Mandatory Exercise: Predecessor, RMQ and LCA

Philip Bille Inge Li Gørtz

1 Heap Jumping Consider the following problem. Let T be a perfectly balanced binary tree with n nodes. Each node v in T is numbered with a distinct integer $i(v)$ from the range $[0,\ldots,n^3]$. The numbers are *heap-ordered*, that is, for any internal node v with children v_l and v_r, we have that $i(v) < i(v_l)$ and $i(v) < i(v_r)$. In particular, along any root-to-leaf path p the numbers on the path are strictly increasing.

Given a leaf node ℓ and an integer x, the *heap jump query* is defined as follows.

- heap-jump(ℓ, x): return the largest numbered node with number smaller than x on the path from the root to the leaf ℓ.

Given a tree T as above, the heap jumping problem is to preprocess T into a compact data structure that supports heap jump queries.

Solve the following exercises.

1.1 Give a data structure for the heap jumping problem that answer queries in $O(\log n)$ time and uses linear space.

1.2 Give a data structure for the heap jumping problem that answers queries in $O(\log\log n)$ time and $O(n\log n)$ space.

1.3 Give a data structure for the heap jumping problem that answers queries in $O(\log\log n)$ time and $O(n)$ space.

Ignore preprocessing in all of the exercises.