CS 561, HW2

Prof. Jared Saia, University of New Mexico

Due: Sept. 25th

Please use no outside references in solving these problems. Feel free to work in groups but do note who your collaborators are, and write up all solutions alone.

1. In this problem you will use Chernoff bounds to show that for most of the levels of a skip list, the size of the level is very tightly bounded around its expectation.

   Chernoff Bounds: Assume you have n independent, indicator random variables $X_1, X_2, ..., X_n$ and let $X = \sum_{i=1}^{n} X_i$ and $\mu = E(X)$. Then Chernoff bounds tell us that for any $0 \leq \delta \leq 1$:

   $Pr(X \leq (1-\delta)\mu \text{ or } X \geq (1+\delta)\mu) \leq 2e^{-\mu\delta^2/4}$

   Use Chernoff and Union bounds to show that with probability at least $1 - 1/n$, for all $0 \leq j \leq \log n - \log\log n - 5$, List j in a skip list contains between $n/2^{j+1}$ and $3n/2^{j+1}$ nodes. Let List 0 be the bottom list, List 1 be the next higher up, etc. You may assume that n is sufficiently large, e.g. n is larger than some constant n_0.

   Note: Chernoff bounds are more powerful than bounds on Binomial distributions since X need not be binomially distributed (although it can be). The only requirement is that the X_i be independent. Hint: Remember that $e^{-x} \leq 2^{-x}$.

2. Exercise 11.2-1: Suppose we use a hash function h to hash n distinct keys into an array T of length m. Assuming a uniform hash function, what is the expected number of collisions? More precisely, what is the expected cardinality of $\{\{k, \ell\}\} : k \neq \ell$ and $h(k) = h(\ell)\}$

3. Consider the recurrence $T(n) = 3T(n/4) + \log^2 n$

   (a) Use the Master method to solve this recurrence

(b) Now use annihilators (and a transformation) to solve the recurrence. Show your work. (This is perhaps stating the obvious, but please note that your two bounds should match)

4. Consider the following function:

```
int f (int n){
  if (n==0) return 3;
  else if (n==1) return 5;
  else{
    int val = 3*f (n-1);
    val = val - 2*f (n-2);
    return val;
  }
}
```

   (a) Write a recurrence relation for the *value* returned by f. Solve the recurrence exactly. (Don't forget to check it)

   (b) Write a recurrence relation for the *running time* of f. Get a tight upperbound (i.e. big-O) on the solution to this recurrence.

5. Problem 7-3 *Stooge-Sort*

6. Where in a max-heap might the smallest element reside assuming that all elements are distinct?

7. Is an array that is in sorted order a min-heap?

8. Exercise 6.4-2: Argue the correctness of heapsort

9. Exercise 6.5-5: Argue the correctness of heap-increase-key

10. Problem 6-3: Young Tableaus