The knapsack problem states that − given a set of items, holding weights and profit values, one must determine the subset of the items to be added in a knapsack such that, the total weight of the items must not exceed the limit of the knapsack and its total profit value is maximum.

It is one of the most popular problems that take greedy approach to be solved. It is called as the **Fractional Knapsack Problem**.

To explain this problem a little easier, consider a test with 12 questions, 10 marks each, out of which only 10 should be attempted to get the maximum mark of 100. The test taker now must calculate the highest profitable questions – the one that he’s confident in – to achieve the maximum mark. However, he cannot attempt all the 12 questions since there will not be any extra marks awarded for those attempted answers. This is the most basic real-world application of the knapsack problem.

## Knapsack Algorithm

The weights (Wi) and profit values (Pi) of the items to be added in the knapsack are taken as an input for the fractional knapsack algorithm and the subset of the items added in the knapsack without exceeding the limit and with maximum profit is achieved as the output.

### Algorithm

* Consider all the items with their weights and profits mentioned respectively.
* Calculate Pi/Wi of all the items and sort the items in descending order based on their Pi/Wi values.
* Without exceeding the limit, add the items into the knapsack.
* If the knapsack can still store some weight, but the weights of other items exceed the limit, the fractional part of the next time can be added.
* Hence, giving it the name fractional knapsack problem.

### Examples

* For the given set of items and the knapsack capacity of 10 kg, find the subset of the items to be added in the knapsack such that the profit is maximum.

| **Items** | 1 | 2 | 3 | 4 | 5 |
| --- | --- | --- | --- | --- | --- |
| **Weights (in kg)** | 3 | 3 | 2 | 5 | 1 |
| **Profits** | 10 | 15 | 10 | 12 | 8 |

### Solution

**Step 1**

Given, n = 5

Wi = {3, 3, 2, 5, 1}

Pi = {10, 15, 10, 12, 8}

Calculate Pi/Wi for all the items

| **Items** | 1 | 2 | 3 | 4 | 5 |
| --- | --- | --- | --- | --- | --- |
| **Weights (in kg)** | 3 | 3 | 2 | 5 | 1 |
| **Profits** | 10 | 15 | 10 | 20 | 8 |
| **Pi/Wi** | 3.3 | 5 | 5 | 4 | 8 |

**Step 2**

Arrange all the items in descending order based on Pi/Wi

| **Items** | 5 | 2 | 3 | 4 | 1 |
| --- | --- | --- | --- | --- | --- |
| **Weights (in kg)** | 1 | 3 | 2 | 5 | 3 |
| **Profits** | 8 | 15 | 10 | 20 | 10 |
| **Pi/Wi** | 8 | 5 | 5 | 4 | 3.3 |

**Step 3**

Without exceeding the knapsack capacity, insert the items in the knapsack with maximum profit.

Knapsack = {5, 2, 3}

However, the knapsack can still hold 4 kg weight, but the next item having 5 kg weight will exceed the capacity. Therefore, only 4 kg weight of the 5 kg will be added in the knapsack.

| **Items** | 5 | 2 | 3 | 4 | 1 |
| --- | --- | --- | --- | --- | --- |
| **Weights (in kg)** | 1 | 3 | 2 | 5 | 3 |
| **Profits** | 8 | 15 | 10 | 20 | 10 |
| **Knapsack** | 1 | 1 | 1 | 4/5 | 0 |

Hence, the knapsack holds the weights = [(1 \* 1) + (1 \* 3) + (1 \* 2) + (4/5 \* 5)] = 10, with maximum profit of [(1 \* 8) + (1 \* 15) + (1 \* 10) + (4/5 \* 20)] = 37.