MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL TECHNICAL UNIVERSITY

«KHARKIV POLYTECHNIC INSTITUTE»

Department of Software Engineering and Management Information Technologies

Report from lab № 11

Discipline «Algorithm and Data Structures»

Executed by:

OUSSAMA BARKANE

Student Group:

1. KH201.8ei2

Checked by:

Prof M.O. Bilova

Kharkiv

2019

**THEME :** GREEDY ALGORITHMS

**Objective**: learn how to use greedy algorithms and evaluate their complexity.

**Task :**

Develop a program that reads the input and solves the problem using greedy

algorithm. Determine the complexity of the algorithm.

**Variant** : Solve the fractional knapsack problem. Input: positive integers N, W (1 <N<256, 1 <W <1024) being the number of products and knapsack capacity, thesequence of N pairs p[i], v[i] being the cost and the weight of product number i. Output: a set of real numbers that indicate how many of each product must be put into a knapsack so that the total cost was maximized.

**Theory :**

A greedy algorithm is a simple, intuitive algorithm that is used in optimization problems. The algorithm makes the optimal choice at each step as it attempts to find the overall optimal way to solve the entire.

The knapsack problem or rucksack problem is a problem in combinatorial optimization: Given a set of items, each with a mass and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most valuable items.

**Progress of the Lab**

**Code :**

#include<iostream>

using namespace std;

int main()

{

int array[2][100], n, w, i, curw, used[100], maxi = -1, totalprofit = 0;

//input number of objects

cout << "Enter number of objects: ";

cin >> n;

//input max weight of knapsack

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

cin >> w;

/\* Array's first row is to store weights

second row is to store profits \*/

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

{

cin >> array[0][i] >> array[1][i];

}

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

{

used[i] = 0;

}

curw = w;

//loop until knapsack is full

while (curw >= 0)

{

maxi = -1;

//loop to find max profit object

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

{

if ((used[i] == 0) && ((maxi == -1) || (((float) array[1][i]

/ (float) array[0][i]) > ((float) array[1][maxi]

/ (float) array[0][maxi]))))

{

maxi = i;

}

}

used[maxi] = 1;

//decrease current wight

curw -= array[0][maxi];

//increase total profit

totalprofit += array[1][maxi];

if (curw >= 0)

{

cout << "\nAdded object " << maxi + 1 << " Weight: "

<< array[0][maxi] << " Profit: " << array[1][maxi]

<< " completely in the bag, Space left: " << curw;

}

else

{

cout << "\nAdded object " << maxi + 1 << " Weight: "

<< (array[0][maxi] + curw) << " Profit: "

<< (array[1][maxi] / array[0][maxi]) \* (array[0][maxi]

+ curw) << " partially in the bag, Space left: 0"

<< " Weight added is: " << curw + array[0][maxi];

totalprofit -= array[1][maxi];

totalprofit += ((array[1][maxi] / array[0][maxi]) \* (array[0][maxi]

+ curw));

}

}

//print total worth of objects filled in knapsack

cout << "\nBags filled with objects worth: " << totalprofit;

return 0;

}

