HandsOn 2

November 7, 2024

2024 · # rust, # algorithms, # data-structures · notes

This is the text of second hands-on of the the course *Competitive Programming and Contests* at University of Pisa in the accademic year 2024-25.

The objective of this hands-on is to implement and play with Segment Trees in Rust. There are two problems to solve.

Problem #1: Min and Max

You are given an array $A[1, n]$ of n positive integers, each integer is at most n. You have to build a data structure to answer two different types of queries:

- `Update(i, j, T)` that replaces every value $A[k]$ with $\min(A[k], T)$, where $i \leq k \leq j$;
- `Max(i, j)` that returns the largest value in $A[i \dots j]$.

You are also given m of these queries to solve. The target solution must run in $O((n + m) \log n)$ time.

Here we have a set of tests with the following format.

Input

The first line contains n and m. The next line contains the n integers in A. Each of the subsequent m lines contains the query. The first value of each line is either 0 (query `Update`) or 1 (query `Max`). For a query `Update` the values of i, j, and T follows. For a query `Max` the values of i and j follows.

Output

Results of `Max` queries.

Example

```
5 3       // n m
5 1 4 3 2 // The array A
0 1 2 2   // Update(1, 2, 2). The array A becomes 2 1 4 3 2.
1 2 4     // Max(2, 4) = 4
1 1 2     // Max(1, 2) = 2
```

The output is

```
4
2
```

Problem #2: Is There

You are given n segments. A segment $\langle l, r \rangle$ is such that $0 \leq l \leq r \leq n - 1$. Then, you are given m queries `IsThere`.

A query `IsThere(i,j,k)` has to return 1 if there exists a position p, with $0 \leq i \leq p \leq j \leq n - 1$, such that *exactly* k segments contain position p, 0 otherwise.

The solution must run in $O((n + m) \log n)$ time.

Here we have a set of tests with the following format.

Input

The first line contains n and m. Each of the next n lines contains a pair integers $\langle l, r \rangle$, one for each segment. Finally, there will be m lines, one for each query. Each of these lines contains i, j and k, separated by a space.

Output

The result of each query in input order.

Example

```
5 4    // n m
0 4    // segments
1 3
1 2
1 1
0 0
0 4 4   // i j k
0 4 0
1 3 1
1 4 1
```

The output is

```
1
0
0
1
```

Submission

Submit the files `main.rs` and `lib.rs` and a file `Handson_2_solution_YOUR_NAME.pdf` to rossano.venturini@gmail.com by 24/11/2024.

- Source code `main.rs` and `lib.rs` contain your implementations.
- A report `Handson_2_solution_YOUR_NAME.pdf` that briefly describes your solutions, your implementations, and an analysis of their time and space complexities. Add references to any relevant source you consulted to find your solutions or to develop their implementations.

Before submitting your solutions,

- make sure your implementation successfully passes all the tests.
- use `cargo fmt` to format your code.
- use `cargo clippy` to check your code.

Cheating

Very important! You are allowed to verbally discuss solutions with other students, **BUT** you must implement all solutions by yourself. Therefore, sharing implementations with others is strictly **forbidden**.

Enjoy Reading This Article?

Here are some more articles you might like to read next:

- HandsOn 3
- HandsOn 1 24/25
- Mo's Algorithm
- Dynamic Prefix Sums with Fenwick Tree
- The Power of Prefix Sums

0 reactions

0 comments

Write Preview Aa

Sign in to comment

Sign in with GitHub