3
	2									
i	3									
	4									
	5									

If $j < w_2, m_{2j} = m_{1j}$.

Otherwise, $m_{2j} = \max\{m_{1j}, m_{1(j-w_2)} + p_2\}$

0/1 Knapsack Problem

Example

- $n = 5, C = 8, (w_1, p_1) = (2,3), (w_2, p_2) = (6,9), (w_3, p_3) = (5,10), (w_4, p_4) = (3,8), (w_5, p_5) = (4,9)$

m_{ij} j

	0	1	2	3	4	5	6	7	8
1	0	0	3	3	3	3	3	3	3
2	0	0	3	3	3	3	9	9	12
i 3									
4									
5									

If $j < w_2, m_{2j} = m_{1j}$.

Otherwise, $m_{2j} = \max\{m_{1j}, m_{1(j-w_2)} + p_2\}$

0/1 Knapsack Problem

Example

- $n = 5, C = 8, (w_1, p_1) = (2,3), (w_2, p_2) = (6,9), (w_3, p_3) = (5,10), (w_4, p_4) = (3,8), (w_5, p_5) = (4,9)$

m_{ij} j

		0	1	2	3	4	5	6	7	8
	1	0	0	3	3	3	3	3	3	3
	2	0	0	3	3	3	3	9	9	12
i	3									
	4									
	5									

If $j < w_3, m_{3j} = m_{2j}$.

Otherwise, $m_{3j} = \max\{m_{2j}, m_{2(j-w_3)} + p_3\}$

0/1 Knapsack Problem

Example

- $n = 5, C = 8, (w_1, p_1) = (2,3), (w_2, p_2) = (6,9), (w_3, p_3) = (5,10), (w_4, p_4) = (3,8), (w_5, p_5) = (4,9)$

m_{ij} j

i	0	1	2	3	4	5	6	7	8
1	0	0	3	3	3	3	3	3	3
2	0	0	3	3	3	3	9	9	12
3	0	0	3	3	3	10	10	13	13
4									
5									

If $j < w_3, m_{3j} = m_{2j}$.

Otherwise, $m_{3j} = \max\{m_{2j}, m_{2(j-w_3)} + p_3\}$

0/1 Knapsack Problem

Example

- $n = 5, C = 8, (w_1, p_1) = (2,3), (w_2, p_2) = (6,9),$ $(w_3, p_3) = (5,10), (w_4, p_4) = (3,8), (w_5, p_5) = (4,9)$

m_{ij}						j				
		0	1	2	3	4	5	6	7	8
	1	0	0	3	3	3	3	3	3	3
	2	0	0	3	3	3	3	9	9	12
i	3	0	0	3	3	3	10	10	13	13
	4									
	5									

If $j < w_4, m_{4j} = m_{3j}$.

Otherwise, $m_{4j} = \max\{m_{3j}, m_{3(j-w_4)} + p_4\}$

0/1 Knapsack Problem

Example

- $n = 5, C = 8, (w_1, p_1) = (2,3), (w_2, p_2) = (6,9), (w_3, p_3) = (5,10), (w_4, p_4) = (3,8), (w_5, p_5) = (4,9)$

m_{ij} j

i		0	1	2	3	4	5	6	7	8
	1	0	0	3	3	3	3	3	3	3
	2	0	0	3	3	3	3	9	9	12
i	3	0	0	3	3	3	10	10	13	13
	4	0	0	3	8	8	11	11	13	18
	5									

If $j < w_4, m_{4j} = m_{3j}$.

Otherwise, $m_{4j} = \max\{m_{3j}, m_{3(j-w_4)} + p_4\}$

0/1 Knapsack Problem

Example

- $n = 5, C = 8, (w_1, p_1) = (2,3), (w_2, p_2) = (6,9), (w_3, p_3) = (5,10), (w_4, p_4) = (3,8), (w_5, p_5) = (4,9)$

m_{ij} j

		0	1	2	3	4	5	6	7	8
	1	0	0	3	3	3	3	3	3	3
	2	0	0	3	3	3	3	9	9	12
i	3	0	0	3	3	3	10	10	13	13
	4	0	0	3	8	8	11	11	13	18
	5									

If $j < w_5, m_{5j} = m_{4j}$.

Otherwise, $m_{5j} = \max\{m_{4j}, m_{4(j-w_5)} + p_5\}$

0/1 Knapsack Problem

Example

- $n = 5, C = 8, (w_1, p_1) = (2,3), (w_2, p_2) = (6,9), (w_3, p_3) = (5,10), (w_4, p_4) = (3,8), (w_5, p_5) = (4,9)$

j

m_{ij}		0	1	2	3	4	5	6	7	8
	1	0	0	3	3	3	3	3	3	3
	2	0	0	3	3	3	3	9	9	12
i	3	0	0	3	3	3	10	10	13	13
	4	0	0	3	8	8	11	11	13	18
	5	0	0	3	8	9	11	12	17	18

Optimal Value

If $j < w_5, m_{5j} = m_{4j}$.

Otherwise, $m_{5j} = \max\{m_{4j}, m_{4(j-w_5)} + p_5\}$

0/1 Knapsack Problem

Constructing the Optimal Solution

- Given the table m_{ij}, we back track from m_{nC}.
- If $m_{ij} = m_{(i-1)j}$, we have $x_i = 0$ and we further check $m_{(i-1)j}$.
- Otherwise, we have $x_i = 1$ and we further check $m_{(i-1)(j-w_i)}$.
- Once $m_{ij} = 0$, we have $x_1 = \cdots = x_i = 0$.

0/1 Knapsack Problem

Example of Constructing the Optimal Solution

- $n = 5, C = 8, (w_1, p_1) = (2,3), (w_2, p_2) = (6,9), (w_3, p_3) = (5,10), (w_4, p_4) = (3,8), (w_5, p_5) = (4,9)$

m_{ij} j

i	0	1	2	3	4	5	6	7	8
1	0	0	3	3	3	3	3	3	3
2	0	0	3	3	3	3	9	9	12
3	0	0	3	3	3	10	10	13	13
4	0	0	3	8	8	11	11	13	18
5	0	0	3	8	9	11	12	17	18

$m_{58} = m_{48}$ → $x_5 = 0$. Further check m_{48}

0/1 Knapsack Problem

Example of Constructing the Optimal Solution

- $n = 5, C = 8, (w_1, p_1) = (2,3), (w_2, p_2) = (6,9), (w_3, p_3) = (5,10), (w_4, p_4) = (3,8), (w_5, p_5) = (4,9)$

m_{ij} j

		0	1	2	3	4	5	6	7	8
	1	0	0	3	3	3	3	3	3	3
	2	0	0	3	3	3	3	9	9	12
i	3	0	0	3	3	3	10	10	13	13
	4	0	0	3	8	8	11	11	13	18
	5	0	0	3	8	9	11	12	17	18

$x_5 = 0$

$m_{48} \neq m_{38}$ → $x_4 = 1$. Further check $m_{3(8-w_4)} = m_{35}$

0/1 Knapsack Problem

Example of Constructing the Optimal Solution

- $n = 5, C = 8, (w_1, p_1) = (2,3), (w_2, p_2) = (6,9), (w_3, p_3) = (5,10), (w_4, p_4) = (3,8), (w_5, p_5) = (4,9)$

j

m_{ij}		0	1	2	3	4	5	6	7	8
	1	0	0	3	3	3	3	3	3	3
	2	0	0	3	3	3	3	9	9	12
i	3	0	0	3	3	3	10	10	13	13
	4	0	0	3	8	8	11	11	13	18
	5	0	0	3	8	9	11	12	17	18

$x_4 = 1$

$x_5 = 0$

$m_{35} \neq m_{25}$ ➡ $x_3 = 1$. Further check $m_{2(5-w_3)} = m_{20}$

0/1 Knapsack Problem

Example of Constructing the Optimal Solution

- $n = 5, C = 8, (w_1, p_1) = (2,3), (w_2, p_2) = (6,9), (w_3, p_3) = (5,10), (w_4, p_4) = (3,8), (w_5, p_5) = (4,9)$

m_{ij} | | j | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| | 1 | 0 | 0 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| | 2 | 0 | 0 | 3 | 3 | 3 | 3 | 9 | 9 | 12 |
| i | 3 | 0 | 0 | 3 | 3 | 3 | 10 | 10 | 13 | 13 |
| | 4 | 0 | 0 | 3 | 8 | 8 | 11 | 11 | 13 | 18 |
| | 5 | 0 | 0 | 3 | 8 | 9 | 11 | 12 | 17 | 18 |

$x_3 = 1$

$x_4 = 1$

$x_5 = 0$

$m_{20} = 0$ ➡ $x_2 = x_1 = 0.$

0/1 Knapsack Problem

Example of Constructing the Optimal Solution

- $n = 5, C = 8, (w_1, p_1) = (2,3), (w_2, p_2) = (6,9), (w_3, p_3) = (5,10), (w_4, p_4) = (3,8), (w_5, p_5) = (4,9)$

m_{ij} j

		0	1	2	3	4	5	6	7	8
	1	0	0	3	3	3	3	3	3	3
	2	0	0	3	3	3	3	9	9	12
i	3	0	0	3	3	3	10	10	13	13
	4	0	0	3	8	8	11	11	13	18
	5	0	0	3	8	9	11	12	17	18

Optimal solution: $x_1 = 0, x_2 = 0, x_3 = 1, x_4 = 1, x_5 = 0$

0/1 Knapsack Problem

Time Complexity

- Obtain the maximal total value:
  - Constructing the table m_{ij} of size $n \times (C + 1)$.
  - Time complexity for calculating each entry is $O(1)$.
  - Total time complexity is $O(nC)$.

- Obtain the optimal solution:
  - $O(n)$.

0/1 Knapsack Problem

Summary

- Why dynamic-programming algorithm is suitable?
  - Optimal substructure.
  - Overlapping subproblems.

- How to design the dynamic-programming algorithm?

1. Characterize the structure of an optimal solution.
2. Recursively define the value of an optimal solution.
3. Compute the value of an optimal solution, typically in a **bottom-up** fashion.
4. Construct an optimal solution from computed information.