ADA Lab

Assignment - 6

Dated: 10-03-2021

Sub Code: CSE-228

Vivek Kumar Ahirwar

191112419

CSE - 3

Department of
Computer Science and Engineering

Subject Coordinator:
Prof. Manish Pandey

Maulana Azad
National Institute of Technology,
BHOPAL – 462 003 (INDIA)

Contents

Problem 1: 0-1 Knapsack Problem

Write program (in any language) to solve the following problem: Consider two integer arrays val[0..n-1] and wt[0..n-1] which represent values and weights associated with n items respectively. Also given an integer W which represents knapsack capacity, find out the maximum value subset of val[] such that sum of the weights of this subset is smaller than or equal to W. You cannot break an item, either pick the complete item or don't pick it.

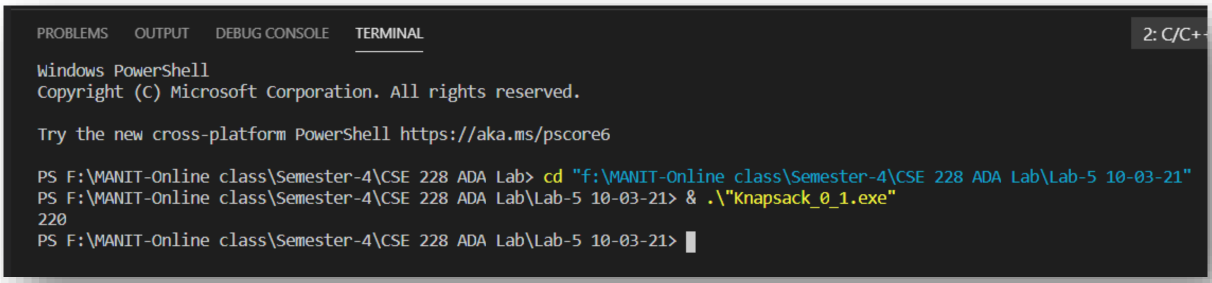
Value[] = {60, 100, 120}
Weight[] = {10, 20, 30}
W = 50

Code

```
// Keep Changing....@Vi

// 0-1 Knapsack Problem

#include <bits/stdc++.h>
using namespace std;

// Top-down (Recursive) approach of Knapsack Problem

// Returns maximum profit value
int knapSackRec(int W, int wt[], int val[], int i, int **memo)
{
    // base condition
    if (i < 0)
        return 0;
    if (memo[i][W] != -1)
        return memo[i][W];

    if (wt[i] > W)
    {

        // Store the value of function call
        // stack in table before return
        memo[i][W] = knapSackRec(W, wt, val, i - 1, memo);
        return memo[i][W];
    }
    else
    {
        // Store value in a table before return
        memo[i][W] = max(val[i] + knapSackRec(W - wt[i], wt, val, i - 1, memo)
,
                         knapSackRec(W, wt, val, i - 1, memo));

        // Return value of table after storing
        return memo[i][W];
```

```cpp
    }
}

int knapSack(int W, int wt[], int val[], int n)
{
    int **memo;
    memo = new int *[n];

    // loop to create the table dynamically
    for (int i = 0; i < n; i++)
        memo[i] = new int[W + 1];

    // fill the table with -1
    for (int i = 0; i < n; i++)
        for (int j = 0; j < W + 1; j++)
            memo[i][j] = -1;

    return knapSackRec(W, wt, val, n - 1, memo);
}

int main()
{
    int val[] = {60, 100, 120};
    int wt[] = {10, 20, 30};
    int W = 50;
    int n = sizeof(val) / sizeof(val[0]);
    cout << knapSack(W, wt, val, n);
    return 0;
}
```

Output



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS F:\MANIT-Online class\Semester-4\CSE 228 ADA Lab> cd "f:\MANIT-Online class\Semester-4\CSE 228 ADA Lab\Lab-5 10-03-21"
PS F:\MANIT-Online class\Semester-4\CSE 228 ADA Lab\Lab-5 10-03-21> & .\"Knapsack_0_1.exe"
220
PS F:\MANIT-Online class\Semester-4\CSE 228 ADA Lab\Lab-5 10-03-21> 
```

Analysis

Time Complexity: O(N*W).
As redundant calculations of states are avoided.

Auxiliary Space: O(N*W).
The use of 2D array data structure for storing intermediate states.