Greedy Algorithms

OBJECTIVES

After studying this chapter, the student will be able to

- Understand the concept of greedy algorithms
- Solve 0/1 knapsack problem using greedy algorithms

1- CONCEPT OF GREEDY APPROACH

Greedy approach of solving a problem calls for the selection of the most promising intermediate solution at that instance. The intermediate solution which seems promising at a point might not be that good in the long run. In order to solve a problem via greedy approach, we select an input. If the solution satisfies the greedy goal, then it is taken in the solution set; otherwise it is left. The process is depicted as follows in Algorithm 1.

Algorithm 1 Greedy (X, n)

```
{
SET Solution Set to φ
SELECT y: y∈ X.
If y is a feasible solution, then include it in the solution set, else
proceed.
REPEAT the above two steps till all the elements of the array X have
been processed.
Return solution set.
}
```

The greedy algorithms work most of the times; however, they are not always optimal. The economies run on the basis of greedy approach. Companies are able to survive because of greedy approach. Even those who rule us follow the greedy approach. The modus operandi may not be good for the people, the country, and the humanity, but at least gives transient gains to those who make the policies.

This section explains the greedy approach by taking an example of 0/1 knapsack problem.

2- 0/1 KNAPSACK PROBLEM

In the knapsack problem, a subset of items is to be selected from the given set of items. The subset should completely (or almost completely) fill the bag and the profit earned by the selected elements should be maximum. The capacity of the bag, weights of the items, and the profit earned by selecting the items are given as an input of the problem.

Input

- The set of items $x : \{x_1, x_2, x_3, \ldots, x_n\}$.
- The weights of the above items $W : \{w_1, w_2, w_3, \ldots, w_n\}$ and
- The profits earned by picking the items $P : \{p_1, p_2, p_3, \ldots, p_n\}$.

Output

- $x_n = 1$ denotes that the item has been picked and $x_n = 0$ means that the item has not been picked.

Constraint

- The total weight of the selected items is less than or equal to the weight of the bag, i.e.,

$$x_1 \times w_1 + x_1 \times w_2 + x_3 \times w_1 \ldots \leq m$$

  where m is the weight of the bag.
- The profit earned is to be maximized, i.e.,

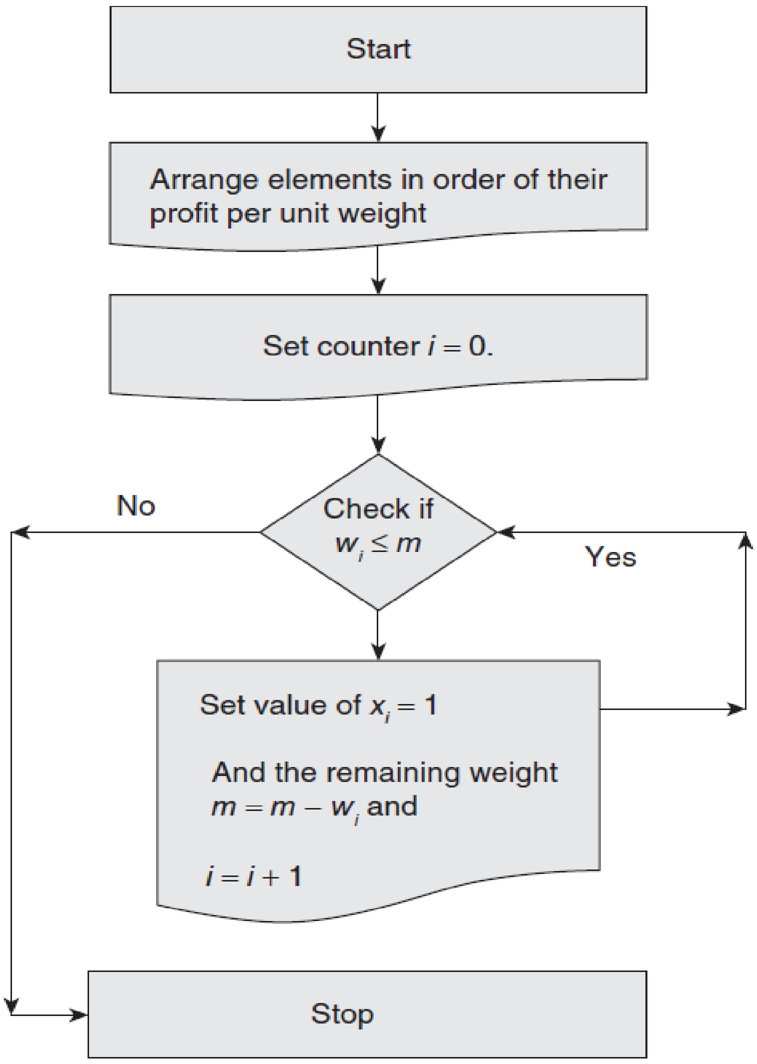
$$x_1 \times p_1 + x_1 \times p_2 + x_3 \times p_1$$

  is to be maximized.

Definition The selection of items from a given set in such a way that the total weight is less than or equal to the given weight, and the profit earned by picking up the elements is maximum is referred to as ***knapsack problem***.

Solution strategy

1. Find out profit per unit weight of the items. This is because while selecting an edge, it is important to maximize the profit at the same time it is desired to fill the bag as much as possible.
2. Arrange the array obtained in the previous step in decreasing order.
3. Pick the items from the sorted array one by one till there is a space in the bag.


Start
Arrange elements in order of their profit per unit weight
Set counter $i = 0$.
Check if $w_i \leq m$
No
Yes
Set value of $x_i = 1$
And the remaining weight $m = m - w_i$ and
$i = i + 1$
Stop

Procedure for solving knapsack problem

Algorithm 2 Knapsack (X, W, P, m) returns profit earned

```
{
//X is the array which has items, W is the array containing the weights of the
//items, P is the array containing profits of the items and m is the capacity of
//the Knapsack, t is the remaining weight, the variable p depicts profit earned
//Arrange the items in the non-increasing order of their p[i]/w[i];
t=m;
while (t>W[i])
         {
         Pick the item X[i];
         p=p+P[i];
         t=t-W[i];
         }
return p;
}
```

Applications

The problem, though, can be solved by many approaches; the approach discussed in this section is one of the easiest. Moreover, the greedy approach is sure to optimize a smaller instance of the problem. Knapsack problem is used in the following domains:

- data mining
- networking
- regression testing
- test data generation, etc.

Complexity Analysis

Suppose there are n items in a set. The complexity of finding out profit per unit weight of each item would be $O(n)$. Moreover, the array needs to be sorted. In order to carry out the task, the complexity would be $O(n^2)$ or $O(n \times log_2 n)$, depending upon the type of algorithm selected. This is followed by picking of an item and checking whether there is any more space in the bag. The total complexity will therefore be *O(n log n) or* $O(n^2)$, depending upon the algorithm. Obviously, if n is too large, then the time complexity would be too high.

Example:

Let us consider that the capacity of the knapsack is W = 60 and the items are as shown in the following table.

Item	**A**	**B**	**C**
Price	100	280	120
Weight	10	40	20

Solution:

First we compute the ratio pi/wi.

Item	**A**	**B**	**C**
Price	100	280	120
Weight	10	40	20
Ratio	10	7	6

Using the Greedy approach, first item ***A*** is selected. Then, the next item ***B*** is chosen. Hence, the solution is **{A, B}** and the total profit is **100 + 280 = 380**.

However, the optimal solution of this instance can be achieved by selecting items, ***B*** and ***C***, where the total profit is **280 + 120 = 400**.

Hence, it can be concluded that Greedy approach may not give an optimal solution.

H.W

Using the Greedy approach, solve the following 0/1 knapsack problem if the capacity of the knapsack is W = 25 and the items are as shown in the following table.

Item	**A**	**B**	**C**	**D**
Profit	24	18	18	10
Weight	24	10	10	7