UNIVERSITÀ DI PISA
Dipartimento di Informatica
Corso di Laurea in Informatica

Assignment 02

Competitive Programming and Contests

Prof. Rossano Venturini

Giacomo Trapani - 600124

Academic Year 2024/2025

Exercise 1 This exercise requires the implementation of a data structure on which - given an input array `A[1; n]` - the following kind of queries take $O((n+m)\log n)$ time:

- `update(i, j, T)` that replaces every value `A[k]` with $\min(A[k], T)$ with $k \in [i; j]$,
- `max(i, j)` that returns the largest value in `A[i..j]`.

The data structure of choice are segment trees with lazy propagation, for which the implementation given is vector-based.

First, we use the array `A` to build the segment tree `T` (ref. to method `from_vector`). The idea behind the algorithms for both `update` and `max` is to iterate through the nodes of `T`; each node represents an interval which may partially or totally overlap with `(i, j)`, or not overlap at all. For each of these cases, the algorithms work in similar ways, refer to `rec_update` and `rec_max`.

Exercise 2 This exercise requires the implementation of a data structure on which - given `n` intervals (which are referred to as "segments") - it is possible to answer `m` queries of the following kind in $O((n+m)\log n)$ time:

- `is_there(i, j, k)` that returns 1 if there exists a position $p \in [i; j]$ s.t. exactly k segments contain p, 0 otherwise.

The data structure of choice are segment trees with lazy propagation, for which the implementation given is vector-based.

The algorithm implemented works in a way similar to that of the ones in the first exercise, refer to `rec_is_there`.

How to run The code can be run via `cargo run`. The following directory structure is expected

```
src
├── lib.rs
└── main.rs
tests-1
tests-2
Cargo.toml
Cargo.lock
```