**Greedy algorithms: The general method, Fractional Knapsack problem**

Greedy Method

Among all the algorithmic approaches, the simplest and straightforward approach is the Greedy method. In this approach, the decision is taken on the basis of current available information without worrying about the effect of the current decision in future.

Greedy algorithms build a solution part by part, choosing the next part in such a way, that it gives an immediate benefit. This approach never reconsiders the choices taken previously. This approach is mainly used to solve optimization problems. Greedy method is easy to implement and quite efficient in most of the cases. Hence, we can say that Greedy algorithm is an algorithmic paradigm based on heuristic that follows local optimal choice at each step with the hope of finding global optimal solution.

In many problems, it does not produce an optimal solution though it gives an approximate (near optimal) solution in a reasonable time.

**All greedy algorithms follow a basic structure:**

getOptimal(Item, arr[], int n)

1) Initialize empty result : result = {}

2) While (All items are not considered)

// We make a greedy choice to select

// an item.

i = SelectAnItem()

// If i is feasible, add i to the

// result

if (feasible(i))

result = result U i

3) return result

## **Components of Greedy Algorithm**

Greedy algorithms have the following five components −

* **A candidate set** − A solution is created from this set.
* **A selection function** − Used to choose the best candidate to be added to the solution.
* **A feasibility function** − Used to determine whether a candidate can be used to contribute to the solution.
* **An objective function** − Used to assign a value to a solution or a partial solution.
* **A solution function** − Used to indicate whether a complete solution has been reached.

## **Areas of Application**

Greedy approach is used to solve many problems, such as

* Finding the shortest path between two vertices using Dijkstra’s algorithm.
* Finding the minimal spanning tree in a graph using Prim’s /Kruskal’s algorithm, etc.

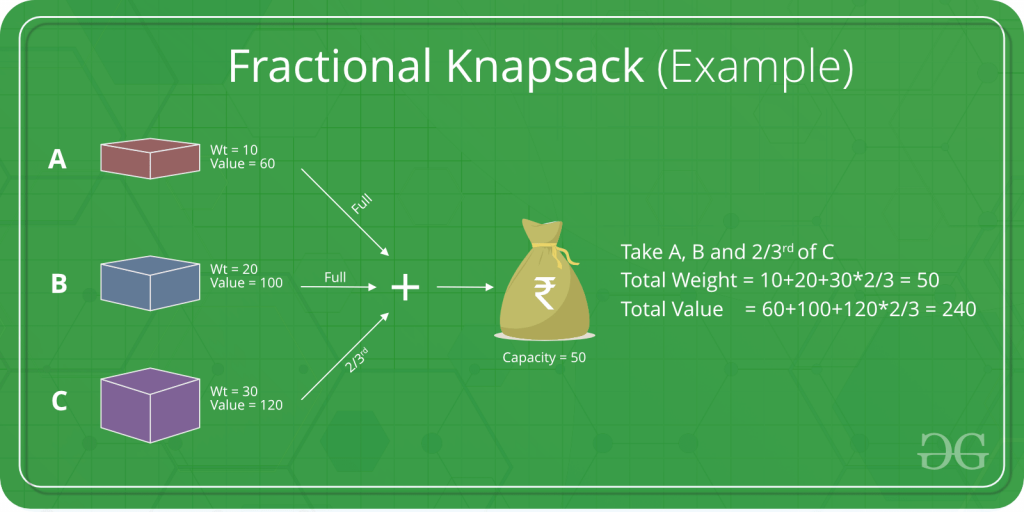
## **Where Greedy Approach Fails**

In many problems, Greedy algorithm fails to find an optimal solution, moreover it may produce a worst solution. Problems like Travelling Salesman and Knapsack cannot be solved using this approach.

# Fractional Knapsack

The Greedy algorithm could be understood very well with a well-known problem referred to as Knapsack problem. Although the same problem could be solved by employing other algorithmic approaches, Greedy approach solves Fractional Knapsack problem reasonably in a good time. Let us discuss the Knapsack problem in detail.

For example consider the [Fractional Knapsack Problem](https://www.geeksforgeeks.org/fractional-knapsack-problem/). The local optimal strategy is to choose the item that has maximum value vs weight ratio. This strategy also leads to global optimal solution because we allowed to take fractions of an item.



## **Knapsack Problem**

Given a set of items, each with a weight and a value, determine a subset of items 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.

The knapsack problem is in combinatorial optimization problem. It appears as a subproblem in many, more complex mathematical models of real-world problems. One general approach to difficult problems is to identify the most restrictive constraint, ignore the others, solve a knapsack problem, and somehow adjust the solution to satisfy the ignored constraints.

### **Applications**

In many cases of resource allocation along with some constraint, the problem can be derived in a similar way of Knapsack problem. Following is a set of example.

* Finding the least wasteful way to cut raw materials
* portfolio optimization
* Cutting stock problems

### **Problem Scenario**

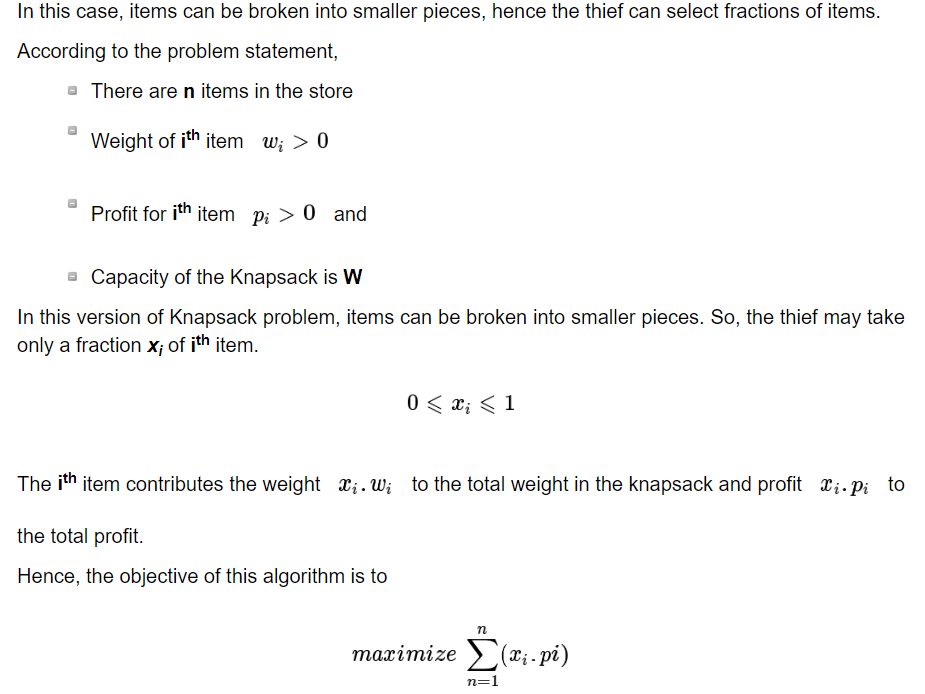
A thief is robbing a store and can carry a maximal weight of ***W*** into his knapsack. There are n items available in the store and weight of ***ith*** item is ***wi*** and its profit is ***pi***. What items should the thief take?

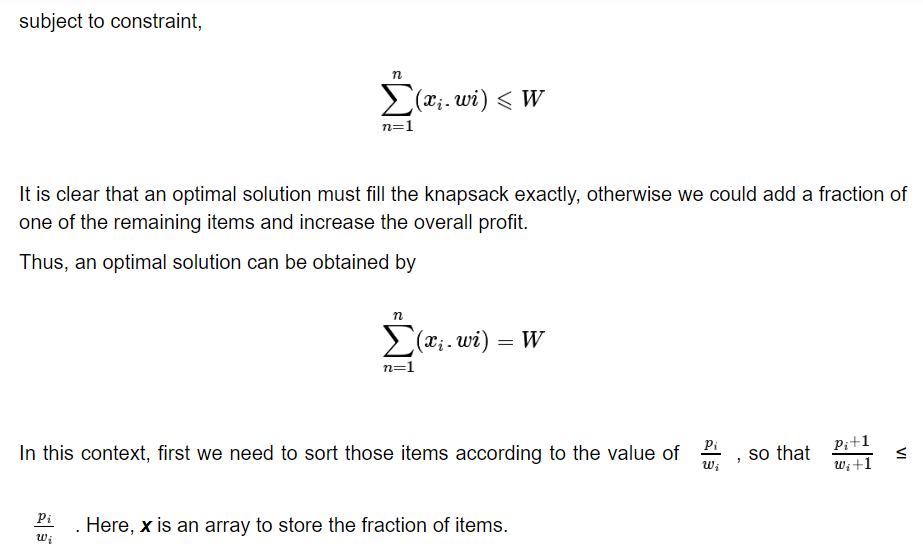
In this context, the items should be selected in such a way that the thief will carry those items for which he will gain maximum profit. Hence, the objective of the thief is to maximize the profit.

Based on the nature of the items, Knapsack problems are categorized as

* Fractional Knapsack
* Knapsack

## **Fractional Knapsack**

****

****

**Algorithm: Greedy-Fractional-Knapsack (w[1..n], p[1..n], W)**

for i = 1 to n

do x[i] = 0

weight = 0

for i = 1 to n

if weight + w[i] ≤ W then

x[i] = 1

weight = weight + w[i]

else

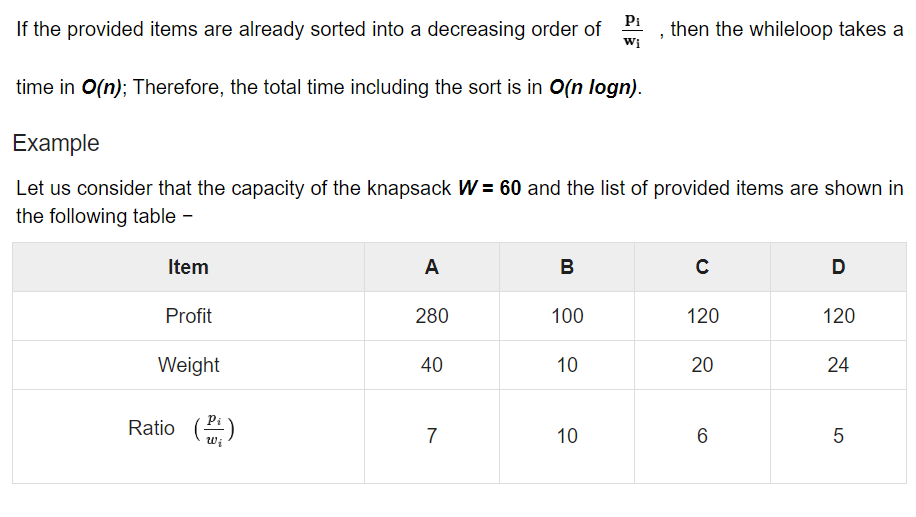
x[i] = (W - weight) / w[i]

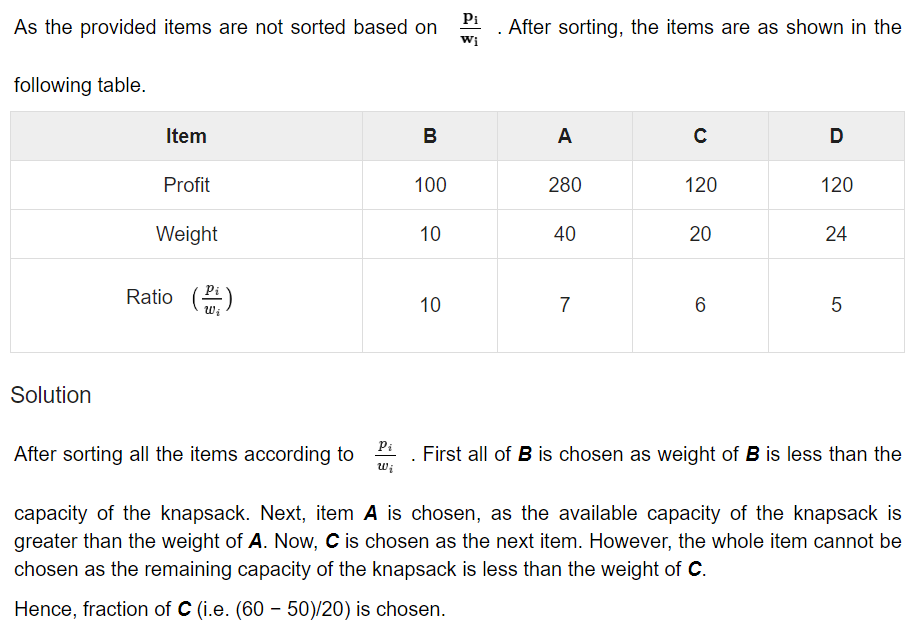
weight = W

break

return x

**Analysis**





Now, the capacity of the Knapsack is equal to the selected items. Hence, no more item can be selected.

The total weight of the selected items is **10 + 40 + 20 \* (10/20) = 60**

And the total profit is **100 + 280 + 120 \* (10/20) = 380 + 60 = 440**

This is the optimal solution. We cannot gain more profit selecting any different combination of items.

**RELEVANT READING MATERIAL AND REFERENCES:**

**Source Notes:**

1. <https://www.tutorialspoint.com/design_and_analysis_of_algorithms/design_and_analysis_of_algorithms_fractional_knapsack.htm>
2. <https://www.geeksforgeeks.org/greedy-algorithms-general-structure-and-applications/>
3. <https://www.geeksforgeeks.org/greedy-algorithms/>

**Lecture Video:**

1. <https://youtu.be/m1p-eWxrt6g>

**Online Notes:**

1. <http://vssut.ac.in/lecture_notes/lecture1428551222.pdf>

**Text Book Reading:**

1. Cormen, Leiserson, Rivest, Stein, “*Introduction to Algorithms*”, Prentice Hall of India, 3rd edition 2012. problem, Graph coloring.

**In addition: PPT can be also be given.**