Lab Assignment 2
DAA
Ikjot Singh
102116071
2CS11

assignment2 > activitySelection.py > ...

```python
st=list(map(int,input("enter start: ").split()))
fin=list(map(int,input("enter finish: ").split()))
def selection(st,fin):
    i=0
    print(i,end=' ')
    for j in range(len(st)):
        if st[j]>=fin[i]:
            print(j,end=' ')
            i=j
selection(st,fin)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER GITLENS COMMENTS

```
PS C:\Users\Ikjot singh\Coding\DAA\assignment2> cd "c:/Users/Ikjot singh/Coding/DAA/ass
PS C:\Users\Ikjot singh\Coding\DAA\assignment2> & "C:/Users/Ikjot singh/AppData/Local/P
jot singh/Coding/DAA/assignment2/activitySelection.py"
enter start: 1 3 0 5 8 5
enter finish: 2 4 6 7 9 9
0 1 3 4
PS C:\Users\Ikjot singh\Coding\DAA\assignment2>
```

assignment2 > jobSequencing.py > ...

```python
def JobSequencing(arr,t):
    n = len(arr)
    for i in range(n):
        for j in range(n - 1 - i):
            if arr[j][2] < arr[j + 1][2]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
        result = [False] * t
        job = ['-1'] * t
        for i in range(len(arr)):
            for j in range(min(t - 1, arr[i][1] - 1), -1, -1):
                if result[j] is False:
                    result[j] = True
                    job[j] = arr[i][0]
                    break
        print(job)

arr = [['a', 2, 100],  # Job Array
       ['b', 1, 19],
       ['c', 2, 27],
       ['d', 1, 25],
       ['e', 3, 15]]
print("the max profit job sequence:")
JobSequencing(arr,3)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER GITLENS COMMENTS

```
PS C:\Users\Ikjot singh\Coding\DAA\assignment2> cd "c:/Users/Ikjot singh/Coding/DAA/assignment2"
PS C:\Users\Ikjot singh\Coding\DAA\assignment2> & "C:/Users/Ikjot singh/AppData/Local/Programs/Python/Python310/python.exe" "c:/Users/Ikjot singh/Coding/DAA/assignment2/jobSequencing.py"
the max profit job sequence:
['c', 'a', 'e']
['c', 'a', 'e']
['c', 'a', 'e']
['c', 'a', 'e']
['c', 'a', 'e']
```

assignment2 > fractionalKnapsack.py > ...

```python
wt = 50
a=[[60,10],[100,20],[120,30]]
for i in range(len(a)):
    for j in range(i+1,len(a)):
        if(a[i][0]/a[i][1] < a[j][0]/a[j][1]):
            a[i],a[j]=a[j],a[i]

f=0
for i in a:
    if i[1]<=wt:
        wt-=i[1]
        f+=i[0]
    else:
        f+=i[0]*wt/i[1]
print(f)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER GITLENS COMMENTS

```
PS C:\Users\Ikjot singh\Coding\DAA\assignment2> cd "c:/Users/Ikjot singh/Coding/DAA/ass
PS C:\Users\Ikjot singh\Coding\DAA\assignment2> & "C:/Users/Ikjot singh/AppData/Local/Pr
jot singh/Coding/DAA/assignment2/fractionalKnapsack.py"
240.0
PS C:\Users\Ikjot singh\Coding\DAA\assignment2>
```

Description Discussion Submissions Hints

Programming / Greedy Algorithm / Majority Element

Majority Element

Bookmark

Easy 54.5% Success 392 40

Asked In: Microsoft Yahoo Google more >

Problem Description

Given an array of size **N**, find the majority element. The majority element is the element that appears more than **floor(N/2)** times.
You may assume that the array is non-empty and the majority element always exist in the array.

Problem Constraints

$1 <= |A| <= 10^6$
$1 <= A_i <= 10^9$

Time taken: 2 min. Score: 400 / 400 Python 3 (Python-3.8) Old view

```python
class Solution:
    # @param A : tuple of integers
    # @return an integer
    def majorityElement(self, A):
        n = len(A)
        candidate = A[0]
        count = 1

        for i in range(1, n):
            if count == 0:
                candidate = A[i]
                count = 1
            elif A[i] == candidate:
                count += 1
            else:
                count -= 1

        return candidate if A.count(candidate) > n // 2 else None
```

Output Result

Your Answer is correct ! Share

Next question

Programming / Greedy Algorithm / Distribute Candy

Distribute Candy

Bookmark

Medium 55.2% Success 344 13

Asked In: Microsoft Flipkart Amazon

Problem Description

N children are standing in a line. Each child is assigned a **rating** value.

You are giving candies to these children subjected to the following requirements:

1. Each child must have at least one candy.
2. Children with a higher rating get more candies than their neighbors.

What is the minimum number of candies you must give?

Problem Constraints

$1 <= N <= 10^5$

$-10^9 <= A[i] <= 10^9$

Input Format

The first and only argument is an integer array A representing the rating of children.

```python
class Solution:
    # @param A : list of integers
    # @return an integer
    def candy(self, A):
        self.A=A
        n = len(A)
        candies = [1] * n

        for i in range(1, n):
            if A[i] > A[i-1]:
                candies[i] = candies[i-1] + 1

        for i in range(n-2, -1, -1):
            if A[i] > A[i+1]:
                candies[i] = max(candies[i], candies[i+1] + 1)

        return sum(candies)
```

Output Result

Your Answer is correct ! Share

```
1
5 10
8 5 4 3 2
```

```
3
```

Enter your code or Upload your code as file.

Save Python 3 (python 3.9.5)

```python
def max_bottles(n, x, a):
    a.sort()
    k = 0
    for j in a:
        if j <= x:
            x -= j
            k += 1
        else:
            break

    return k

t = int(input())
for i in range(t):
    n, x = map(int, input().split())
    a = list(map(int, input().split()))
    result = max_bottles(n, x, a)
    print(result)
```

11:13 vscode

Test against custom input

Compile & Test code

Submit code

Submission ID: 79460779 / 1 second ago

RESULT: Accepted

Refer judge environment

Statement Submissions Solution

Chefland has a very famous university. The university offers **N** courses. Each course runs for some consecutive range of days. You are given starting and ending days of the ith course by $\mathbf{start_i}$ and $\mathbf{end_i}$, respectively.

Our Chef wanted to enroll himself in the university. But he is not sure about the exact time period for which he wants to study. Though he has **Q** such tentative plans in his mind. Each plan consists of a start date $\mathbf{plan\_start_j}$ and an end date $\mathbf{plan\_end_j}$.

Chef wants your help in finding out the maximum number of courses he can complete during each of his plans. Note that at a time, Chef can not handle multiple courses, i.e. he can attend at most one course during a day. Also, a course will be considered completed only if Chef attends all the classes of the course.

Input

There is a single test case.

The first line of the input contains two space separated integers **N** and **Q** denoting the number of courses university offers and the number of plans Chef has in mind, respectively.

The ith of the next **N** lines contains two space separated integers $\mathbf{start_i}$ and $\mathbf{end_i}$ denoting the starting and the ending day of the ith course.

The jth of the next **Q** lines contains two space separated integers $\mathbf{plan\_start_j}$ and $\mathbf{plan\_end_j}$, denoting the start and the end day of Chef's plan.

Output

Output **Q** lines - each containing an integer corresponding to the maximum number of the courses Chef can complete in the corresponding planned visit.

PYTH 3

```python
def university(start,end,courses):
    count=0
    for i in courses:
        if start<=end and i[0]>=start and i[1]<=end:
            count+=1
            start=i[1]
    return count
t=[int(x) for x in input().split()]
n1,n2=t[0],t[1]
c=[]
visits=[]
result=[]
for i in range(n1):
    c.append([int(x) for x in input().split()])
c.sort(key=lambda x: x[1])
for j in range(n2):
    q=[int(x) for x in input().split()]
    result.append(university(q[0],q[1],c))
for i in result:
    print(i)
```

6:16

Test against Custom Input

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

Statement Submissions Solution

Our chef has recently opened a new restaurant with a unique style. The restaurant is divided into **K** compartments (numbered from **1** to **K**) and each compartment can be occupied by at most one customer.

Each customer that visits the restaurant has a strongly preferred compartment **p** ($1 \le p \le K$), and if that compartment is already occupied, then the customer simply leaves. Now obviously, the chef wants to maximize the total number of customers that dine at his restaurant and so he allows (or disallows) certain customers so as to achieve this task. You are to help him with this.

Given a list of **N** customers with their arrival time, departure time and the preferred compartment, you need to calculate the maximum number of customers that can dine at the restaurant.

Input

The first line contains an integer **T** denoting the number of test cases. Each of the next **T** lines contains two integers **N** and **K** , the number of customers that plan to visit the chef's restaurant and the number of compartments the restaurant is divided into respectively. Each of the next **N** lines contains three integers s_i, f_i and p_i , the arrival time, departure time and the strongly preferred compartment of the ith customer respectively.
Note that the ith customer wants to occupy the p_i^{th} compartment from $[s_i, f_i)$ i.e the ith customer leaves just before f_i so that another customer can occupy that compartment from f_i onwards.

Output

For every test case, print in a single line the maximum number of customers that dine at the restaurant.

Constraints

- $1 \le T \le 30$
- $0 \le N \le 10^5$
- $1 \le K \le 10^9$

PYTH 3

```python
d=int(input())
while d>0:
    n,k=map(int,input().split())
    t={}
    for i in range(n):
        s,f,p=map(int,input().split())
        if p in t:
            t[p].append((f,s))
        else:
            t[p]=[(f,s)]
    count=0
    for i in t:
        t[i].sort()
        start=-1
        for j in t[i]:
            if j[1]>=start:
                count+=1
                start=j[0]
    print(count)

    d-=1
```

20:8

Test against Custom Input

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

Alice gives Bob a board composed of 1×1 wooden squares and asks him to find the minimum cost of breaking the board back down into its individual squares. To break the board down, Bob must make cuts along its horizontal and vertical lines.

To reduce the board to squares, Bob makes horizontal and vertical cuts across the entire board. Each cut has a given cost, $cost\_y[i]$ or $cost\_x[j]$ for each cut along a row or column across one board, so the cost of a cut must be multiplied by the number of segments it crosses. The cost of cutting the whole board down into 1×1 squares is the sum of the costs of each successive cut.

Can you help Bob find the minimum cost? The number may be large, so print the value modulo $10^9 + 7$.

For example, you start with a 2×2 board. There are two cuts to be made at a cost of $cost\_y[1] = 3$ for the horizontal and $cost\_x[1] = 1$ for the vertical. Your first cut is across 1 piece, the whole board. You choose to make the horizontal cut between rows 1 and 2 for a cost of $1 \times 3 = 3$. The second cuts are vertical through the two smaller boards created in step 1 between columns 1 and 2. Their cost is $2 \times 1 = 2$. The total cost is $3 + 2 = 5$ and $5\%(10^9 + 7) = 5$.

Function Description

Complete the boardCutting function in the editor below. It should return an integer.

boardCutting has the following parameter(s):

- cost_x: an array of integers, the costs of vertical cuts
- cost_y: an array of integers, the costs of horizontal cuts

Input Format

The first line contains an integer q, the number of queries.

The following q sets of lines are as follows:

- The first line has two positive space-separated integers m and n, the number of rows and columns in the board.
- The second line contains $m - 1$ space-separated integers cost_y[i], the cost of a horizontal cut between rows $[i]$ and $[i + 1]$ of one board.
- The third line contains $n - 1$ space-separated integers cost_x[j], the cost of a vertical cut between columns $[j]$

Change Theme Language Python 3

```python
# The function accepts following parameters:
#  1. INTEGER_ARRAY cost_y
#  2. INTEGER_ARRAY cost_x
#

def boardCutting(cost_y, cost_x):
    cost_x.sort(reverse=True)
    cost_y.sort(reverse=True)
    minCost = 0
    x, y = 1, 1
    for i in range(len(cost_x) + len(cost_y)):
        if y > len(cost_y) or (x <= len(cost_x) and cost_x[x-1] >= cost_y[y-1]):
            minCost += cost_x[x-1] * y
            x += 1
        else:
            minCost += cost_y[y-1] * x
            y += 1
    return minCost % (10**9 + 7)



if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    q = int(input().strip())

    for q_itr in range(q):
        first_multiple_input = input().rstrip().split()

        m = int(first_multiple_input[0])

        n = int(first_multiple_input[1])

        cost_y = list(map(int, input().rstrip().split()))

        cost_x = list(map(int, input().rstrip().split()))
```