CS361 Homework #3
Due Tuesday, October 17th

1. Suppose I have a hash table with 50 locations. I would like to know how many items I can store in it before it becomes fairly likely that I have a collision, i.e., that two items get hashed to the same location. Assume that the hash function is random, and solve this problem in two ways:

   (a) Find the smallest value of n for which, if I store n items, the probability I *don't* get a collision falls below $1/2$. Do this by calculating this probability *exactly* for $n = 1$, $n = 2$, and so on.

   (b) Find the smallest value of n for which the *expected* (or average) number of colliding pairs is equal to or greater than 1. Do this by calculating this average exactly for the relevant values of n.

   Do these two methods give roughly the same answer? Can you explain why they are slightly different?

2. Consider the following question:

   > Input: a min-heap H containing a set of n numbers, an integer k, and a number x
   >
   > Question: does H contain k or more numbers which are smaller than x?

   Show how to answer this question in $O(k)$ time; in other words, in an amount of time that grows linearly with k, and doesn't depend on n.

   Hint 1: note that you are not being asked to find the kth smallest elements, just to find out whether or not they are all less than x. Hint 2: try "excavating" the heap, like an archeological dig. Feel free to look not just at the root of the tree, but inside the subtrees as well.

3. A *d-ary heap* is a heap based on a balanced d-ary tree, where each node has d children (except that the bottom row may be incomplete).

   (a) How would you implement a d-ary heap in an array?

   (b) How should `downSift` work in a d-ary heap?

   (c) Analyze the running time of `makeHeap`, `deleteMin`, and `insert` in terms of n and d. Find how the constant in Θ depends on d, and discuss whether they are faster or slower than in a binary heap.

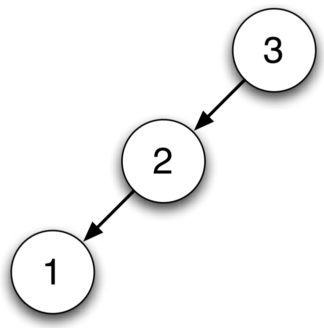

Figure 1: The binary search tree resulting from inserting three items $3, 2, 1$ in decreasing order.

4. Suppose I have a binary search tree. It starts out empty, and I insert three items into it. The shape of the resulting tree depends on which of the 6 possible orders I insert them in. For instance, if I insert them in decreasing order $3, 2, 1$, the tree will look like Figure 1.

   Assume that the 6 possible orders are equally likely; then, give the possible shapes the tree can have, and calculate the probability for each one.

   Now, for each of these shapes, suppose that I search for one of the three items, where each of the three is equally likely. Calculate the average number of steps I need to find this item, i.e., its average depth in the tree, where the root has depth 0: for instance, for the tree in Figure 1 we get $(1/3)(0 + 1 + 2) = 1$.

   Finally, take the average over the possible shapes, weighted by their probabilities, to get the average of this average: in other words, calculate the average depth of a random item in a random tree. If all goes well, you should get $1/3$ times the average number of comparisons Quicksort needs to sort 3 items, or $8/9$!

5. Let's prove that an AVL tree with n nodes has depth $O(\log n)$. We will do this by solving the opposite problem: finding the smallest number of nodes that can cause an AVL tree to have a certain depth. Call an AVL tree "extreme" if every node except the leaves is unbalanced to the right, and let $f(d)$ be the number of nodes in an extreme tree of depth d. Then $f(1) = 1$, $f(2) = 2$, $f(3) = 4$, and $f(4) = 7$; for instance, the extreme tree of depth 4 is shown in Figure 2.

   Find a recurrence for $f(d)$ and solve it within Θ. Then argue that $n \geq f(d)$ and therefore $d \leq f^{-1}(n)$ where f^{-1} is the inverse function of f. End by proving that $d = O(\log n)$. What is the base of the logarithm? How much deeper are AVL trees than perfectly balanced binary trees?

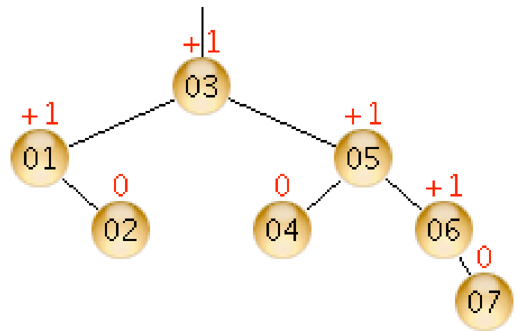

Figure 2: The extreme AVL tree of depth 4, with 7 nodes.

Extra Credit: We analyzed the version of Quicksort in which the pivot is a randomly chosen element. We found that the average number of comparisons is

$$f(n) = 2n \ln n$$

We did this in the following way. Let x be the position in the list that the pivot ends up in, and let $P(x)$ be the probability of a given value of x. Then the recurrence for the average number of comparisons is

$$\begin{aligned} f(n) &= \underbrace{n-1}_{\text{partitioning}} + \sum_{x=1}^{n} P(k) \underbrace{\left(f(x-1) + f(n-x)\right)}_{\text{recursion}} \\ &\approx n + 2\int_0^n P(x) f(x)\,\mathrm{d}x \end{aligned} \tag{1}$$

In the second line we assumed that $P(k)$ is symmetric, i.e., that $P(x) = P(n-x)$.

Now, if the pivot is random, its rank is equally likely to take any value from 0 to $n-1$, so every value of x occurs with equal probability $1/n$:

$$P(x) = \frac{1}{n} \text{ for all } 0 \le x < n$$

Then Equation (1) becomes

$$f(n) = n + \frac{2}{n}\int_0^n \mathrm{d}x\, f(x)$$

By substituting a solution of the form $f(n) = An \ln n$ into this equation and solving for A, we get $A = 2$ as before.

Now, an *improved* version of Quicksort uses the *median of three random elements* as the pivot. In this case, $P(x)$ is the probability distribution of the *median of three random numbers* in the unit interval $[0, 1]$. Calculate $P(x)$ by asking, given a random number x between 0 and n, what is the probability that if I choose two more random numbers y and z, then x is greater than y and less than z? Then, how many ways are there for this to happen? (You might want to check your work by confirming that the total probability $\int_0^n P(x)\,\mathrm{d}x$ is 1.) Intuitively, rather than being uniform, $P(x)$ should be peaked at $x = n/2$.

Finally, use this expression for $P(x)$ in Equation (1) and again try a solution of the form $f(n) = An \ln n$. You should be able to derive that A is now somewhat smaller than 2, indicating that this method of choosing the pivot improves the constant in the number of comparisons.

Even more extra credit: What happens if we choose the pivot by taking the median of 5, 7, ... elements?