Problem Set 8.5

Data Structures and Algorithms, Fall 2022

Due: November 10.

Problem 1

Consider the following modified algorithm for incrementing a binary counter.

```
INC(B[0, ···])
1: i ← 0.
2: while (B[i] = 1) do
3:     B[i] ← 0.
4:     i ← i + 1.
5: B[i] ← 1.
6: SOMETHINGELSE(i).
```

The only difference from the standard algorithm is the function call at the end, to a black-box subroutine called SOMETHINGELSE. Suppose we call INC n times, starting with a counter with value 0. The amortized time of each INC clearly depends on the running time of SOMETHINGELSE. Let $T(i)$ denote the worst-case running time of SOMETHINGELSE(i). For example, we proved in class that the INC algorithm runs in $O(1)$ amortized time when $T(i) = 0$.

(a) What is the amortized time per INC if $T(i) = 4$? Prove your answer.

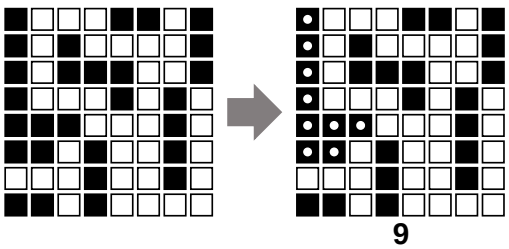
(b) What is the amortized time per INC if $T(i) = 2^i$? Prove your answer.

(c) What is the amortized time per INC if $T(i) = 4^i$? Prove your answer.

Problem 2

Describe and analyze an algorithm to compute the size of the largest connected component of black pixels in an $n \times n$ bitmap $B[1 \cdots n, 1 \cdots n]$. You should try to come up with an algorithm as fast as possible, and you need to analyze the runtime of your algorithm.

For example, given the bitmap below as input, your algorithm should return the number 9, because the largest connected black component (marked with white dots on the right) contains 9 pixels.



Problem 3

Suppose we want to maintain an array $X[1, \cdots, n]$ 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 $X[j] = 0$. (Since we never change $X[n]$, such an index always exists.)

If we use the array $X[1, \cdots, n]$ 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 in the following time bounds.

(a) The amortized time for BLACKEN is $O(\log n)$, and the worst-case time for NEXTWHITE is $O(1)$. You need to argue why your proposed solution achieves these time complexities.

(b) The worst-case time for BLACKEN is $O(\log n)$, and the amortized time for NEXTWHITE is $O(\log^* n)$. You need to argue why your proposed solution achieves these time complexities.

(c) [Bonus Question] BLACKEN in $O(1)$ worst-case, and NEXTWHITE in $O(\log^* n)$ amortized. You need to argue why your proposed solution achieves these time complexities.