

ナップサック問題のダイナミックプログラミングによる解法

Solving the Knapsack Problem by Dynamic Programming

Problem real numbers $a[1], a[2], \dots, a[n], w[1], w[2], \dots, w[n], b$ given
 $a[i]$ are profit of item i , $w[i]$ are size, and b is the size of knapsack. Let I be a subset of $\{1, 2, \dots, n\}$. Find I that gives maximum to

$$\text{Max } \sum_{i \in I} a[i] \quad \text{subject to } \sum_{i \in I} w[i] \leq b$$

We maximize the total profit while the total size is limited by b .

Let $f[i][j]$ be the solution of the above problem with $n=i$ and $b=j$, that is, $f[i][j]$ is the solution for the problem of size i and j . That is, we use items $1, \dots, i$, and the the limit of the knapsack is j . We compute $f[n][b]$ by computing $f[i][j]$ from smaller values of i and j step by step based on the following principle of optimality.

$$f[0][j] = 0 \quad (j=1, \dots, b), \quad f[i][j]=0 \text{ for } j=0, \text{ and } f[i][j]=-99 \text{ (minus infinity) for } j<0$$

$$f[i][j] = \max\{f[i-1][j], f[i-1][j-w[i]]+a[i]\}$$

If we pack item i for $f[i][j]$, the optimal packing of items up to $i-1$ must give the optimal solution for the knapsack of limit $j-w[i]$. If item i is not packed, $f[i][j]$ must be the optimal solution for the problem for items up to $i-1$, and the limit j , that is, $f[i-1][j]$. We take the maximum of these two. Time is $O(bn)$.

Example. $a = (10, 6, 12, 9, 29)$, $w = (3, 7, 2, 2, 3)$, $b = 8$. Greedy gives 50.

	a	j	0	1	2	3	4	5	6	7	8
i											
0			0	0	0	0	0	0	0	0	0
1	10	3	0	0	0	10	10	10	10	10	10
2	6	7	0	0	0	10	10	10	10	10	10
3	12	2	0	0	12	12	12	22	22	22	22
4	9	2	0	0	12	12	21	22	22	31	31
5	29	3	0	0	12	29	29	41	41	50	51

In the above table, we trace the bold face characters from the bottom right corner, that is, 51, the solution. As the weight of item 5 is 3, we reduce the limit of knapsack by 3, that is, $51 = \max\{f[4][8], f[4][5]+29\} = \max\{31, 51\}$. By similar trace, we know items 1, 3, and 5 are used for the optimal solution.