CS 240E: Structures and Data Management **Winter 2021**

Tutorial 4: Amortized Analysis and Skip Lists

1. A *binary n-bit counter* stores the current value of a counter as an array A of length n that contains 0 or 1. It supports the operation *Increment*, which adds 1 to the counter and operates as shown below:

```
void Increment(A, n) {
// A is an n-bit counter whose
// value is less than 2^n - 1
    i <- 1
    while (A[n-i] != 0) {
          A[n-i] <- 0
          i <- i + 1
    }
    A[n-i] <- 1

}
```

The running time for $Increment(A, n)$ is $\Theta(k)$, where k is the final value of variable i. This is $\Theta(n)$ in the worst case. Argue that the *amortised* cost of $Increment(A, n)$ is $\Theta(1)$

2. Show that for a skip list with n keys, the probability that the height exceeds $3 \log n$ is at most $1/n^2$.

3. In this problem, we will explore an alternate implementation of a min-ordered priority queue. That is, implement a data structure such that inserting a new element into the priority queue takes $O(\log n)$ expected time, while deleting the minimum element from the priority queue takes $O(1)$ expected time. Hint: use skip lists.

4. Optional. Recall that a binary search tree is called *perfectly balanced* if for every node v we have

$$|v.left.size - v.right.size| \leq 1,$$

i.e., the size-difference between the left and right is as small as possible. Show that in any perfectly balanced binary search tree T, the leaves are only on the bottom two levels.

Hint: First consider the case where $n = 2^k - 1$ for some integer k. Then consider the case where $n = 2^k$ for some integer k. Finally for arbitrary n, let let k be the integer with $2^k \leq n < 2^{k+1}$. In all three cases, what are the sizes of the subtrees, and hence where are the leaves, relative to k?