CSCI 3104 **Prof. Aaron Clauset**
Problem Set 6 **Spring 2014, CU-Boulder**

1. (25 pts) Complete the following table for the listed implementations of a dictionary ADT. You will need to look up each of these implementations yourself and determine the running times for each operation, in each case. Then, explain why hash tables become poor choices relative to these alternatives when $\alpha = \omega(\log n)$.

	average case			worst case		
implementation	`Add(x)`	`Find(x)`	`Remove(x)`	`Add(x)`	`Find(x)`	`Remove(x)`
hash table	$O(1+\alpha)$	$O(1+\alpha)$	$O(1+\alpha)$	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$
splay tree						
skip list						
AVL tree						
red-black tree						

2. (50 pts total) Acme Corp. now wants Prof. Flitwick to help them develop a new email application, with a user-specific "blacklist" of n messages that are spam.

   Here's what Acme wants. When a new message arrives in a user's mail queue, the app needs to test whether the message is in the user's blacklist. (If so, it will then mark it as spam.) However, space is at a premium and Acme doesn't want to literally store all n spam messages in memory (assume each message is many bits in length).

   Flitwick's proposal is to use a *Bloom filter*,[1] a probabilistic data structure with k hash functions that can be used to test whether an element is in a set. Assume that each hash function uses an its own binary array of length ℓ, can map a string of arbitrary length (an email) to some bit within this array, and is a uniform hash. Under this scheme, a *false positive* occurs when a non-spam message is incorrectly labeled as spam, i.e., is hashed to the same location as one of the n spam messages.

   (a) (15 pts) For a false positive rate of no more than $f = 1/100$, what is the minimum number of bits of space needed to guarantee that Flitwick can test in $O(1)$ time whether a new message is in the blacklist?
   Hint: Recall that $1 - x \approx \mathrm{e}^{-x}$ when $x \ll 1$.

   (b) (10 pts) Assume that Flitwick uses $k = \Theta(1)$ hash functions. Using asymptotic notation, how many bits will Flitwick need if he wants the probability of false positives to be no more than $f = 1/n$?

   (c) (15 pts) Assume that Flitwick uses $k = \Theta(\lg n)$ hash functions. Using asymptotic notation, how many bits will he need if he wants the probability of false positives to be no more than $f = 1/n$? Explain why this answer differs from that of 2b.

[1]See `http://en.wikipedia.org/wiki/Bloom_filter`

(d) (10 pts) Suppose the adversary knows that Acme is using Flitwick's scheme, but does not have access to the hash functions being used. Describe two sequences of emails, one that would defeat the Bloom filter approach by rendering the system unusable and one that would defeat it by getting spam messages through the filter. Explain why each of your sequences always succeeds.

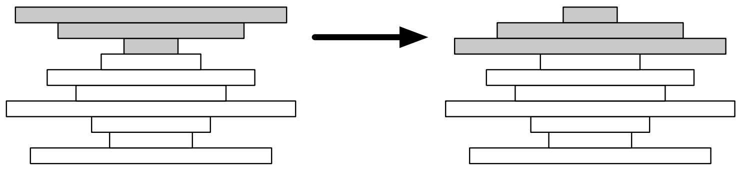
3. (25 pts) Prof. Dumbledore gives you an array $A[1 \ldots n]$ with the special property that $A[1] \geq A[2]$ and $A[n-1] \leq A[n]$. He instructs you that an element $A[x]$ is a *local minimum* if it is less than or equal to both its neighbors, or more formally, if $A[x-1] \geq A[x]$ and $A[x] \leq A[x+1]$. For example, there are five local minima (each boldfaced) in the following array:

9	7	7	2	**1**	3	7	5	**4**	7	**3**	**3**	4	8	**6**	9

Even a neophyte Algorithms wizard can find a local minimum in $O(n)$ time by scanning through the array. Dumbledore asks you to instead (a) prove that A must contain at least one local minimum, and (b) describe, analyze and prove the correctness for an algorithm that always finds *a single* such local minimum in $O(\log n)$ time.

Hint: Modify an algorithm you already know that takes $O(\log n)$ time.

4. (10 pts extra credit) Recall the pancake problem from the midterm, in which we have a stack of n pancakes of different sizes that we want to sort so that smaller pancakes are on top of larger pancakes. The only operation we can perform to change the ordering is a *flip*: insert a spatula under the top k pancakes, for some k between 1 and n, and flip them all over.


Flipping the top three pancakes.

(a) Describe an algorithm based on *Insertion Sort* that sorts an arbitrary stack of n pancakes, and prove that your algorithm is correct.

(b) In the worst case, exactly how many flips does your algorithm perform?

(c) Suppose one side of each pancake is burned. In the worst case, exactly how many flips do you need to sort the pancakes *and* have the burned side of every pancake on the bottom?