

10/11/2021

Lab Report-3

Course Title: Algorithms Lab

Course Code: CSE-206

Date of signature:………………………………..

|  |  |
| --- | --- |
| **Submitted by** **Name: Jahid Hasan**  ID: 201015069  Section: 201-EC  GREEN UNIVERSITY OF BANGLADESH | **Submitted to** **Tamim Al Mahmud**  Assistant Professor Dept. of CSE  GREEN UNIVERSITY OF BANGLADESH |

Submission Date:10-11-21

# OBJECTIVE:

The objective of this lab is to Implementation of 0-1 Knapsack Problem.

* Understand the basic of Knapsack Problem
* Apply dynamic programming to solve real-life optimal decision making

# RESOURCE:

* C++

# PROGRAM LOGIC:

1. Find the ratio value/weight for each item and short the item on the basis of the ratio.
2. Choose the item with the highest ratio and add them until we can’t add the next item as a whole.
3. In the end, add the next item as much as we can.
4. Print the maximum profit after the above steps.

**PROCEDURE:**

### Go to DEBUG => Run or press CTRL + F9 to run the program

2

# SOURCE CODE:

**Implementation of 0-1 Knapsack Problem:**

**//==== Implementation Of 0-1 Knapsack Problem ====//**

#include <iostream>

using namespace std;

int knapSack(int W, int weight[], int value[], int n){

int i, w;

int P[n+1][W+1];

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

for(w = 0; w<=W ; w++){

if(i==0 || w==0)

P[i][w]=0;

else if (weight[i-1]<=w)

P[i][w] = max(value[i-1]+P[i-1][w-weight[i-1]], P[i-1][w]);

else

P[i][w] = P[i-1][w];

}

}

return P[n][W];

}

int main(){

int n, i, W, weight[20], value[20];

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

cin>>n;

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

cin>>weight[i];

cin>>value[i];

}

cout<<"Enter the size of the KnapSack: ";

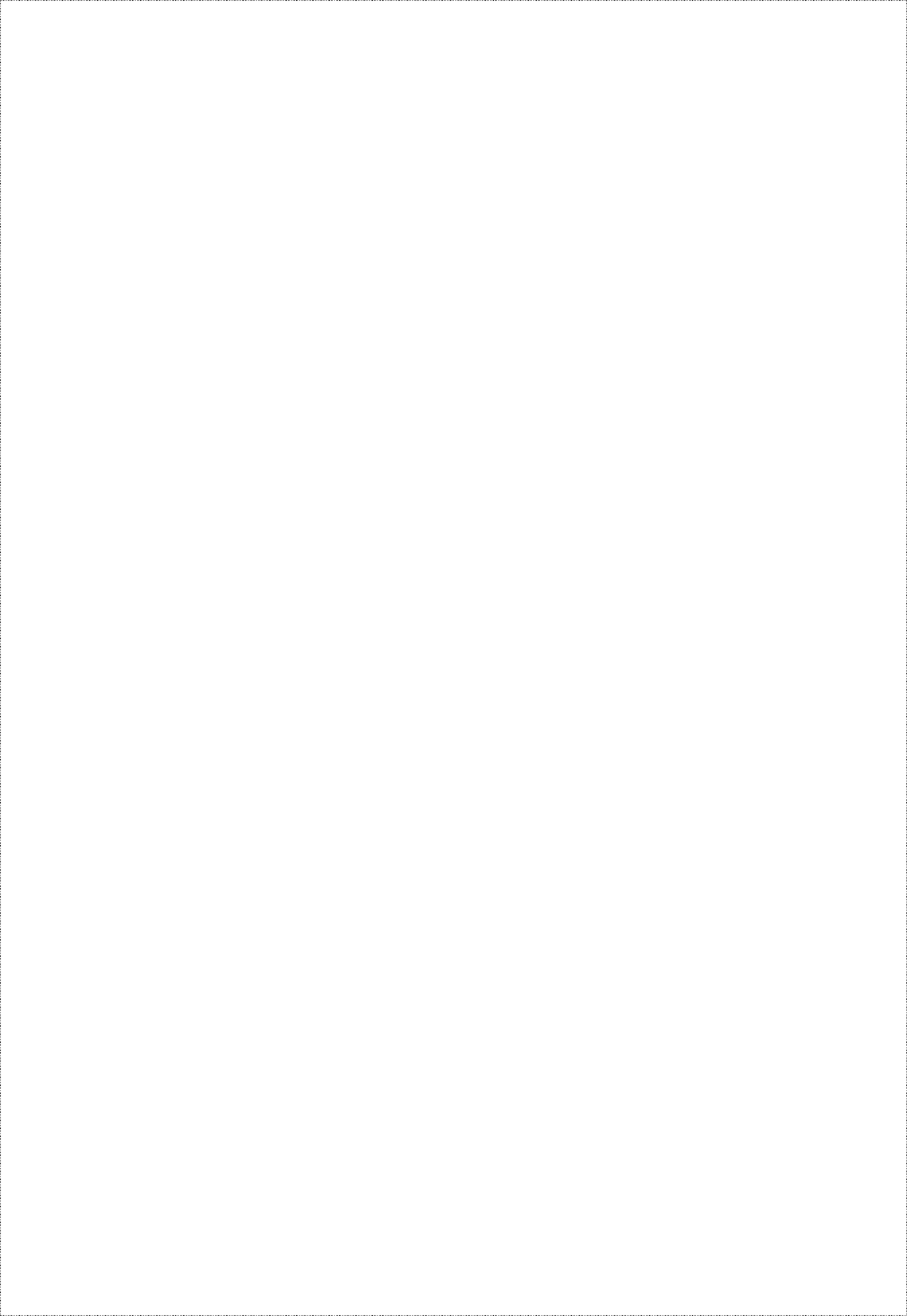
cin>>W;

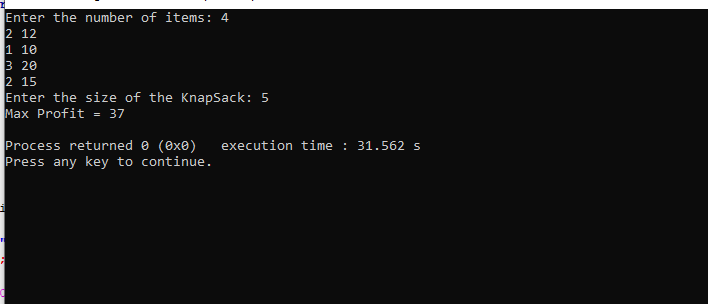
int profit = knapSack(W, weight, value, n);

cout<<"Max Profit = "<< profit<<endl;

}

4

**Output:**



5