

# Competitive Programming

## Lab Assignment 01

Name : R Govardhan Sai Ganesh

Hall Ticket No : 2303A54044

Batch : 48

### Question 1: Fractional Knapsack (Greedy)

#### Problem Statement

You are given N items. Item i has value  $V_i$  and weight  $W_i$ . You have a knapsack of capacity W. You may take any fraction of an item (including the whole item). Your goal is to maximize the total value in the knapsack without exceeding the capacity. For each test case, output the maximum total value achievable. The result must be printed with exactly 6 digits after the decimal point.

#### Input Format

The first line contains an integer T, the number of test cases. For each test case: - The first line contains two integers N and W. - The next N lines each contain two integers  $V_i$  and  $W_i$ .

#### Output Format

For each test case, print one number: the maximum value, formatted to 6 decimal places.

#### Constraints

- 1 ≤ T ≤ 20

- 1 ≤ N ≤ 200000 (sum of N over all test cases ≤ 200000)

- 1 ≤  $V_i$ ,  $W_i$ , W ≤  $10^9$

#### Sample Input

1

3 50

60 10

100 20

120 30

Expected Output: 240.000000

#### Code:

```
import sys
```

```
input = sys.stdin.readline
```

```

t = int(input())
for _ in range(t):
    n, W = map(int, input().split())
    items = []
    for _ in range(n):
        v, w = map(int, input().split())
        items.append((v / w, v, w))
    items.sort(reverse=True)
    total_value = 0.0
    remaining = W

```

*for ratio, value, weight in items:*

```

if remaining == 0:
    break
if weight <= remaining:
    total_value += value
    remaining -= weight
else:
    total_value += ratio * remaining
    remaining = 0

```

*print(f"total\_value:.6f")*

The screenshot shows a Python code editor interface with the following details:

- Top Bar:** Includes buttons for Run, Debug, Stop, Share, Saved, Beautify, Language (Python 3), and settings.
- Code Area:** Displays the script 'main.py' containing the provided code for the knapsack problem.
- Output Area:** Shows the input data (5 items) and the program's output (total value).
- Status Bar:** Shows the message "...Program finished with exit code 0".

```

main.py
1 input = sys.stdin.readline
2 t = int(input())
3 for _ in range(t):
4     n, W = map(int, input().split())
5     items = []
6     for _ in range(n):
7         v, w = map(int, input().split())
8         items.append((v / w, v, w))
9     items.sort(reverse=True)
10    total_value = 0.0
11    remaining = W
12    for ratio, value, weight in items:
13        if remaining == 0:
14            break
15        if weight <= remaining:
16            total_value += value
17            remaining -= weight
18        else:
19            total_value += ratio * remaining
20            remaining = 0
21    print(f"total_value:.6f")
22
23
24
25
26
27
28
29

```

**Input Data:**

```

3 50
60 10
100 20
120 30
240.000000

```

**Output:**

```

total_value:240.000000

```

## Question 2: Package Priority Sorting (Divide and Conquer)

### Problem Statement

A warehouse records package priority scores as integers. To dispatch efficiently, you must sort the scores in non-decreasing order using merge sort (divide and conquer). For each test case, output the sorted list.

### Input Format

The first line contains integer T. For each test case:

- First line: N
- Second line: N integers (priority scores)

### Output Format

For each test case, print the sorted array in one line (space-separated).

### Constraints

- $1 \leq T \leq 20$
- $1 \leq N \leq 200000$  (sum of N over all test cases  $\leq 200000$ )
- $-10^9 \leq A_i \leq 10^9$

### Sample Input

```
1
7
4 1 6 2 5 3 2
```

### Expected Output

```
1 2 2 3 4 5 6
```

### Code:

```
def merge_sort(arr):  
    if len(arr) <= 1:  
        return arr  
  
    mid = len(arr) // 2  
  
    left = merge_sort(arr[:mid])  
    right = merge_sort(arr[mid:])  
  
    return merge(left, right)  
  
def merge(left, right):  
    i = j = 0  
  
    result = []
```

```
while i < len(left) and j < len(right):
```

```
    if left[i] <= right[j]:
```

```
        result.append(left[i])
```

```
        i += 1
```

```
    else:
```

```
        result.append(right[j])
```

```
        j += 1
```

```
result.extend(left[i:])
```

```
result.extend(right[j:])
```

```
return result
```

```
T = int(input())
```

```
for _ in range(T):
```

```
    N = int(input())
```

```
    arr = list(map(int, input().split()))
```

```
    sorted_arr = merge_sort(arr)
```

```
    print(" ".join(map(str, sorted_arr)))
```

The screenshot shows a Python code editor interface with the following details:

- File Bar:** Run, Debug, Stop, Share, Saved, Beautify.
- Language:** Python 3.
- Code Area:** The file `main.py` contains the provided Python code for merge sort and its execution.
- Output Area:** Shows the input array [4, 1, 6, 2, 5, 3, 2, 1, 2, 2, 3, 4, 5, 6] and the output "[1, 2, 2, 3, 4, 5, 6]".
- Bottom Status:** Program finished with exit code 0.

```
1 def merge_sort(arr):
2     if len(arr) <= 1:
3         return arr
4
5     mid = len(arr) // 2
6     left = merge_sort(arr[:mid])
7     right = merge_sort(arr[mid:])
8
9     return merge(left, right)
10
11 def merge(left, right):
12     i = j = 0
13     result = []
14
15     while i < len(left) and j < len(right):
16         if left[i] <= right[j]:
17             result.append(left[i])
18             i += 1
19         else:
20             result.append(right[j])
21             j += 1
22
23     result.extend(left[i:])
24     result.extend(right[j:])
25
26     return result
27
28 T = int(input())
29 for _ in range(T):
```

input

```
1
7
4 1 6 2 5 3 2
1 2 2 3 4 5 6
```

```
...Program finished with exit code 0
Press ENTER to exit console.
```