

## Advanced Data Structures

Spring Semester 2017

### Exercise Set 3

#### Exercise 1:

##### *Indirection*

Recall a problem of *longest common subsequence* or equivalent problem of *edit distance* of two input strings  $u, v$  of length  $n$ . Classical DP solution works in time  $\mathcal{O}(n^2)$ . Show that it can be done in time  $\mathcal{O}(n^2/\log^c n)$  for some (small) constant  $c > 0$ .

##### *Hint:*

DP solution works by filling 2d array with integers. This can be speed-up by cutting the table into square tiles of polylog size. Tile takes its top and left borders, and we only care about its bottom and right borders. However, we need parametrization such that *number* of tiles of size  $x$  grows only as a function of  $x$ , and is independent from  $|\Sigma|$  or  $n$ .

Below, if stated otherwise, we consider arbitrary *associative operation*  $\circ$ , and *product queries* of form  $x_i \circ x_{i+1} \circ \dots \circ x_j$ .

#### Exercise 2:

Show the “trivial” solution from lecture to the product queries in arrays problem: given array on  $n$  elements, precompute it in  $\mathcal{O}(n)$  time and space to answer product queries in  $\mathcal{O}(\log n)$  time.

#### Exercise 3:

Show a solution to Exercise 2 that works in a dynamic setting (supports assignments to any  $x_i$  in time  $\mathcal{O}(\log n)$ ).

#### Exercise 4:

Consider (static) product queries on trees: every node holds value, we query for product of all values on a path between two given vertices. Show that we can preprocess tree in  $\mathcal{O}(n \log n)$  time and space, to support queries for path product in  $\mathcal{O}(\log n)$  time.

##### *Hint:*

There are two equivalent approaches here:

- (i) every node stores some auxiliary values regarding jumps towards the root of certain length
- (ii) every node stores range tree from Exercise 2, for all values between itself and root. Range tree of node differs from range tree of its parent on only few values, so it is enough to use persistence...

#### Exercise 5:

Show that solution to Exercise 4 can be tweaked to have min queries on trees in  $\mathcal{O}(1)$  time.

***Hint:***

Use idempotency of min, that is  $\min(x, x) = x$ .

**Exercise 6:**

Consider static data structure for product queries in arrays, where each query is answered accessing *at most*  $\ell$  cells with preprocessed data. Show, that input can be preprocessed in time:  $\mathcal{O}(n^2)$ ,  $\mathcal{O}(n \log n)$ ,  $\mathcal{O}(n \log \log n)$  and  $\mathcal{O}(n \log^* n)$  for  $\ell = 1, 2, 3$  and 4 respectively.

***Hint:***

For  $\ell = 2$ , you only need to “massage” previous solutions to have 2 memory cells accessed, instead of  $\mathcal{O}(1)$ . For  $\ell > 2$ , it is enough to consider partitioning the array of length  $n$  into segments of particular size. In each segment store all prefix and suffix products. Each query either falls fully into a single segment, or using one stored prefix and one suffix can be aligned with segments (and the solved with solution for  $\ell - 2$ ). Fill in the details, guess proper value of  $x$  for  $\ell = 3, 4$  and solve the recursion.