AOA-Exp 8

Name: Meet Raut

Roll No: 84
Batch: S21

Aim:
To Study and implement 0/1 knapsack problem using dynamic programming approach

Theory:
Given N items where each item has some weight and profit associated with it and also given a bag with capacity W, i.e. the bag can hold at most W weight in it. The task is to put the items into the bag such that the sum of profits associated with them is the maximum possible.
The 0/1 knapsack problem means that the items are either completely or no items are filled in a knapsack. For example, we have two items having weights 2kg and 3kg, respectively. If we pick the 2kg item then we cannot pick 1kg item from the 2kg item (item is not divisible); we have to pick the 2kg item completely. This is a 0/1 knapsack problem in which either we pick the item completely or we will pick that item. The 0/1 knapsack problem is solved by the dynamic programming.

Algorithm:
Follow the below steps to solve the problem:
The maximum value obtained from 'N' items is the max of the following two values.
- Case 1 (include the Nth item): Value of the Nth item plus maximum value obtained by remaining N-1 items and remaining weight i.e. (W-weight of the Nth item).
- Case 2 (exclude the Nth item): Maximum value obtained by N-1 items and W weight.
- If the weight of the 'Nth'item is greater than 'W', then the Nth item cannot be included and Case 2 is the only possibility.

Example

Let us consider that the capacity of the knapsack is W = 8 and the items are as shown in the following table.

Item	**A**	**B**	**C**	**D**
Profit	2	4	7	10
Weight	1	3	5	7

AOA-Exp 8

Step 1

Construct an adjacency table with maximum weight of knapsack as rows and items with respective weights and profits as columns.

Values to be stored in the table are cumulative profits of the items whose weights do not exceed the maximum weight of the knapsack (designated values of each row)

So we add zeroes to the 0^{th} row and 0^{th} column because if the weight of item is 0, then it weighs nothing; if the maximum weight of knapsack is 0, then no item can be added into the knapsack.

Maximum Weights →

Items with Weights and Profits ↓

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1 (1, 2)	0								
2 (3,4)	0								
3 (5,7)	0								
4 (7,10)	0								

The remaining values are filled with the maximum profit achievable with respect to the items and weight per column that can be stored in the knapsack.

The formula to store the profit values is –

$c[i,w]=\max\{c[i-1,w-w[i]]+P[i]\}$

By computing all the values using the formula, the table obtained would be −

AOA-Exp 8

Maximum Weights →

Items with Weights and Profits ↓

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1 (1, 2)	0	1	1	1	1	1	1	1	1
2 (3,4)	0	1	1	4	6	6	6	6	6
3 (5,7)	0	1	1	4	6	7	9	9	11
4 (7,10)	0	1	1	4	6	7	9	10	12

To find the items to be added in the knapsack, recognize the maximum profit from the table and identify the items that make up the profit, in this example, its {1, 7}.

Maximum Weights →

Items with Weights and Profits ↓

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1 (1, 2)	0	1	1	1	1	1	1	1	1
2 (3,4)	0	1	1	4	6	6	6	6	6
3 (5,7)	0	1	1	4	6	7	9	9	11
4 (7,10)	0	1	1	4	6	7	9	10	(12)

The optimal solution is {1, 7} with the maximum profit is 12.

AOA-Exp 8

Code:

```c
#include <stdio.h>


int max(int a, int b) { return (a > b) ? a : b; }


int knapSack(int W, int wt[], int val[], int n)
{

    if (n == 0 || W == 0)
        return 0;

    if (wt[n - 1] > W)
        return knapSack(W, wt, val, n - 1);


    else
        return max(
            val[n - 1] + knapSack(W - wt[n - 1], wt, val, n - 1),
            knapSack(W, wt, val, n - 1));
}


int main()
{
    int n, W;
    printf("Enter the number of items: ");
    scanf("%d", &n);

    int profit[n], weight[n];
    printf("Enter the profits of the items:\n");
    for (int i = 0; i < n; ++i) {
        scanf("%d", &profit[i]);
    }

    printf("Enter the weights of the items:\n");
    for (int i = 0; i < n; ++i) {
        scanf("%d", &weight[i]);
    }

    printf("Enter the capacity of the knapsack: ");
    scanf("%d", &W);

    printf("Maximum value in knapsack: %d", knapSack(W, weight, profit, n));
    return 0;
}
```

AOA-Exp 8

Output:

```
Enter the number of items: 5
Enter the values of the items:
3
4
2
1
5
Enter the weights of the items:
6
4
3
5
8
Enter the capacity of the knapsack: 10
Maximum value in knapsack: 7
```