Assignment 2 - Question 2

due Friday, February 28, 2022

(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 $\log^* n$
- (d) the amortized bound did not depend on the **Union** function being done *by-rank*
- (d) there is no **Whiten**.