

NAME : B. Harshini

Hall Ticket.No: 2303A52242

Batch:36

Assignment 1: Maximum Non-Overlapping Meetings (Greedy)

Problem Statement

You are given N meetings. Each meeting has a start time S_i and an end time E_i . You want to attend the maximum number of meetings. You can attend meeting j after meeting i only if the start time of meeting j is strictly greater than the end time of meeting i ($S_j > E_i$). For each test case, output the maximum number of meetings that can be attended.

Input Format

The first line contains an integer T, the number of test cases. For each test case:

- The first line contains an integer N.
- The next N lines each contain two integers S_i and E_i .

Output Format: For each test case, print a single integer: the maximum number of meetings that can be attended.

Constraints

- $1 \leq T \leq 20$
- $1 \leq N \leq 200000$ (sum of N over all test cases ≤ 200000)
- $0 \leq S_i < E_i \leq 10^9$

Sample Input

1

3

1 3

2 4

3 5

Expected Output : 2

CODE: (PYTHON)

The screenshot shows a Python code editor on the OnlineGDB platform. The code is a solution to the activity selection problem, which involves selecting the maximum number of non-overlapping activities from a list of activities based on their start and end times.

```
main.py
1 T = int(input())
2 code: compile run debug share.
3 IDE
4 My Projects
5 Classroom
6 Learn Programming
7 Programming Questions
8 Sign Up
9 Login
10
11
12
13
14
15
16
17
18
19
20
21
22
23
T = int(input())
for _ in range(T):
    N = int(input())
    activities = []
    for i in range(N):
        s, e = map(int, input().split())
        activities.append((s, e))
    activities.sort(key=lambda x: x[1])
    count = 1
    last_end = activities[0][1]
    for i in range(1, N):
        if activities[i][0] >= last_end:
            count += 1
            last_end = activities[i][1]
    print(count)
```

The code uses a greedy algorithm to select activities. It first reads the number of test cases (T). For each test case, it reads the number of activities (N) and then reads N lines of activity start and end times (s, e). It sorts the activities by their end times. It then iterates through the sorted activities, starting with the first one. For each subsequent activity, if its start time is greater than or equal to the end time of the previous selected activity, it is selected and its end time becomes the new last_end. Finally, it prints the total count of selected activities.

Input:

```
5
3
4
5
2
```

Output:

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

CODE: (JAVA)

The screenshot shows the OnlineGDB IDE interface. The code editor displays a Java file named Main.java with the following content:

```

1 import java.util.*;
2
3 public class Main {
4     static class Activity {
5         int start, end;
6         Activity(int s, int e) {
7             start = s;
8             end = e;
9         }
10    }
11
12    public static void main(String[] args) {
13        Scanner sc = new Scanner(System.in);
14
15        int T = sc.nextInt();
16
17        while (T-- > 0) {
18            int N = sc.nextInt();
19            Activity[] activities = new Activity[N];
20
21            for (int i = 0; i < N; i++) {
22                int s = sc.nextInt();
23                int e = sc.nextInt();
24                activities[i] = new Activity(s, e);
25            }
26
27            Arrays.sort(activities, (a, b) -> a.end - b.end);
28
29            int count = 1;
30            int lastEnd = activities[0].end;
31
32            for (int i = 1; i < N; i++) {
33                if (activities[i].start >= lastEnd) {
34                    count++;
35                }
36            }
37
38        }
39
40        System.out.println(count);
41
42    }
43
44}

```

The input window contains the following integers:

```

1 3
1 3
2 4
3 5
2

```

The output window shows the result:

```

...Program finished with exit code 0
press ENTER to exit console.

```

Assignment-1: Shelf Product Lookup (Divide and Conquer)

Problem Statement

A supermarket maintains a sorted list of product IDs. For each query ID X, determine how many times X appears in the list. You must solve using binary-search style divide and conquer (find first occurrence and last occurrence).

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

- First line: N Q
- Second line: N integers (sorted product IDs)
- Next Q lines: integer X (query ID)

Output Format For each query, print the count of X on a new line.

Constraints

- $1 \leq T \leq 20$
- $1 \leq N \leq 200000$ (sum of N over all test cases ≤ 200000)
- $1 \leq Q \leq 200000$ (sum of Q over all test cases ≤ 200000)
- Product IDs fit in 32-bit signed integer

Sample Input

1

8 3

101 101 102 102 105 110 110 101 102 999

Expected Output

230

Python code:

The screenshot shows a Python code editor interface with a toolbar at the top featuring icons for Run, Debug, Stop, Share, Save, and Beautify. The file tab shows 'main.py'. The code itself consists of two functions: 'first_occurrence' and 'last_occurrence'. Both functions take an array 'arr' and a value 'x' as parameters. They use binary search to find the first or last occurrence of 'x' respectively. If found, they return the index; if not found, they return -1.

```
1 def first_occurrence(arr, x):
2     l, r = 0, len(arr) - 1
3     res = -1
4     while l <= r:
5         m = (l + r) // 2
6         if arr[m] == x:
7             res = m
8             r = m - 1
9         elif arr[m] < x:
10            l = m + 1
11        else:
12            r = m - 1
13    return res
14
15 def last_occurrence(arr, x):
16     l, r = 0, len(arr) - 1
17     res = -1
18     while l <= r:
19         m = (l + r) // 2
20         if arr[m] == x:
21             res = m
22             l = m + 1
23         elif arr[m] < x:
24             l = m + 1
```

```

8         r = m - 1
9     elif arr[m] < x:
10        l = m + 1
11    else:
12        r = m - 1
13    return res
14 def last_occurrence(arr, x):
15     l, r = 0, len(arr) - 1
16     res = -1
17     while l <= r:
18         m = (l + r) // 2
19         if arr[m] == x:
20             res = m
21             l = m + 1
22         elif arr[m] < x:
23             l = m + 1
24         else:
25             r = m - 1
26     return res
27 t = int(input())
28 for _ in range(t):
29     n, q = map(int, input().split())
30     data = list(map(int, input().split()))
31     arr = data[:n]          # product IDs
32     queries = data[n:n+q]   # query IDs
33
34     result = ""
35     for x in queries:
36         f = first_occurrence(arr, x)
37         if f == -1:
38             result += "0"
39         else:
40             l = last_occurrence(arr, x)
41             result += str(l - f + 1)
42
43 print(result)

```

Output:

```

input
1
8 3
101 101 102 102 105 110 110 101 102 999
230

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

```

Java Code:

```
1+ import java.util.*;
2+ class Main {
3+
4+     static int firstOcc(int[] arr, int x) {
5+         int l = 0, r = arr.length - 1, res = -1;
6+         while (l <= r) {
7+             int m = (l + r) / 2;
8+             if (arr[m] == x) {
9+                 res = m;
10+                r = m - 1;
11+            } else if (arr[m] < x) {
12+                l = m + 1;
13+            } else {
14+                r = m - 1;
15+            }
16+        }
17+        return res;
18+    }
19+
20+    static int lastOcc(int[] arr, int x) {
21+        int l = 0, r = arr.length - 1, res = -1;
22+        while (l <= r) {
23+            int m = (l + r) / 2;
24+            if (arr[m] == x) {
25+                res = m;
26+                l = m + 1;
27+            } else if (arr[m] < x) {
28+                l = m + 1;
29+            } else {
30+                r = m - 1;
31+            }
32+        }
33+        return res;
34+    }
35+
36+    public static void main(String[] args) {
37+        Scanner sc = new Scanner(System.in);
38+
39+        int t = sc.nextInt();
40+        while (t-- > 0) {
41+            int n = sc.nextInt();
42+            int q = sc.nextInt();
43+
44+            int[] data = new int[n + q];
45+            for (int i = 0; i < n + q; i++)
46+                data[i] = sc.nextInt();
47+
48+            int[] arr = Arrays.copyOfRange(data, 0, n);
49+
50+            StringBuilder output = new StringBuilder();
51+
52+            for (int i = n; i < n + q; i++) {
53+                int x = data[i];
54+                int f = firstOcc(arr, x);
55+                if (f == -1)
56+                    output.append("0");
57+                else
58+                    output.append(lastOcc(arr, x) - f + 1);
59+            }
60+
61+            System.out.println(output.toString());
62+        }
63+        sc.close();
64+    }
65+}
```

```
1+ import java.util.*;
2+ class Main {
3+
4+     static int firstOcc(int[] arr, int x) {
5+         int l = 0, r = arr.length - 1, res = -1;
6+         while (l <= r) {
7+             int m = (l + r) / 2;
8+             if (arr[m] == x) {
9+                 res = m;
10+                r = m - 1;
11+            } else if (arr[m] < x) {
12+                l = m + 1;
13+            } else {
14+                r = m - 1;
15+            }
16+        }
17+        return res;
18+    }
19+
20+    static int lastOcc(int[] arr, int x) {
21+        int l = 0, r = arr.length - 1, res = -1;
22+        while (l <= r) {
23+            int m = (l + r) / 2;
24+            if (arr[m] == x) {
25+                res = m;
26+                l = m + 1;
27+            } else if (arr[m] < x) {
28+                l = m + 1;
29+            } else {
30+                r = m - 1;
31+            }
32+        }
33+        return res;
34+    }
35+
36+    public static void main(String[] args) {
37+        Scanner sc = new Scanner(System.in);
38+
39+        int t = sc.nextInt();
40+        while (t-- > 0) {
41+            int n = sc.nextInt();
42+            int q = sc.nextInt();
43+
44+            int[] data = new int[n + q];
45+            for (int i = 0; i < n + q; i++)
46+                data[i] = sc.nextInt();
47+
48+            int[] arr = Arrays.copyOfRange(data, 0, n);
49+
50+            StringBuilder output = new StringBuilder();
51+
52+            for (int i = n; i < n + q; i++) {
53+                int x = data[i];
54+                int f = firstOcc(arr, x);
55+                if (f == -1)
56+                    output.append("0");
57+                else
58+                    output.append(lastOcc(arr, x) - f + 1);
59+            }
60+
61+            System.out.println(output.toString());
62+        }
63+        sc.close();
64+    }
65+}
```

Output:



A screenshot of a terminal window titled "input". The window contains the following text:

```
1
8 3
101 101 102 102 105 110 110 101 102 999
230

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