RAJSHAHI UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Lab report: 04
Course No.: CSE 2202

Date of Experiment: 20.11.2018
Date of Submission: 28.11.2018

Submitted to:
Biprodip Pal
Assistant Professor,
Department of Computer Science and Engineering
Rajshahi University of Engineering and Technology

Submitted by:
Riyad Morshed Shoeb
Roll No: 1603013
Section: A
Department of Computer Science and Engineering
Rajshahi University of Engineering and Technology

Problem: 0/1 Knapsack Problem

Approach:

1. **Initialize N objects with their weights and profits**
2. **For a knapsack having 70% weight capacity of total weight of N objects**

- **Find the maximum profit for 0/1 knapsack using brute force approach generating 2^N solutions and checking.**
- **Find the maximum profit for 0/1 knapsack using Greedy approach.**

Code:

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <ctime>
#include <vector>
#include <utility>
using namespace std;

class MoneyStack
{
public:
  int weight;
  int profit;
  int position;
};

int main(void)
{
  int INPUT_SIZE;
  int iterator;
  int BagCapacity;
  int WeightLifted;
  int ProfitLifted;
  int weight;
  int profit;
  int j;
  int temp;
  double time_taken;

  cout << "Enter the size of the input: ";
  cin >> INPUT_SIZE;
  MoneyStack InputData[INPUT_SIZE];
  BagCapacity = 0;
  srand(time(0));
  for(iterator=0; iterator<INPUT_SIZE; iterator++)
  {
    weight = (rand()%25)+1;
    profit = (rand()%1000)+1;
    BagCapacity += weight;
    InputData[iterator].weight = weight;
    InputData[iterator].profit = profit;
    InputData[iterator].position = iterator+1;
  }
```

```
  BagCapacity = (BagCapacity*7)/10;
  cout << "Money stacks\nWeight Profit" << endl;
  for(iterator=0; iterator<INPUT_SIZE; iterator++)
    cout    <<    InputData[iterator].weight    <<    "    "    <<
InputData[iterator].profit << endl;
  cout << "Capacity of Bag=" << BagCapacity << endl;

  //greedy approach
  cout << "Greedy Approach" << endl;

  //just using bubble sort here
  //since the input is small
  for(iterator=0; iterator<INPUT_SIZE; iterator++)
  {
    for(j=iterator+1; j<INPUT_SIZE; j++)
    {
      if(InputData[iterator].profit < InputData[j].profit)
      {
        //profit swap
        temp = InputData[iterator].profit;
        InputData[iterator].profit = InputData[j].profit;
        InputData[j].profit = temp;
        //weight swap
        temp = InputData[iterator].weight;
        InputData[iterator].weight = InputData[j].weight;
        InputData[j].weight = temp;
        //position swap
        temp = InputData[iterator].position;
        InputData[iterator].position = InputData[j].position;
        InputData[j].position = temp;
      }
    }
  }

  time_taken = clock();
  cout << "Stacks taken: ";
  WeightLifted = InputData[0].weight;
  ProfitLifted = InputData[0].profit;
  cout << InputData[0].position << " ";
  for(iterator=1; iterator<INPUT_SIZE; iterator++)
  {
    if((WeightLifted+InputData[iterator].weight) <= BagCapacity)
    {
      WeightLifted += InputData[iterator].weight;
      ProfitLifted += InputData[iterator].profit;
      cout << InputData[iterator].position << " ";
    }
  }
  cout << endl;
  time_taken = (clock() - time_taken);
  cout << "Weight Lifted= " << WeightLifted << endl;
  cout << "Profit Lifted= " << ProfitLifted << endl;
  cout << "Time taken= " << time_taken << endl;
}
```

Output:

```
knapsack
Enter the size of the input: 10
Money stacks
Weight Profit
6 484
3 577
10 759
6 355
24 949
22 115
15 556
5 715
5 384
6 771
Capacity of Bag=71
Greedy Approach
Stacks taken: 5 10 3 8 2 7 1
Weight Lifted= 69
Profit Lifted= 4811
Time taken= 1ms

Press any key to continue . . .
```