Dynamic Programming

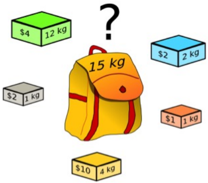
- The Knapsack Problem
- Dynamic Programming Formulation
- Implementation
- Fractional Knapsack and Subset Sum Problem

Imdad ullah Khan

Knapsack Problem

Logistic problem involving transportation of freights

- A container/truck has a fixed maximum capacity
- Bunch of items each has a certain volume and a profit (return)
- Transporter would like to select items to maximize profit
  - ▷ Called the knapsack problem (named after the burglar's knapsack)



A classic optimization problem with many application in allocating space to items with certain volumes and values

Knapsack Problem

Input:

- Items: $U = \{a_1, \ldots, a_n\}$ ▷ Fixed order
- Weights: $w : U \to \mathbb{Z}^+$ ▷ $(w_1, \ldots, w_n)$
- Values: $v : U \to \mathbb{R}^+$ ▷ $(v_1, \ldots, v_n)$
- Capacity: $C \in \mathbb{R}^+$

Output:

- A subset $S \subset U$
- Capacity constraint:

$$\sum_{a_i \in S} w_i \leq C$$

- Objective: Maximize

$$\sum_{a_i \in S} v_i$$

Knapsack Problem

Input:

- Items: $U = \{a_1, \ldots, a_n\}$ (fixed order)
- Weights: $w : U \to \mathbb{Z}^+$: $w_1, \ldots, w_n$
- Values: $v : U \to \mathbb{R}^+$: $v_1, \ldots, v_n$
- Capacity: $C \in \mathbb{R}^+$

Output:

- A subset $S \subset U$
- Capacity constraint: $\sum_{a_i \in S} w_i \leq C$
- Objective: Maximize $\sum_{a_i \in S} v_i$

ID	weight	value
1	1	1
2	2	6
3	5	18
4	6	22
5	7	28
6	98	99

$C = 11$

- $\{1, 2\}$ weight 3, value 7
- $\{1, 2, 4\}$ weight 9, value 29
- $\{3, 4\}$ weight 11, value 40
- $\{4, 5\}$ weight 13, value 50

Knapsack Problem: Greedy Algorithms

Input:

- Items: $U = \{a_1, \ldots, a_n\}$ (fixed order)
- Weights: $w : U \to \mathbb{Z}^+$: $w_1, \ldots, w_n$
- Values: $v : U \to \mathbb{R}^+$: $v_1, \ldots, v_n$
- Capacity: $C \in \mathbb{R}^+$

Output:

- A subset $S \subset U$
- Capacity constraint: $\sum_{a_i \in S} w_i \leq C$
- Objective: Maximize $\sum_{a_i \in S} v_i$

Greedy Approach

- Select the most profitable item
- Add if it fits remaining capacity
- Repeat

ID	weight	value
1	51	51
2	50	50
3	50	50

$C = 100$

$\{1\}$ weight 51, value 51

Optimal $\{2, 3\}$ weight 100, value 100

Knapsack Problem: Greedy Algorithms

Input:

- Items: $U = \{a_1, \ldots, a_n\}$ (fixed order)
- Weights: $w : U \to \mathbb{Z}^+$: $w_1, \ldots, w_n$
- Values: $v : U \to \mathbb{R}^+$: $v_1, \ldots, v_n$
- Capacity: $C \in \mathbb{R}^+$

Output:

- A subset $S \subset U$
- Capacity constraint: $\sum_{a_i \in S} w_i \leq C$
- Objective: Maximize $\sum_{a_i \in S} v_i$

Greedy Approach

- Select the least weighted item
- Add if it fits remaining capacity
- Repeat

ID	weight	value
1	1	1
2	50	50
3	50	50

$C = 100$

$\{1, 2\}$ weight 51, value 51

Optimal $\{2, 3\}$ weight 100, value 100

Knapsack Problem: Greedy Algorithms

Input:

- Items: $U = \{a_1, \ldots, a_n\}$ (fixed order)
- Weights: $w : U \to \mathbb{Z}^+$: $w_1, \ldots, w_n$
- Values: $v : U \to \mathbb{R}^+$: $v_1, \ldots, v_n$
- Capacity: $C \in \mathbb{R}^+$

Output:

- A subset $S \subset U$
- Capacity constraint: $\sum_{a_i \in S} w_i \leq C$
- Objective: Maximize $\sum_{a_i \in S} v_i$

Greedy Approach

- Select item with highest v_i/w_i
- Add if it fits capacity
- Repeat

ID	weight	value	v_i/w_i
1	1	1	1
2	2	6	3
3	5	18	3.6
4	6	22	3.67
5	7	28	4
6	98	99	1.01

$C = 11$

$\{5, 2, 1\}$ weight 10, value 35

Optimal $\{3, 4\}$ weight 11, value 40