D. Recommendations

time limit per test: 2 seconds

memory limit per test: 512 megabytes

Suppose you are working in some audio streaming service. The service has n active users and 10^9 tracks users can listen to. Users can like tracks and, based on likes, the service should recommend them new tracks.

Tracks are numbered from 1 to 10^9. It turned out that tracks the i-th user likes form a segment $[l_i, r_i]$.

Let's say that the user j is a *predictor* for user i ($j \neq i$) if user j likes all tracks the i-th user likes (and, possibly, some other tracks too).

Also, let's say that a track is *strongly recommended* for user i if the track is not liked by the i-th user yet, but it is liked by **every** predictor for the i-th user.

Calculate the number of strongly recommended tracks for each user i. If a user doesn't have any predictors, then print 0 for that user.

Input

The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases. Next, t cases follow.

The first line of each test case contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of users.

The next n lines contain two integers l_i and r_i per line ($1 \le l_i \le r_i \le 10^9$) — the segment of tracks the i-th user likes.

Additional constraint on the input: the sum of n over all test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case, print n integers, where the i-th integer is the number of strongly recommended tracks for the i-th user (or 0, if that user doesn't have any predictors).

Example

input

```
4
3
3 8
2 5
4 5
2
42 42
1 1000000000
3
42 42
1 1000000000
42 42
6
1 10
3 10
3 7
5 7
4 4
1 2
```

output

```
0
0
1
999999999
0
0
0
0
0
0
```

```
2
3
2
4
8
```

Note

In the first test case:

- the first user has no predictors;
- the second user has no predictors;
- the third user has two predictors: users 1 and 2; only track 3 is liked by both of them and not liked by the third user.

In the second test case, the second user is a predictor for the first user. Therefore, all tracks, except 42, are strongly recommended for the first user.

In the third test case, the first user has two predictors: users 2 and 3, but there is no track that is liked by them and not liked by the first user himself.