Design and Analysis of Algorithms

Lecture – 21

Dynamic Programming – Knapsack Problem

Success is always inevitable with Hard Work and Perseverance

N. Ravitha Rajalakshmi

Learning Objective

- Derive optimal solution for knapsack (unbounded and bounded variants) using dynamic programming

(Only solutions using Tabulation are discussed)

Knapsack Problem with repetitions (Unbounded)

- Find the optimal solution for filling a knapsack of weight W with n items whose weights [weight[]] and values[cost[]] are known

- Similar to Coin denomination, items are unlimited, any number of copies of an item can be used to fill the knapsack

Pause and Think

- What is the optimal solution for a knapsack problem?

  Maximize the total cost of items used to fill the knapsack

- How the problem instance is defined?

  Using the weight of knapsack

Knapsack Problem with repetitions

- To fill the knapsack with weight W value(W)

First item can be selected cost[0] + value(W- weight[0])

Second item can be selected cost[1] + value (W-weight[1])

……..

soon

$$\text{value(W)} = \max_{0 \le \text{i} \le n}(\text{value}(\text{W} - weight[\text{i}]) + cost[i])$$

Pause & Think

- What does value(5) indicate?

  Optimal value obtained by filling a knapsack with capacity 5

- Value(0) ?

  It indicates knapsack is empty

  Value(0) = 0

Bottom up Programming

Function TabulationKnapRep(W, weights, cost, n)

```
val [0] = 0
for  i  from 1 to W:
        for j from 0 to n:
                if(w[j]<=i){
                        c = cost[j] + val[i-w[j]]
                        if(c>val[i]){
                                val[i] = c
                            }
                        }

return value[W]
```

Time Complexity
Basic operation – Comparison

O(W*n)

Problem

Consider W = 5 and number of items n = 4

Items	1	2	3	4
Weight	1	3	4	5
Cost	10	30	50	60

$$\text{value(W)} = \max_{0 \le i \le n}(\text{value}(W - weight[\text{i}]) + cost[i])$$

In bottom up dynamic programming, all instances will be solved

W	0	1	2	3	4	5
Value	0					
Optimal Solution	{}					

Problem

Consider W = 5 and number of items n = 4

Items	1	2	3	4
Weight	1	3	4	5
Cost	10	30	50	60

W	0	1	2	3	4	5
Value	0					
Optimal Solution	{}					

Pick Item 1 Yes

Value[1] = Value[1 – 1] + 10
= 0 + 10
{item1}

Pick Item 2 No
Pick Item 3 No
Pick Item 4 No

Problem

Consider W = 5 and number of items n = 4

Items	1	2	3	4
Weight	1	3	4	5
Cost	10	30	50	60

W	0	1	2	3	4	5
Value	0	10				
Optimal Solution	{}	{1}				

Pick Item 1 Yes

Value[2] = Value[2 - 1] + 10
= Value[1]+ 10
= 10 + 10 = 20
{item1, item1}

Pick Item 2 No
Pick Item 3 No
Pick Item 4 No

Problem

Consider W = 5 and number of items n = 4

Items	1	2	3	4
Weight	1	3	4	5
Cost	10	30	50	60

W	0	1	2	3	4	5
Value	0	10	20			
Optimal Solution	{}	{1}	{1,1}			

Pick Item 1 Yes

Value[3] = Value[3 - 1] + 10
= Value[2]+ 10
= 20 + 10 = 30
{item1, item1, item1}

Pick Item 2 Yes
Value[3] = Value[3-3] + 30
= 0 + 30 = 30
{item 2}

Pick Item 3 No
Pick Item 4 No

Problem

Consider W = 5 and number of items n = 4

Items	1	2	3	4
Weight	1	3	4	5
Cost	10	30	50	60

W	0	1	2	3	4	5
Value	0	10	20	30		
Optimal Solution	{}	{1}	{1,1}	{2}		

Pick Item 1 Yes

Value[4] = Value[4 – 1] + 10
= Value[3]+ 10
= 30 + 10 = 40
{item2, item1}

Pick Item 2 Yes
Value[4] = Value[4-3] + 30
= 10 + 30 = 40
{item1, item 2}

Pick Item 3 Yes
Value[4] = Value[4-4] + 30
= 0 + 50 = 50
{item 3}

Pick Item 4 No

Problem

Consider W = 5 and number of items n = 4

Items	1	2	3	4
Weight	1	3	4	5
Cost	10	30	50	60

W	0	1	2	3	4	5
Value	0	10	20	30	50	
Optimal Solution	{}	{1}	{1,1}	{2}	{3}	

Pick Item 1 Yes

Value[5] = Value[5 – 1] + 10
= Value[4]+ 10
= 50 + 10 = 60
{item3, item1}

Pick Item 2 Yes
Value[5] = Value[5-3] + 30
= 20 + 30 = 50
{item1,item1, item 2}

Pick Item 3 Yes
Value[5] = Value[5-4] + 50
= 10 + 50 = 60
{item 3, item1}

Pick Item 4 Yes
Value[5] = Value[5-5] + 60
= 0 + 60 = 60
{item4}

Problem

Consider W = 5 and number of items n = 4

Items	1	2	3	4
Weight	1	3	4	5
Cost	10	30	50	60

W	0	1	2	3	4	5
Value	0	10	20	30	50	60
Optimal Solution	{}	{1}	{1,1}	{2}	{3}	{4}

Knapsack Problem without repetitions

- Find the optimal solution for filling a knapsack of weight W with n items whose weights [weight[]] and values[cost[]] are known

- Here, an item cannot be selected multiple items. Only one instance of an item is available.

Knapsack Problem without repetitions

$$\text{value(W)} = \max_{0 \le \text{i} \le n} (\text{value}(\text{W} - weight[\text{i}]) + cost[i])$$

In the previous case, Optimization solution considers all items

Now, we need to keep track of items already selected for filling the bag

Rephrase the solution to include the item that is considered to fill the knapsack

Value(i , W) denote the value obtained by filling a knapsack of weight W considering an items from 1 to i

Optimization Solution

$$
\text{Value}(i, W) = \max \begin{cases} \text{Value}(i-1, W) \\ \text{(item not considered, fill it with remaining items)} \\ \\ \text{Value}(i-1, W-w[i]) + \text{cost}[i] \quad \text{if } w[i] \le W \\ \text{(item considered, remaining weight filled} \\ \text{optimally with items from 1 to } i-1\text{)} \end{cases}
$$

- Every possible weight of knapsack and every possible item to be considered

Items	1	2	3	4
Weight	1	3	4	5
Cost	10	30	50	60

Knapsack weight is empty

No items

Item/ Weight	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0					
2	0					
3	0					
4	0					

Items	1	2	3	4
Weight	1	3	4	5
Cost	10	30	50	60

Consider item1

Item/ Weight	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	10 {1}	10 {1}	10 {1}	10 {1}	10 {1}
2	0					
3	0					
4	0					

Cost [1,2] = max(10 + cost [0,1], cost[0,1])

= 10

Items	1	2	3	4
Weight	1	3	4	5
Cost	10	30	50	60

Consider item2

Item/ Weight	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	10 {1}	10 {1}	10 {1}	10 {1}	10 {1}
2	0	10 {1}	10 {1}	30 {2}	40 {1,2}	40 {1,2}
3	0	10 {1}	10 {1}	30 {2}	50 {3}	60 {1,3}
4	0	10 {1}	10 {1}	30 {2}	50 {3}	60 {4}

Summary

- Discussed about dynamic programming solution for knapsack problem.

Thank You
Happy Learning

Success is always inevitable with Hard Work and Perseverance