Shri Ramdeobaba College of Engineering & Management Nagpur-13

Department of Computer Application

Session: 2023-2024

Submission for

Course Name: Design Analysis and Algorithm Lab

Course Code: MCP546

Name of the Student: Jayesh Lalit Nandanwar

Class Roll No: 26

Semester: MCA II semester

Shift: 2

Batch: 2

Under the Guidance of

Prof. Manda Ukey

Date of submission: 13/04/2024

Practical 9

Aim: Perform 0-1 knapsack using the greedy method to find the optimum load that the knapsack can carry.
Input the no. of objects, weights, profits, and the maximum capacity of the container.
Display the time taken to perform this.

Code:

```
import java.util.*;

public class GreedyKnapsack {

   static class Item {
      int index;
      double weight;
      double profit;

      public Item(int index, double weight, double profit) {
         this.index = index;
         this.weight = weight;
         this.profit = profit;
      }
   }

   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      System.out.print("Enter the number of objects: ");
      int n = scanner.nextInt();

      double[] weights = new double[n];
      double[] profits = new double[n];

      System.out.println("Enter the weights of objects:");
      for (int i = 0; i < n; i++) {
         weights[i] = scanner.nextInt();
      }

      System.out.println("Enter the profits of objects:");
      for (int i = 0; i < n; i++) {
         profits[i] = scanner.nextInt();
      }

      System.out.print("Enter the maximum capacity of the knapsack: ");
      int capacity = scanner.nextInt();
```

```
        long startTime = System.nanoTime();
        KnapsackResult result = knapsack(weights, profits, capacity);
        long endTime = System.nanoTime();

        System.out.println("Maximum profit that can be achieved: " + result.maxProfit);
        System.out.println("Objects picked in order:");
        for (int index : result.pickedItems) {
            System.out.println("Object " + index + " (Weight: " + weights[index] + ",
Profit: " + profits[index] + ")");
        }
        System.out.println("Time taken: " + (endTime - startTime) + " nanoseconds");

        scanner.close();
    }

    static class KnapsackResult {
        double maxProfit;
        List<Integer> pickedItems;

        public KnapsackResult(double maxProfit, List<Integer> pickedItems) {
            this.maxProfit = maxProfit;
            this.pickedItems = pickedItems;
        }
    }

    public static KnapsackResult knapsack(double[] weights, double[] profits, int
capacity) {
        int n = weights.length;
        Item[] items = new Item[n];

        for (int i = 0; i < n; i++) {
            items[i] = new Item(i, weights[i], profits[i]);
        }

        Arrays.sort(items, (a, b) -> Double.compare(b.profit / b.weight, a.profit /
a.weight));

        double maxProfit = 0;
        List<Integer> pickedItems = new ArrayList<>();

        for (Item item : items) {
            if (capacity >= item.weight) {
                maxProfit += item.profit;
                capacity -= item.weight;
                pickedItems.add(item.index);
```

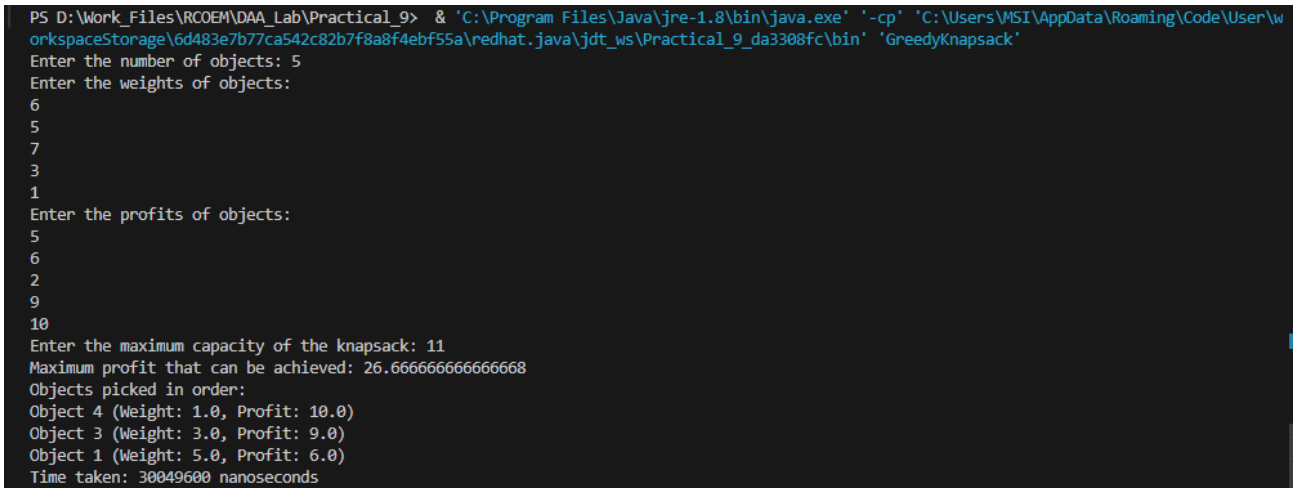
```
            } else {
                maxProfit += item.profit * (capacity / item.weight);
                break;
            }
        }

        return new KnapsackResult(maxProfit, pickedItems);
    }
}
```

Data Used:

Object	Weight	Profit
1	6	5
2	5	6
3	7	2
4	3	9
5	1	10

Knapsack Capacity: 11

Output:



Time Taken: 30049600 nanoseconds