0-1 Knapsack Problem

Dynamic Programming

Share

(https://www.facebook.com/sharer.php?u=https://www.dyclassroom.com/dynamic-programming/0-1-knapsack-problem) (https://twitter.com/home?status=Check this out! https://www.dyclassroom.com/dynamic-programming/0-1-knapsack-problem) (https://plus.google.com/share?url=https://www.dyclassroom.com/dynamic-programming/0-1-knapsack-problem) (https://www.linkedin.com/cws/share?url=https://www.dyclassroom.com/dynamic-programming/0-1-knapsack-problem) (https://www.pinterest.com/pin/create/button/?url=https://www.dyclassroom.com/dynamic-programming/0-1-knapsack-problem&media=https%3A%2F%2Fwww.dyclassroom.com%2Fimage%2Fdyclassroom-logo-black-512x512.png&description=check%20this%20out) (https://www.tumblr.com/share?t=Check%20this%20out&v=3&u=https://www.dyclassroom.com/dynamic-programming/0-1-knapsack-problem) (https://reddit.com/submit?title=Check%20%this%20out&url=https://www.dyclassroom.com/dynamic-programming/0-1-knapsack-problem) (mailto:?Subject=Check%20this%20out%20from%20dyclassroom!&Body=Hello,%0A%0ACheck%20this%20out%20from%20programming/0-1-knapsack-problem%0A%0AThanks)

Next → (/dynamic-programming/coin-changing-problem)

Dynamic Programming | 0-1 Knapsack Problem - step by step guide

In this tutorial we will be learning about 0-1 Knapsack problem. In this dynamic programming problem we have n items each with an associated weight and value (benefit or profit). The objective is to fill the knapsack with items such that we have a maximum profit without crossing the weight limit of the knapsack. Since this is a 0-1 knapsack problem hence we can either take an entire item or reject it completely. We can not break an item and fill the knapsack.

Point to remember

- In this problem we have a Knapsack that has a weight limit W.
- There are items i1, i2, ..., in each having weight w1, w2, … wn and some benefit (value or profit) associated with it v1, v2, ... vn
- Our objective is to maximise the benefit such that the total weight inside the knapsack is at most W.
- Since this is a 0-1 Knapsack problem so we can either take an entire item or reject it completely. We can not break an item and fill the knapsack.

Problem

Assume that we have a knapsack with max weight capacity W = 5
Our objective is to fill the knapsack with items such that the benefit (value or profit) is maximum.

Following table contains the items along with their value and weight.

item i	1	2	3	4
value val	100	20	60	40
weight wt	3	2	4	1

Total items n = 4

ere, i denotes number of items and w denotes the

lenote the weight.

vs from 0 to 4.

s W = 5 so, we have 6 columns from 0 to 5

	1	2	3	4	5

eans when 0 item is considered weight is 0.

0. This means when weight is 0 then items

```
if wt[i] > w then
V[i,w] = V[i-1,w]

else if wt[i] <= w then
V[i,w] = max( V[i-1,w], val[i] + V[i-1, w - wt[i]] )
```

After calculation, the value table V

V[i,w]	w = 0	1	2	3	4	5
i = 0	0	0	0	0	0	0
1	0	0	0	100	100	100
2	0	0	20	100	100	120
3	0	0	20	100	100	120
4	0	40	40	100	140	140

Maximum value earned
Max Value = V[n,W]
= V[4,5]
= 140

```
Items that were put inside the knapsack
are found using the following rule

set i = n and w = W

while i and w > 0 do
  if V[i,w] != V[i-1,w] then
    mark the ith item
    set w = w - wt[i]
    set i = i - 1
  else
    set i = i - 1
  endif
endwhile
```

So, items we are putting inside the knapsack are 4 and 1.

C Code

```
#include<stdio.h>

void knapSack(int W, int n, int val[], int wt[]);
int getMax(int x, int y);

int main(void) {

        //the first element is set to -1 as
        //we are storing item from index 1
        //in val[] and wt[] array
        int val[] = {-1, 100, 20, 60, 40};      //value of the items
        int wt[] = {-1, 3, 2, 4, 1};            //weight of the items

        int n = 4;      //total items
        int W = 5;      //capacity of knapsack

        knapSack(W, n, val, wt);

        return 0;
}

int getMax(int x, int y) {
        if(x > y) {
                return x;
        } else {
                return y;
        }
}

void knapSack(int W, int n, int val[], int wt[]) {
        int i, w;

        //value table having n+1 rows and W+1 columns
        int V[n+1][W+1];

        //fill the row i=0 with value 0
        for(w = 0; w <= W; w++) {
```

```
                V[0][w] = 0;
        }

        //fille the column w=0 with value 0
        for(i = 0; i <= n; i++) {
                V[i][0] = 0;
        }

        //fill the value table
        for(i = 1; i <= n; i++) {
                for(w = 1; w <= W; w++) {
                        if(wt[i] <= w) {
                                V[i][w] = getMax(V[i-1][w], val[i] + V
                        } else {
                                V[i][w] = V[i-1][w];
                        }
                }
        }

        //max value that can be put inside the knapsack
        printf("Max Value: %d\n", V[n][W]);
}
```

Time complexity

Time complexity of 0-1 Knapsack problem is O(nW) where, n is the number of items and W is the capacity of knapsack.

Next → (/dynamic-programming/coin-changing-problem)

programming/0-1-knapsack-problem) (https://www.pinterest.com/pin/create/button/?
url=https://www.dyclassroom.com/dynamic-programming/0-1-knapsack-
problem&media=https%3A%2F%2Fwww.dyclassroom.com%2Fimage%2Fdyclassroom-logo-black-
512x512.png&description=check%20this%20out) (https://www.tumblr.com/share?
t=Check%20this%20out&v=3&u=https://www.dyclassroom.com/dynamic-programming/0-1-knapsack-problem)
(https://reddit.com/submit?title=Check%20%this%20out&url=https://www.dyclassroom.com/dynamic-
programming/0-1-knapsack-problem) (mailto:?
Subject=Check%20this%20out%20from%20dyclassroom!&Body=Hello,%0A%0ACheck%20this%20out%20from%20dyc
programming/0-1-knapsack-problem%0A%0AThanks)

Recently Added

C - Structure in Structure (https://www.dyclassroom.com/c/c-structure-in-structure) C Programming

Shell Programming - Variables (https://www.dyclassroom.com/unix/shell-programming-variables) Unix

Shell Programming - Introduction (https://www.dyclassroom.com/unix/shell-programming-introduction) Unix

jQuery - Handling Events (https://www.dyclassroom.com/jquery/jquery-handling-events) jQuery

C - Function returning structure (https://www.dyclassroom.com/c/c-function-returning-structure) C Programming

C - Passing structure to function (https://www.dyclassroom.com/c/c-passing-structure-to-function) C Programming

C - Structures and Arrays (https://www.dyclassroom.com/c/c-structures-and-arrays) C Programming