

# ASSIGNMENT-1

Jukanti lohith

2303A51528

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  $\leq 200000$ )
- $0 \leq S_i < E_i \leq 10^9$

### Sample Input

1

3

1 3

2 4

3 5

Expected Output 2

# ASSIGNMENT-1

The screenshot shows an IDE window titled "COMPUTITIVE PROGRAMMING". The code is in Java and defines a `Meeting` class and a `Main` class. The `Meeting` class has attributes `start` and `end`, and a constructor `Meeting(int s, int e)`. The `Main` class has a `main` method that reads input from `System.in`, creates an array of `Meeting` objects, sorts them by end time, and counts the number of non-overlapping meetings.

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

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

## ASSIGNMENT-1