Exercise sheet 5.

Data structures SoSe 2020

László Kozma, Katharina Klost

Due 12:00, May 29th, 2020

Exercise 1 Cartesian trees $4 \times 2 + 2 \times 3$ *Points*

Recall the Cartesian tree (treap) built from an array A, using indices $1, \ldots, n$ as search keys, and the corresponding array entries $A[1], \ldots, A[n]$ as min-heap priorities.

(a) Which Cartesian tree corresponds to the array $[1, 2, \ldots, n]$? What about the array $[1, 3, 5, \ldots, 2n+1, 2, 4, 6, \ldots, 2n]$?

(b) Give an array of size 15 whose Cartesian tree is a complete balanced tree.

(c) A *left-to-right* minimum of an array is an entry that is smaller than all preceding entries. Show how to find the left-to-right minima of an array A in the Cartesian tree of A.

(d) Let $rmq(i, j)$ denote the index of the smallest entry in $A[i \ldots j]$, and let $lca(i, j)$ denote the lowest common ancestor of i and j in the Cartesian tree of A. Show that $lca(i, j) = rmq(i, j)$. (We sketched this in the lecture.)

(e) Suppose now that the array A contains a permutation of the integers $1, \ldots, n$ chosen uniformly at random from the set of all permutations. Let T be the Cartesian tree of A. Given i and j, what is the probability (depending on i and j) that node $(i, A[i])$ is the ancestor of node $(j, A[j])$?

Hint: x is the ancestor of y if and only if $x = lca(x, y)$. Use the correspondence between lca and rmq.

(f) Show that in the Cartesian tree of the previous question the expected depth of every node is $O(\log n)$.

Hint: Use the result of the previous question. The depth of a node is the number of ancestors it has.

Exercise 2 Range sum queries *4 Points*

Suppose we want to preprocess an array A of size n such as to be able to answer queries *range-sum*(i, j), returning the sum $A[i] + A[i+1] + \cdots + A[j]$. We have seen how to do this with $O(n)$ preprocessing and $O(1)$ query time, using subtraction. (We just compute all prefix sums of the array.)

Suppose now that we are not allowed to do subtractions, only *one addition* per query. Show that we can store $O(n \log n)$ well-chosen partial sums, so that each range-sum query can be answered with a single addition (i.e. by adding together two of the stored partial sums).

Example: If A is of size 4, then it is sufficient to store $A[1]$, $A[2]$, $A[3]$, $A[4]$, $A[1] + A[2]$, and $A[3] + A[4]$, and from these six partial sums an arbitrary range sum can be computed with a single addition.

Bonus question +5p: Suppose now that we can do k additions for each query (for some constant $k > 1$). How much can you reduce the number of partial sums that need to be stored?

Exercise 3 Ancestors in a tree — 4 *Points*

We are given a fixed tree of size n with nodes indexed 1 to n. Design a data structure that stores one *label* for each node, so that for given i and j we can decide whether i is the ancestor of j, just by examining the labels of i and j. Think of labels as cells of an array $A[1], \dots, A[n]$, where the data structure can answer a query (i, j) by examining the cells $A[i]$ and $A[j]$, but no other cells. We would like to use as little space as possible. Show that $2 \log n$ bits per label are sufficient.

Total: 22 points. Have fun with the solutions!