HW4 CS305

Alex Petralia

March 2019

1 Question 1

a)
Let x be a node.
If $(x.has2Children() == true)$ then $successor(x).totalChildren() = ?$ We know if x has a right subtree (which it does) then the successor will
be the left-most node in that right subtree.
Thus the successor can have only a right child or no children at all. If the successor had a left child then it wouldn't be the successor to x. Thus the successor can have 1 right child or no children.

b)
(follows the same logic but with the roles of the right and left nodes are switched)
Let x be a node.
If $(x.has2Children() == true)$ then $predecessor(x).totalChildren() = ?$ We know if x has a left subtree (which it does) then the successor will
be the right-most node in the left subtree.
Thus the predecessor can have only a left child or no children at all. If the predecessor had a right child then it wouldn't be the predecessor to x. Thus the predecessor can have 1 left child or no children.

2 Question 2

a)

funA does not work since we can have numbers that don't satisfy the
properties of a BST by embedding the suspect numbers in the left and/or right subtrees of the root node.

funB is correct as the edge case for funA still works for funA.
Although funB works it seems inefficient since the function will traverse parts of the tree multiple times.

b)

For funA every element is compared to another element by simply just traversing
through each subtree. Since we are not checking the properties of the min and max value we are only doing one comparison for each node thus we have $\Omega(n)$ for all cases since the comparisons are strictly linear.

Looking at funB we noticed that it was innefficient since we have to traverse through parts of the tree multiple times.
Specifically, when we are comparing the min and max to the current node we are requiring comparisons of each node to eachother.
Thus, in the worst case, we have $\theta(n^2)$.

3 Question 3

a)

We know $n = 6$ and $m = 7$ for this case where n is the number of entries and m is the number of spots in the hash table.
Let X be the number of probes.
First we look at the definition for expectation with the give m value:

$$E[X] = \sum_{i=1}^{\infty} x_i p_i$$

Let p_i be the probability of a successful search and we will denote this as the function $p(i)$.
Since the success rate is $p(i)$ and we are computing an unsuccessful search, this means that our x_i will be $(i - 1)$.

Then we will simplify our definition with the given k value:

$$\begin{aligned} E[X] &= \sum_{i=1}^{\infty} x_i p_i \\ &= \sum_{i=1}^{\infty} (i-1) * p(i) \\ &= \sum_{i=1}^{\infty} i * p(i) + \sum_{i=1}^{\infty} p(i) \\ &= \sum_{i=1}^{\infty} i * p(i) + 0 \end{aligned} \tag{1}$$

Now let's look at the definition for a geometric series:

$$\sum_{i=0}^{\infty} ar^i = \frac{a}{1-r}$$

Let $\alpha = \frac{n}{m} = \frac{6}{7}$ and $a = 1$.
Suppose X is the number of probes made in the unsuccessful search. Now if we use (1) we get:

$$\begin{aligned} E[X] &= \sum_{i=1}^{\infty} i * p(i) \\ &= \sum_{i=1}^{\infty} a * \alpha^{i-1} \\ &= \sum_{i=1}^{m} \alpha^{i-1} \\ &= \sum_{i=0}^{m} \alpha^{i} \\ &= 1 + \alpha + \alpha^2 + ... + \alpha^i \\ &= 1 + (\frac{6}{7}) + (\frac{6}{7})^2 + ... + (\frac{6}{7})^i \\ &= \frac{1}{1 - \frac{6}{7}} \\ &= 7 \end{aligned} \tag{2}$$

We get to step (2) by knowing that $i*p(i) = i(p(x \geq i) - p(x \geq i+1)) = p(x \geq i)$. Then we find that:

$$\begin{aligned} p(x \geq i) &= \frac{n}{m} * \frac{n-1}{m-1} * ... * \frac{n-i+2}{m-i+2} \\ &\geq (\frac{n}{m})^{i-1} \\ &= \alpha^{i-1} \end{aligned}$$

Thus $p(x \geq i) = \frac{n}{m}^{i-1} = \alpha^{i-1}$ (where α is the load factor).
Note that $n = 6$ and $m = 7$, where $\frac{6}{7} = \frac{n}{m}$ is the load factor.

Therefore the expected number of probes is $\frac{1}{1-\frac{6}{7}} = 7$ and
$\frac{1}{1-\frac{n}{m}}$ in the general case.

b)

Let k be the key we are searching for and $\alpha = \frac{n}{m} = \frac{6}{7}$.
Now we know it requires at most $\frac{1}{1-\alpha}$ probes. For the upper bound we must require k to be the $(i+1)$st key inserted. Therefore the expected number of probes is at most $\frac{1}{1-\frac{i}{m}} = \frac{m}{m-i}$ which will be averaged over n. Thus:

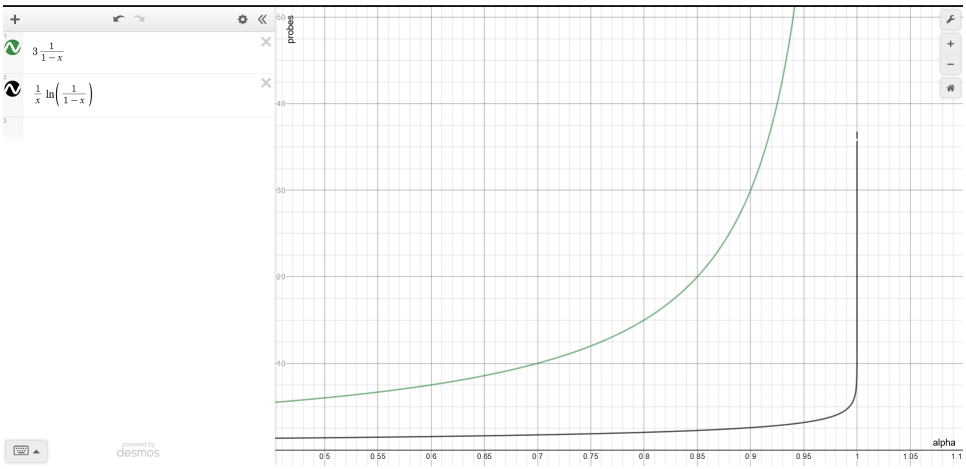
$$\begin{aligned} \frac{1}{n}\sum_{i=0}^{n-1}\frac{m}{m-i} &= \frac{m}{n}\sum_{i=0}^{n-1}\frac{1}{m-i} \\ &= \frac{1}{\alpha}\sum_{k=m-n+1}^{m}\frac{1}{k} \\ &\geq \frac{1}{\alpha}\int_{m-n}^{m}\frac{1}{x}dx \qquad (3) \\ &= \frac{1}{\alpha}ln(\frac{m}{m-n}) \\ &= \frac{1}{\alpha}ln(\frac{1}{1-\alpha}) \end{aligned}$$

(At (3) we used an integral inequality) Therefore $\frac{1}{\alpha}ln(\frac{1}{1-\alpha}) = \frac{1}{\frac{6}{7}}ln(\frac{1}{1-\frac{6}{7}}) = 2.27022850723 = 2$ (approx) probes. Consequently the upper bound is $\frac{1}{\alpha}ln(\frac{1}{1-\alpha})$.

4 Question 4

We know the expectation for a success is $\frac{1}{f}ln(\frac{1}{1-f})$
and for an unsuccessful search we have $\frac{1}{1-f}$.

Now if we graph these two as functions we can then see if they start to converge at any point:



Let $A(f) = \frac{1}{f}ln(\frac{1}{1-f})$ and $B(f) = 3\frac{1}{1-f}$.
As we can see, as f approaches 1 we notice that $A(f)$
is bounded above by $B(f)$.
When f gets near $f = 1$ the values of $A(f)$ get infinitely
smaller by comparison to the previous f value. (eg: $f = 0.998, 0.999$, etc.).
Both functions are undefined at $f = 0$ but $A(f)$ is bounded below $B(f)$ and $A(f)$ becomes undefined much faster than $B(f)$. The only way for these functions to converge and find a f value that satisfies the conditions given would be for f to be 1 which is undefined since $f < 1$ for the functions to be defined and the load factor can't be 1.