Assignment 2 - Question 2

due Monday, February 1, 2021

(from Er) Suppose we want to maintain an array \(X[1\ldots n]\) of bits, which are all initially zero, subject to the following operations.

- **LOOKUP\(i\)**: Given an index \(i\), return \(X[i]\).
- **BLACKEN\(i\)**: Given an index \(i < n\), set \(X[i] \leftarrow 1\).
- **NEXTWHITE\(i\)**: Given an index \(i\), return the smallest index \(j \geq i\) such that \(X[j] = 0\). (Because we never change \(X[n]\), such an index always exists.)

If we use the array \(X[1\ldots n]\) itself as the only data structure, it is trivial to implement **LOOKUP** and **BLACKEN** in \(O(1)\) time and **NEXTWHITE** in \(O(n)\) time. But you can do better! Describe data structures that support **LOOKUP** in \(O(1)\) worst-case time and the other two operations in the following time bounds. (We want a different data structure for each set of time bounds, not one data structure that satisfies all bounds simultaneously!)

(a) The worst-case time for both **BLACKEN** and **NEXTWHITE** is \(O(\log n)\).

(d) The worst-case time for **BLACKEN** is \(O(1)\), and the amortized time for **NEXTWHITE** is \(O(\alpha(n))\).

(Hints)

- (a) think of a self-balancing search tree
- (a) you may need the **SUCCESSOR** function
- (d) \(\alpha(n)\) can be replaced by \(\lg^* n\)
- (d) the amortized bound did not depend on the **UNION** function being done by-rank
- (d) there is no **WHITEN**.