

Ex 3: Knapsack

Suppose we want to add item n to some solution for the first $n-1$ items. What do we need to know?

- (1) value of that solution
- (2) weight of that solution

solution on $n-1$ items	add item n
weight	weight
value	value
don't need to know the actual solution	

Consider items $I_1, \dots, I_n$. values $v_i > 0$
weights $w_i > 0$
total capacity C .

Let $O(j, w)$ denote the optimal score on items $I_1, \dots, I_j$, using a maximum capacity of w .

Want: $O(n, C)$

If item n is not in an optimal solution:

$$O(n, C) = O(n-1, C)$$

If item n is in an opt. sol!

$$O(n, C) = v_n + O(n-1, C - w_n)$$

As a recurrence:

$$O(j, w) = \begin{cases} 0, & j=0 \\ O(j-1, w), & w_j > w \\ \max(O(j-1, w), v_j + O(j-1, w - w_j)), & \text{otherwise} \end{cases}$$

$$O(j, 10)$$

item j

weight 20

value 100

$$O(j-1, 10)$$

$$100 + O(j-1, -10)$$

Run time:

What is getting memoized?

All pairs (j, w) where $1 \leq j \leq n$, $0 \leq w \leq C$

Total # of calls : $n \cdot C$

$$O(n \cdot C)$$

Algo. returns optimal score $\rightarrow$ we want actual solution.

Trace through the memo dict.

look at $\text{memo}[n, C]$
 $= 0$? Solution is empty.

$= \text{memo}[n-1, C]$?
If so, solution does not contain item n .

$= v_n + \text{memo}[n-1, C - w_n]$?
If so, solution does contain item n .

Ex #4: TSP - there is a D.P. algo.

$$O(n!)$$

Speed: $O(n^2 \cdot 2^n)$.

Topic 10 - Object-Oriented Programming