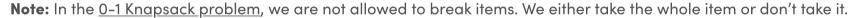


Fractional Knapsack Problem

Given the weights and values of **N** items, put these items in a knapsack of capacity **W** to get the maximum total value in the knapsack. In **Fractional Knapsack**, we can break items for maximizing the total value of the knapsack







Input:

Items as (value, weight) pairs arr[] = {{60, 10}, {100, 20}, {120, 30}} Knapsack Capacity, W = 50

Output: Maximum possible value = 240

Explanation: by taking items of weight 10 and 20 kg and 2/3 fraction of 30 kg.

Hence total price will be 60+100+(2/3)(120) = 240

Input:

Items as (value, weight) pairs arr[] = {{500, 30}} Knapsack Capacity, W = 10

Output: 166.667

Naive Approach: Try all possible subsets with all different fractions but that will be very inefficient.

밂 Dash 000 All \Box Articles Videos </> Problems (?) Quiz \mathcal{Q} Contest << >>

Greedy approach for fractional knapsack problem:

An **efficient solution** is to use the Greedy approach. The basic idea of the greedy approach is to calculate the ratio value/weight for each item and sort the item on the basis of this ratio. Then take the item with the highest ratio and add them until we can't add the next item as a whole and at the end add the next item as much as we can. Which will always be the optimal solution to this problem.



Follow the given steps to solve the problem using the above approach:



- Calculate the ratio(value/weight) for each item.
- Sort all the items in decreasing order of the ratio.
- Initialize res =0, curr_cap = given_cap.
- Do the following for every item "i" in the sorted order:
 - If the weight of the current item is less than or equal to the remaining capacity then add the value of that item into the result
 - o else add the current item as much as we can and break out of the loop
- Return res

Below is the implementation of the above approach:

```
# Structure for an item which stores weight and
# corresponding value of Item
class Item:
def __init__(self, value, weight):
    self.value = value
    self.weight = weight
```

Main areedy function to solve mobilem Courses **Tutorials** Johs Practice Contests # SULLTING TEEM ON DASTS OF LACTO arr.sort(key=lambda x: (x.value/x.weight), reverse=True) Dash # Uncomment to see new order of Items with their # ratio 000 # for item in arr: print(item.value, item.weight, item.value/item.weight) Αll # Result(value in Knapsack) \Box finalvalue = 0.0Articles # Looping through all Items for item in arr: # If adding Item won't overflow, add it completely Videos if item.weight <= W:</pre> W -= item.weight finalvalue += item.value </> Problems # If we can't add current Item, add fractional part # of it else: (?) finalvalue += item.value * W / item.weight Quiz break # Returning final value return finalvalue Contest # Driver's Code if name == " main ": # Weight of Knapsack W = 50<< arr = [Item(60, 10), Item(100, 20), Item(120, 30)]>> # Function call





```
Dash
  000
  All
  \Box
Articles
  Videos
  </>
Problems
  (?)
  Quiz
Contest
<<
    >>
```

```
max_val = fractionalKnapsack(W, arr)
print ('Maximum value we can obtain = {}'.format(max_val))
```

Output

Maximum value we can obtain = 240

Time Complexity: O(N log N)

Auxiliary Space: O(N)





Mark as Read

Report An Issue

If you are facing any issue on this page. Please let us know.