

# Recitation 1

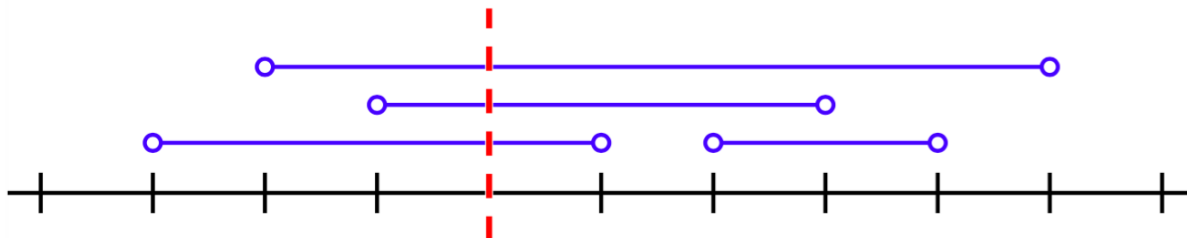
## Augmented Tables

### 1.1 Announcements

- *RangeLab* has been released, and is due *Friday afternoon*.
- *BridgeLab* will be released on Friday. The written portion will be due the following Friday, while the coding portion will be due the Monday after that.

## 1.2 Interval Checking

Suppose you're given a set of intervals  $I \subset \mathbb{Z} \times \mathbb{Z}$  and some  $k \in \mathbb{Z}$ , and you're interested in determining whether or not there exists  $(l, r) \in I$  such that  $l < k < r$ . For simplicity, let's assume that no two intervals share an endpoint.



**Task 1.1.** *Implement a function*

```
val intervalCheck : (int * int) Seq.t → int → bool
```

where `(intervalCheck I k)` answers the query mentioned above. Your function must be staged such that the line

```
val q = intervalCheck I
```

performs  $O(|I| \log |I|)$  work and  $O(\log^2 |I|)$  span, while each subsequent call  $q(k)$  only performs  $O(\log |I|)$  work and span. Try solving this problem with augmented tables.

## 1.3 Interval Counting

Now suppose you want to solve a more general problem. Given  $I$  and  $k$ , you want to return  $|\{(l, r) \in I \mid l < k < r\}|$ . Once again, for simplicity, we'll assume all endpoints are distinct.

**Task 1.2.** *Implement a function*

```
val intervalCount : (int * int) Seq.t → int → int
```

*where (intervalCheck I k) answers the interval counting query as mentioned above. Your function must be staged, just like Task ??.*