

Competitive Programming

Lab Assignment 01

Name : **BG.Sreevani**
Hall Ticket No : **2303A54066**
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:

- $1 \leq T \leq 20$
- $1 \leq N \leq 200000$ (sum of N over all test cases ≤ 200000)
- $1 \leq V_i, W_i, W \leq 10^9$

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, and Language (Python 3).
- Code Area:** Displays the script `main.py` containing the provided Python code.
- Output Area:** Shows the input data and the program's output. The input data is:


```

1
3 50
60 10
100 20
120 30
240.000000
      
```

 The output is:


```

...Program finished with exit code 0
      
```

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 ≤ 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 code editor interface with a dark theme. At the top, there's a toolbar with icons for Run, Debug, Stop, Share, and Save. The status bar indicates the language is Python 3. The main area contains the following code:

```

main.py
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):

```

Below the code editor is a terminal window showing the execution of the program. The user inputs two sets of numbers:

```

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

```

The output shows the sorted arrays:

```

input
1
2
3
4 1 6 2 5 3 2
5
6 1 2 2 3 4 5 6
...Program finished with exit code 0
Press ENTER to exit console.

```