Lecture 5.
Knapsack problem

Algorithms and Data Structures
Ivan Solomatin
Harbour.Space@UTCC Bangkok

Outline

- Knapsack problem
  - Problem statement
  - DP solution
  - Answer restoration (argmax)
  - Another modifications

Knapsack problem

- **Problem statement**
- DP solution
- Answer restoration (argmax)
- Another modifications

Knapsack problem
Problem statement

Problem: Ali Baba 3 (classical knapsack problem):
Ali Baba returned back to cave, and took his knapsack with carrying capacity W to take treasures out of the cave. When he came into cave he found a room with N gold bars in it. For each bar he knows weight and cost. As usual, he wants to maximize total cost of treasures taken out. The bars are undividable.

More formally. You have a knapsack with carrying capacity W and N items with weight w_i and cost c_i, you need to choose several items with indices $i_0, \ldots, i_{K-1}$, such that:

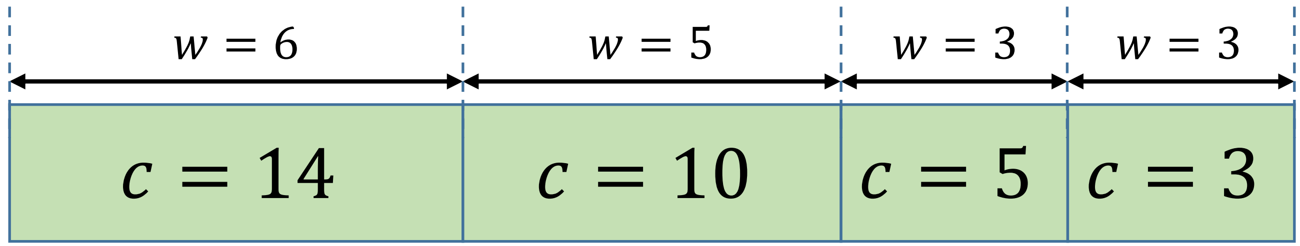
$$\sum_{k=0}^{K-1} w_{i_k} \le W, \qquad \sum_{k=0}^{K-1} c_{i_k} \to \max$$

This problem is NP-hard. In common case it cannot be solved faster than $O(2^N)$.
(of course, if $P \neq NP$)

But if w_i are integer and W is rather small, we can solve it in $O(NW)$ operations.

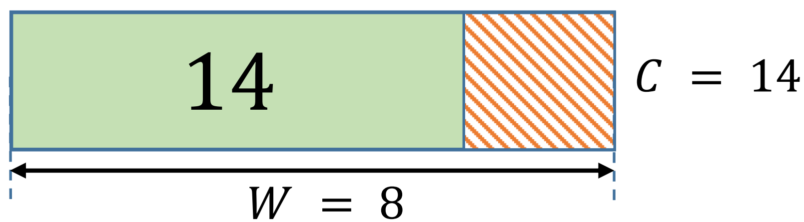
Knapsack problem

Problem statement

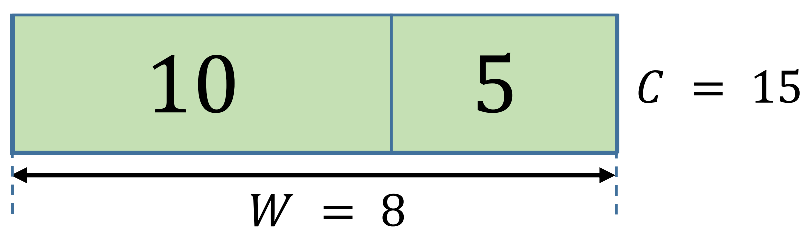


Greedy algorithm.

Items are sorted by $\frac{c}{w}$:



Optimal solution:



Knapsack problem

- Problem statement
- **DP solution**
- Answer restoration (argmax)
- Another modifications

Knapsack problem

DP solution

1. Subproblems:

   $d[i][w]$ – maximum total cost we can get using first i items and knapsack with capacity w

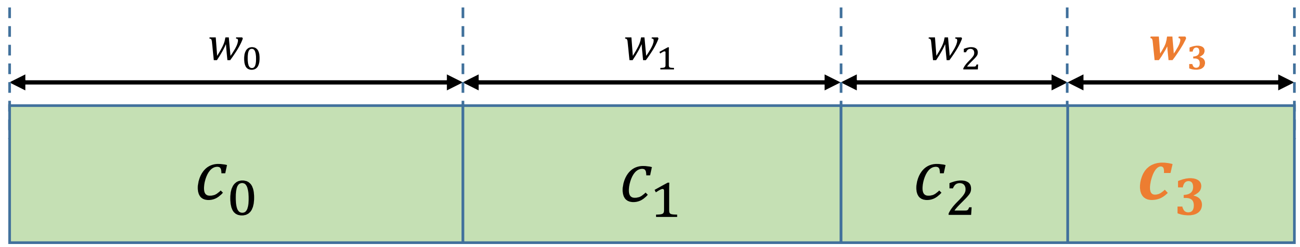
2. Basis:

   $d[:][0] = 0$ Let’s suppose that all weights are positive. Otherwise we can automatically take all items with $w \leq 0$, and solve problem for the rest items

   $d[0][:] = 0$ if we don’t have items, we can’t take any

Knapsack problem

DP solution



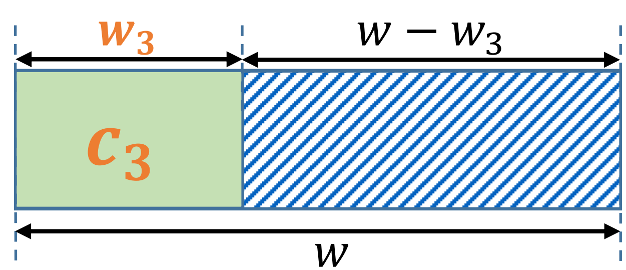
Let's assume that we have calculated all solutions for $i \leq 3$.
This means we know how we can optimally fill knapsacks with different capacities with first 3 items: $(0, 1, 2)$.
Now let's analyze how the solutions may change if we add one more item:

$$i = 4$$

To fill $d[4][:]$ we need to solve the subproblem for each capacity w.
Actually, for each capacity w, adding new item gives two possible cases:

- We ignore this new item. In this case, total cost is $d[3][w] = d[i-1][w]$.

- We place this new item into knapsack and fill free space with the rest of the items. Total cost in this case: $d[3][w - w_3] + c_3 =$
$= d[i-1][w - w_{i-1}] + c_{i-1}$



Knapsack problem

DP solution

1. Subproblems:

   $d[i][w]$ – maximum total cost we can get using first i items and knapsack with capacity w

2. Basis:

   $d[:][0] = 0$ Let's suppose that all weights are positive. Otherwise we can automatically take all items with $w \leq 0$, and solve problem for the rest items

   $d[0][:] = 0$ if we don't have items, we can't take any

3. Inductive step:

$$d[i][w] = \max \begin{Bmatrix} d[i-1][w] & \\ d[i-1][w - w_{i-1}] + c_{i-1}, & if\ w_{i-1} \leq w \end{Bmatrix}$$

4. Answer:

$$d[N][W]$$

Knapsack problem

DP solution

```
d = [[0] * (W + 1) for i in range(N + 1)]
for i in range(1, N + 1):
  for w in range(1, W + 1):
    if w_{i-1} > w:
      d[i][w] = d[i - 1][w]
    else:
      d[i][w] = max(d[i - 1][w],
                    d[i - 1][w - w_{i-1}] + c_{i-1})
```

	w_{i-1}	c_{i-1}
$i = 1$	3	3
$i = 2$	3	5
$i = 3$	5	10
$i = 4$	6	14

i \ w	0	1	2	3	4	5	6	7	8
$i = 0$	0	0	0	0	0	0	0	0	0
$i = 1$	0	0	0	3	3	3	3	3	3
$i = 2$	0	0	0	5	5	5	8	8	8
$i = 3$	0	0	0	5	5	10	10	10	15
$i = 4$	0	0	0	5	5	10	14	14	15

Knapsack problem

- Problem statement
- DP solution
- **Answer restoration (argmax)**
- Another modifications

Knapsack problem
Answer restoration (argmax)

i \ w	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
$i = 0$	0	0	0	0	0	0	0	0	0
$i = 1$	0	0	0	3	3	3	3	3	3
$i = 2$	0	0	0	5	5	5	8	8	8
$i = 3$	0	0	0	5	5	10	10	10	15
$i = 4$	0	0	0	5	5	10	14	14	**15**

Objects in optimal solution: 3 , 2

Knapsack problem

- Problem statement
- DP solution
- Answer restoration (argmax)
- **Another modifications**

Knapsack problem

Another problem modifications

Problem 1. (Ali-Baba 1). Given N items with different costs and knapsack can contain up to W items. Maximize total cost.

$$w_i = 1$$

This problem can be solved using greedy algorithm.

Problem 2. Given N items with different weights and knapsack with capacity W. Maximize number of taken elements:

$$c_i = 1$$

This problem can be solved using greedy algorithm.

Problem 3. We can use each item (c_i, w_i) any number of times:

$$d[:] = 0$$

$$d[w] = \max_{w_i \le w}(c_i + d[w - w_i])$$

Knapsack problem

Another problem modifications

Problem 4. Does a given number W equal a sum of any subset of numbers $w_0, w_1, \ldots, w_{N-1}$?

We don't need costs here.

Subproblems:

$d[i][w] -$ does w equal a sum of any subset of $w_0, \ldots, w_{i-1}$

Basis:

$$d[:][0] = True, \qquad d[0][1:] = False$$

Inductive step:

$$d[i][w] = \begin{cases} d[i-1][w], & if\ w < w_{i-1} \\ d[i-1][w]\ OR\ d[i-1][w - w_{i-1}], & if\ w \geq w_{i-1} \end{cases}$$

Answer:

$$d[N][W]$$

Knapsack problem

Another problem modifications

```
d = [[1] + [0] * (W) for i in range(N + 1)]
for i in range(1, N + 1):
  for w in range(1, W + 1):
    if w_{i-1} > w:
      d[i][w] = d[i - 1][w]
    else:
      d[i][w] = (d[i - 1][w] or
                 d[i - 1][w - w_{i-1}])
```

	w_{i-1}
$i = 1$	3
$i = 2$	3
$i = 3$	5
$i = 4$	6

i \ w	0	1	2	3	4	5	6	7	8
$i = 0$	+	−	−	−	−	−	−	−	−
$i = 1$	+	−	−	+	−	−	−	−	−
$i = 2$	+	−	−	+	−	−	+	−	−
$i = 3$	+	−	−	+	−	+	+	−	+
$i = 4$	+	−	−	+	−	+	+	−	+

We can obtain: $0, 3, 5, 6, 8, \ldots$

Knapsack problem
Another problem modifications

Problem 5. Given number W and numbers $w_0, w_1, \dots, w_{N-1}$. Find a closest to W number W^*, which is a sum of a subset of $w_0, w_1, \dots, w_{N-1}$.

Same as **Problem 4:**

Subproblems:

$d[i][w] -$ does w equal a sum of any subset of $w_0, \dots, w_{i-1}$

Basis:

$$d[:][0] = True, \qquad d[0][1:] = False$$

Inductive step:

$$d[i][w] = \begin{cases} d[i-1][w], & if\ w < w_{i-1} \\ d[i-1][w]\ OR\ d[i-1][w-w_{i-1}], & if\ w \geq w_{i-1} \end{cases}$$

Answer:

$$W^*: \qquad d[N][W^*] = True, \qquad |W - W^*| \to min$$

Conclusion

Thank you for watching!