Advanced Algorithms & Data Structures

Department of CSE

ADVANCED ALGORITHMS AND DATA STRUCTURES
23CS03HF

Topic:

Dynamic Programming – 0/1 Knapsack Problem

AIM OF THE SESSION

To familiarize students with the concept of Knapsack problem.

INSTRUCTIONAL OBJECTIVES

This Session is designed to:

1. Demonstrate :- Knapsack problem.
2. Describe :- Bounding function.

LEARNING OUTCOMES

At the end of this session, you should be able to:

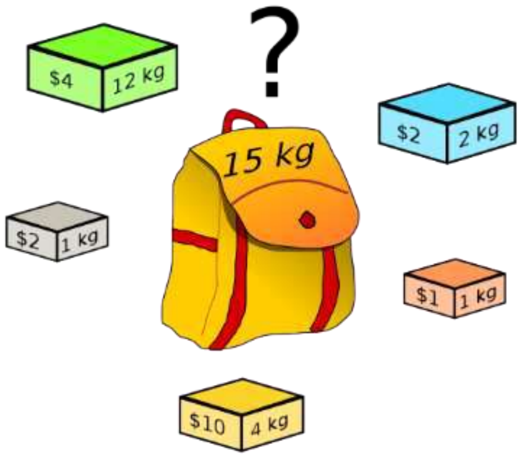
1. Define :- Knapsack problem.
2. Describe :- Bounding function.
3. Summarize:- Backtracking solution to the 0/1 knapsack problemming.

Knapsack Problem

You are given the following-

– A knapsack - with limited weight capacity.

– items - having some weight and value.



The problem states-

Which items should be placed into the knapsack such that-

- The value or profit obtained by putting the items into the knapsack is maximum.
- And the weight limit of the knapsack does not exceed.

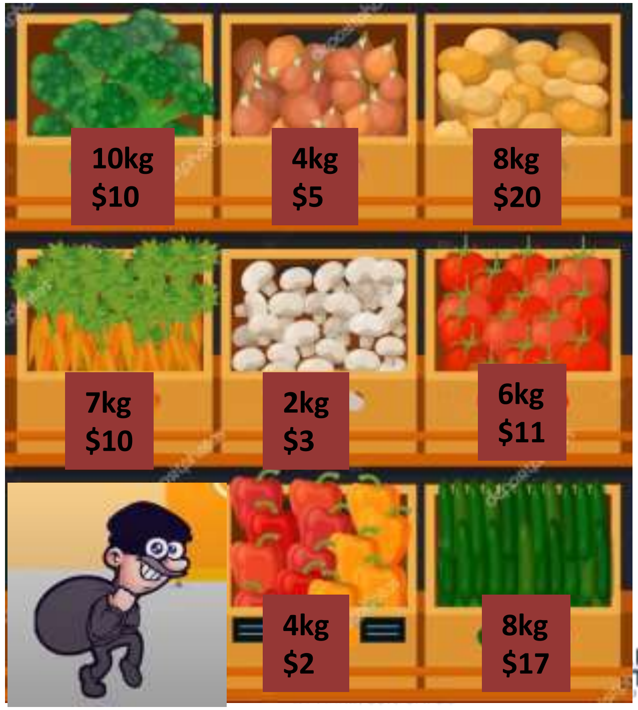
Knapsack Problem variants

Variants:

- **0/1 Knapsack.**
  - ➢ not allowed to break items. We either take the whole item or don't take it.

Fractional Knapsack

- ➢ can break items for maximizing the total value of knapsack



0/1 Knapsack Problem

In 0/1 Knapsack Problem,

- **As the name suggests, items are indivisible here.**
- **We can not take the fraction of any item.**
- **We have to either take an item completely or leave it completely.**

Example:

Consider the knapsack instance n = 3, $(w_1, w_2, w_3) = (2, 3, 4)$, $(p_1, p_2, p_3) = (1, 2, 5)$ and m = 6.

Probability of Chosen Items (x_i) = [{0, 0, 0}, {0, 0, 1}, ……, {1, 1, 1}]

No. of Possible Solutions $(2^n) = 2^3 = 8$.

The problem is to find the Best Optimal Solution among the 8 for the 0/1 Knapsack.

0/1 Knapsack Problem-solution: Set Method

Initially $S^0 = \{(0,0)\}$

$$S_1^i = \{ (P,W) / (P - p_{i+1}, W - w_{i+1}) \in S^i \}$$

S^{i+1} can be computed by merging from S^i and S_1^i

- **Purging or dominance rule: if S^{i+1} contains two pairs (p_j, w_j) and (p_k, w_k) with the property that $p_j \leq p_k$ and $w_j \geq w_k$ then the pair (p_j, w_j) can be discarded.**
- **When generating S^i, we can also purge all pairs *(p, w)* with *w > m* as these pairs determine the value of $f_n(x)$ only for *x > m.***
- **The optimal solution $f_n(m)$ is given by the *highest profit pair.***

Example 1: Consider the knapsack instance $n = 3$,
$(w_1,w_2,w_3) = (2,3,4)$, $(p_1,p_2,p_3) = (1,2,5)$, and $m = 6$.

Initially

$S^0 = \{(0,0)\}$

Include 1st object

$S_1^0 = \{ (0+1,0+2) \} = \{ (1,2) \}$

Next Stage can be obtained S^{0+1} (S^1)can be computed by merging from S^0 and S_1^0

$S^1 = \{(0,0)(1,2)\}$

Include 2nd object

$S_1^1 = \{ (0+2,0+3) (1+2, 2+3) \} = \{ (2,3) (3,5) \}$

Next Stage can be obtained S^{1+1} (S^2)can be computed by merging from S^1 and S_1^1

$S^2 = \{(0,0)(1,2)(2,3)(3,5)\}$

Include 3rd object

$S_1^2 = \{ (0+5,0+4) (1+5, 2+4)(2+5,3+4) (3+5,5+4) \}$

$= \{(5,4)(6,6)(7,7)(8,9)\}$

Next Stage can be obtained S^{2+1} (S^3)can be computed by merging from S^2 and S_1^2

S^3 ={(0,0)(1,2)(2,3)(3,5) (5,4)(6,6)(7,7)(8,9)}

Apply Purging rule

Pairs(3,5) (7,7)(8,9) will be discarded

Therefore ,

S^3 ={(0,0)(1,2)(2,3)(5,4)(6,6)}

To find the included items:

(6,6) is in S^3 So $x_3 = 1$

$P-p_3= 6-5 =1$ & $W - w_3 = 6-4=2$

(1,2) is in S^1 So $x_1=1$

Therefore X=(1,0,1)

0/1 Knapsack Algorithm

Algorithm DKP(p,w,n,m)

{

S^0 := {(0,0)};

for i := 1 to n-1 do

{

S^{i-1} ={(P,W)|(P-p_i,W-w_i) ϵ S^{i-1} and W <= m);

S^i = MergePurge(S^{i-1} , S_1^{i-1});

}

(P_X,W_X) = last pair in S^{n-1} ;

(P_Y,W_Y)=(P^1+ p_n ,W^1+w_n) where W^1 is the largest W in any pair in S^{n-1} such that W+ w_n <= m;

// Trace back for x_n ,x_{n-1},.....,$x_{1.}$

if ($P_X > P_Y$) then

$x_n = 0$;

else x_n=1;

TraceBackFor(x_{n-1},.....,x_1);

}

Complexity Analysis:

The complexity of the algorithm is O(nW) where, n is the number of items and W is the capacity of knapsack.

- **Example 2** *Solve Knapsack instance $M = 8$, and $n = 4$. Let P_i and W_i are as shown below.*

i	P_i	W_i
1	1	2
2	2	3
3	5	4
4	6	5

Solution : Let us build the sequence of decision S^0, S^1, S^2.

$$S^0 = \{(0, 0)\} \text{ initially}$$

$$S_1^0 = \{(1, 2)\}$$

That means while building S_1^0 we select the next i^{th} pair. For S_1^0 we have selected first (P, W) pair which is (1, 2).

Now

$$S^1 = \{\text{Merge } S^0 \text{ and } S_0^1\}$$

$$= \{(0, 0), (1, 2)\}$$

$$S_1^1 = \{\text{ Select next (P, W) pair and add it with } S^1 \}$$

$$= \{ (2, 3), (2+0, 3+0), (2+1, 3+2) \}$$

$S_1^1 = \{(2, 3), (3, 5)\}$ $\because$ Repetition of (2, 3) is avoided.

$S^2 = \{\text{Merge candidates from } S^1 \text{ and } S_1^1\}$

$= \{(0, 0), (1, 2), (2, 3), (3, 5)\}$

$\therefore$ $S_1^2 = \{\text{Select next (P, W) pair and add it with } S^2\}$

$= \{(5, 4), (6, 6), (7, 7), (8, 9)\}$

Now $S^3 = \{\text{Merge candidates from } S^2 \text{ and } S_1^2\}$

$S^3 = \{(0, 0), (1, 2), (2, 3), (5, 4), (6, 6), (7, 7), (8, 9)\}$

Note that the pair (3, 5) is purged from S^3. This is because, let us assume $(P_j, W_j) = (3, 5)$ and $(P_k, W_k) = (5, 4)$. Here $P_j \leq P_k$ and $W_j > W_k$ is true hence we will eliminate pair (P_j, W_j) i.e. (3, 5) from S^3.

$S_1^3 = \{(6, 5), (7, 7), (8, 8), (11, 9), (12, 11), (13, 12), (14, 14)\}$

$S^4 = \{(0,0), (1,2), (2,3), (5,4), (6,6), (7,7), (8,9)$
$(6,5), (8,8), (11,9), (12,11), (13,12), (14,14)\}$

Now we are interested in M = 8. We get pair (8, 8) in S^4. Hence we will set $x_4 = 1$. Now to select next object $(P - P_4)$ and $(W - W_4)$.

i.e $(8 - 6)$ and $(8 - 5)$.

i.e. (2, 3)

Pair $(2, 3) \in S^2$. Hence set $x_2 = 1$. So we get the final solution as (0, 1, 0, 1).

Example 3:

Consider the knapsack instance $n = 3$, $m=50$, $(w_1, w_2, w_3) = (10,20,30)$ & $(p_1, p_2, p_3) = (60, 100, 120)$.

SUMMARY

The 0/1 Knapsack problem can be defined as follows: We have (N) items, each with a weight ((w_i)) and a value ((v_i)). We also have a bag with a capacity (W). The goal is to select items to place in the bag such that the total value is maximized, without exceeding the bag's capacity.

SELF-ASSESSMENT QUESTIONS

1. What is the main objective of the 01 Knapsack problem?

A. To maximize the total weight of items in the knapsack

B. To minimize the total value of items in the knapsack

C. To maximize the total value of items in the knapsack without exceeding the weight capacity

D. To minimize the total weight of items in the knapsack

In the 0/1 Knapsack problem, if (n) is the number of items and (W) is the maximum weight capacity, what is the time complexity of the dynamic programming solution?

A. (O(n + W))

B. (O(nW))

C. (O(n^2))

D. (O(n^3))

TERMINAL QUESTIONS

1. **What is the dynamic 0/1 knapsack problem?**
2. **Can we solve knapsack problem using dynamic programming?**
3. **What is the formula for the 0/1 knapsack problem?**

REFERENCES FOR FURTHER LEARNING OF THE SESSION

Reference Books :

1. Introduction to Algorithms, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein., 3rd, 2009, The MIT Press.
2 Algorithm Design Manual, Steven S. Skiena., 2nd, 2008, Springer.
3 Data Structures and Algorithms in Python, Michael T. Goodrich, Roberto Tamassia, and Michael H. Goldwasser., 2nd, 2013, Wiley.
4 The Art of Computer Programming, Donald E. Knuth, 3rd, 1997, Addison-Wesley Professiona.

MOOCS :

1. https://www.coursera.org/specializations/algorithms?=
2.https://www.coursera.org/learn/dynamic-programming-greedy-algorithms#modules

THANK YOU