**Experiment-5**

**Aim-** Implement 0/1 Knapsack Problem using Dynamic Programming method.

**Theory-**

The 0/1 Knapsack Problem is an optimization problem where we have to select items with given weights and values to maximize the total value without exceeding a weight limit. Using dynamic programming, we solve the problem by building a table dp[i][w], where each entry represents the maximum value that can be obtained with the first i items and a knapsack capacity of w. The solution involves deciding whether to include or exclude each item, ensuring optimal substructure and overlapping subproblems.

**Software Used –** Visual Studio Code

**Code-**

#include <iostream>

#include <vector>

#include <algorithm>

#include <cmath>

#include <map>

#include <set>

#include <queue>

#include <stack>

#include <string>

using namespace std;

typedef long long int ll;

int knapsackRec(vector<int> val,vector<int> wt,int W,int n,vector<vector<int> > dp){//O(n\*W)

if((n==0) || (W==0)){

return 0;

}

if(dp[n][W] != -1){

return dp[n][W];

}

int itemWt=wt[n-1];

int itemval=val[n-1];

if(itemWt<=W){// valid wt

//include

int ans1= knapsackRec(val,wt,W-itemWt,n-1,dp)+itemval;

int ans2= knapsackRec(val,wt,W,n-1,dp);

dp[n][W]= max(ans1,ans2);

}else{

dp[n][W]= knapsackRec(val,wt,W,n-1,dp);

}

return dp[n][W];

}

int main() {

int n;

cout << "Enter the number of items: ";

cin >> n;

// Input the values and weights of the items

vector<int> val(n), wt(n);

cout << "Enter the values of the items: ";

for (int i = 0; i < n; i++) {

cin >> val[i];

}

cout << "Enter the weights of the items: ";

for (int i = 0; i < n; i++) {

cin >> wt[i];

}

// Input the capacity of the knapsack

int W;

cout << "Enter the capacity of the knapsack: ";

cin >> W;

// Create a DP table initialized to -1

vector<vector<int> > dp(n + 1, vector<int>(W + 1, -1));

// Call the knapsack recursive function

int maxProfit = knapsackRec(val, wt, W, n, dp);

// Output the result

cout << "The maximum value that can be obtained is: " << maxProfit << endl;

return 0;

}

**Output-**

A screen shot of a computer

Description automatically generated