International Journal of Modern Trends in Engineering and Research
www.ijmter.com
e-ISSN: 2349-9745

Comparative Analysis of Dynamic and Greedy Approaches for Dynamic Programming

Jay Vala[1], Dhara Monaka[2], Jaymit Pandya[3]
[1]*Asst. Prof., I.T. Department, G H Patel College of Engg & Tech, jayvala1623@gmail.com*
[2]*Asst. Prof., B.C.A. Department, Nandkunvarba Mahila College, dhara.monaka123@gmail.com*
[3]*Asst. Prof., I.T. Department, G H Patel College of Engg & Tech, erpandyajaymit@gmail.com*

Abstract— This paper analyze few algorithms of the 0/1 Knapsack Problem and fractional knapsack problem. This problem is a combinatorial optimization problem in which one has to maximize the benefit of objects without exceeding capacity. As it is an NP-complete problem, an exact solution for a large input is not possible. Hence, paper presents a comparative study of the Greedy and dynamic methods. It also gives complexity of each algorithm with respect to time and space requirements. Our experimental results show that the most promising approaches are dynamic programming.

Keywords- knapsack, dynamic programming, greedy programming, NP-Complete, complexity

I. INTRODUCTION

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 [1].
The 0-1 knapsack problem can solve using a partitioning algorithm for finding break item, weak reduction, strong upper bound, finding the solution vector and minimality.

II. DIFFERENT APPROACHES TO PROBLEM

1) Greedy Approach
A thief robbing a store and can carry a maximal weight of *w* into their knapsack. There are n items and i^{th} item weigh *wi* and is worth *vi* dollars. What items should thief take? This version of problem is known as Fractional knapsack problem. The setup is same, but the thief can take fractions of items, meaning that the items can be broken into smaller pieces so that thief may decide to carry only a fraction of *xi* of item *i*, where $0 \leq xi \leq 1$ [2][3].

2) Dynamic Approach
Again a thief robbing a store and can carry a maximal weight of *w* into their knapsack. There are n items and *ith* item weigh *w i* and is worth *vi* dollars. What items should thief take? This version of problem is known as 0-1 knapsack problem. The setup is the same, but the items

may not be broken into smaller pieces, so thief may decide either to take an item or to leave it (binary choice), but may not take a fraction of an item [2][3].

III. GREEDY ALGORITHM

It is as follows.

1. Calculate $V_i = v_i/s_i$ for i=1,2,…,n
2. Sort the items by declaring V_i
3. Find j such that $s_1+s_2+\ldots..+ s_{n<= S} < s_1+s_2+\ldots..+ s_{n+1}$
4.

IV. DYNAMIC ALGORITHM

It is as follows.

for i=1 to n
if s_i < = s
    if v_i + V[i-1,s- s_i] > V[i-1, s]
        V[i,s] = v_i + V[i-1,s- s_i]
    else
        V[i,s] = V[i-1,s]
else V[i,s] = V[i-1,s]

V. RESULT ANALYSIS

Implemented knapsack problem with different values of weight and profit or value in Turbo C. If we consider a data value is w={1, 2, 5}, v={1, 6, 18} and Carrying capacity W= 7 then output of greedy is:

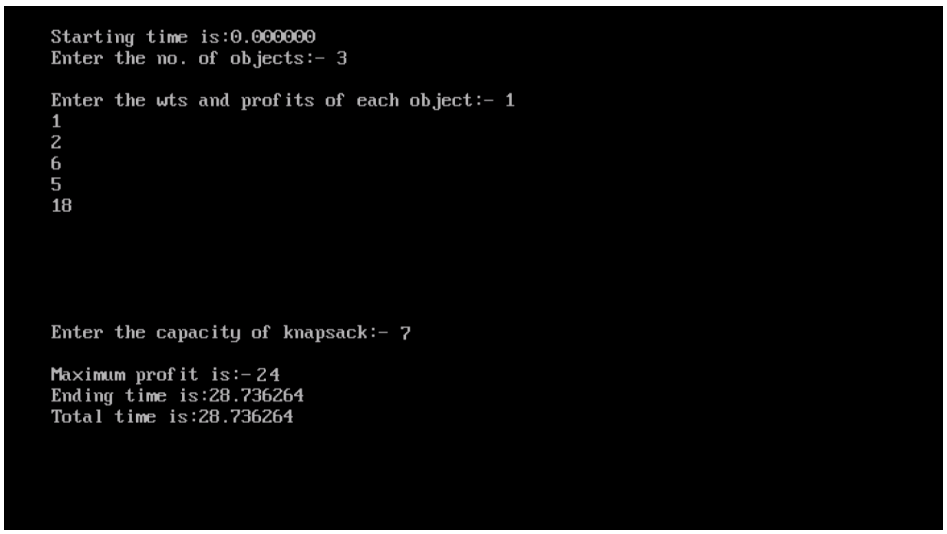

Fig. 1 Solved by Greedy approach

Same data values and solving by Dynamic programming.

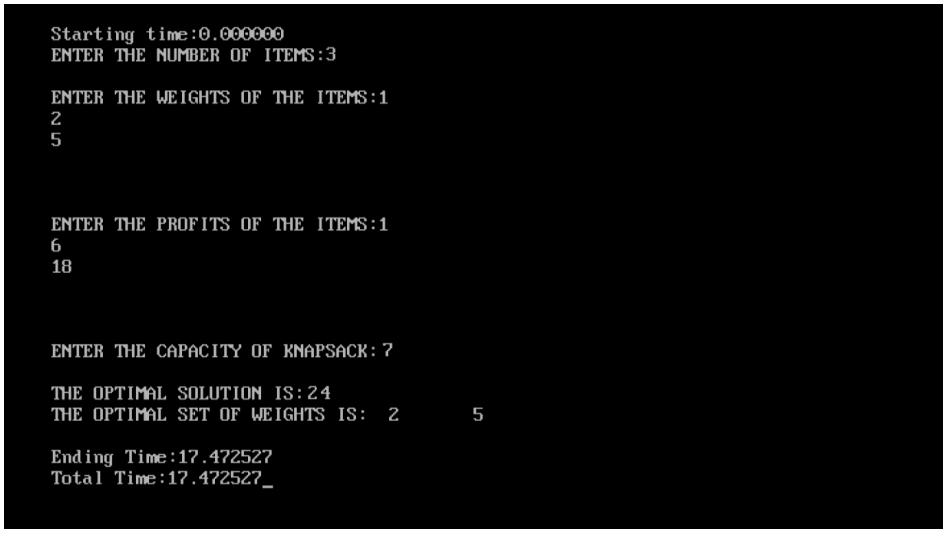

Fig. 2 Solved by Dynamic programming

After implemented knapsack problem in c programming for different values of weight and profit. Result of both methods gives same optimal solution and different time.

Method	Input Data	Capacity	Profit	Time
Greedy	W={1,3,4,5} V={1,4,5,6}	9	11	15.86
	W={1,2,5,6,7} , V={1,6,18,20,28}	11	40	25.73
	W={10,20,30,40,50} V={10,30,64,50,60}	100	156	21.68
Dynamic	W={1,3,4,5} V={1,4,5,6}	9	11	17.69
	W={1,2,5,6,7} , V={1,6,18,20,28}	11	40	18.47
	W={10,20,30,40,50} V={10,30,64,50,60}	100	156	19.65

VI. CONCLUSION

In this paper we conclude that for particular one knapsack problem we can implement two methods greedy and dynamic. But when we implemented both method for different dataset values then we notice something like, we consider comparison parameter as optimal profit or total value for filling knapsack using available weight then dynamic and greedy both are gaining same profit. If we consider time then dynamic take less amount of time compare with greedy. So we can conclude that dynamic is better than greedy with respect to time.

REFERENCES

[1]. George B. Dantzig, Discrete-Variable Extremum Problems, Operations Research Vol. 5, No. 2, April 1957, pp. 266–288,doi:10.1287/opre.5.2.266
[2] Gossett, Eric. Discreet Mathematics with Proof. New Jersey: Pearson Education Inc., 2003.

e-ISSN: 2349-9745

[3] Levitin, Anany. The Design and Analysis of Algorithms. New Jersey: Pearson Education Inc., 2003.
[4] Mitchell, Melanie. An Introduction to Genetic Algorithms. Massachusettss: The MIT Press, 1998.
[5] Obitko, Marek. ―Basic Description.‖ IV. Genetic Algorithm. Czech Technical University (CTU). http://cs.felk.cvut.cz/~xobitko/ga/gaintro.html
[6] Different Approaches to Solve the 0/1 Knapsack Problem. Maya Hristakeva, Dipti Shrestha; Simpson Colleges

