

ASSIGNMENT-1

Name: M.nishitha

H NO: 2303A51540

Batch - 19

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

The screenshot shows a Java application running in a debugger. The code editor displays a Java file named Main.java with the following content:

```
A_1.java > Main > main(String[] args)
1 import java.util.*;
2 class Main {
3     static class Meeting {
4         int start, end;
5         Meeting(int s, int e) {
6             start = s;
7             end = e;
8         }
9     }
10    public static void main(String[] args) {
11        Scanner sc = new Scanner(System.in);
12        int T = sc.nextInt();
13        while (T-- > 0) {
14            int N = sc.nextInt();
15            Meeting[] meetings = new Meeting[N];
16            for (int i = 0; i < N; i++) {
17                meetings[i] = new Meeting(sc.nextInt(), sc.nextInt());
18            }
19            Arrays.sort(meetings, (a, b) -> Integer.compare(a.end, b.end));
20            int count = 0;
21            int lastEnd = -1;
22            for (Meeting m : meetings) {
23                if (m.start >= lastEnd) {
24                    count++;
25                    lastEnd = m.end;
26                }
27            }
28        }
29        System.out.println(count);
30    }
31    sc.close();
32 }
33 }
```

The call stack pane shows several threads running, including "Attach Listener", "Finalizer", "Reference Handler", "Signal Dispatcher", "Notification Thread", and "Common-Cleaner". The debug console pane shows the output of the program:

```
17°C Mostly sunny
Launch Java App (COMPETITIVE PROGRAMMING) User program finished
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
```

The terminal pane shows the execution of the Java program:

```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Listening on 59384
User program running
→ 1
→ 3
→ 1 3
→ 2 4
→ 3 5
User program finished
2
```

