CSC263: Problem Set 4

October 15, 2019

1 Problem 1

(a) The worst case running time will be in $O(m)$, where m is the length of the hashtable. For example, consider the scenario where an object is to be inserted into the 0 position into a hashtable T that runs from 0 to $m-1$, and the first $m-2$ spots are occupied. Then if an object was supposed to be inserted into the 0 position, which is occupied, it would move over and check again for an empty spot to be inserted into. This would keep repeating till an empty spot would be found, which in this case would be the $m-1$ position. Thus, $Insert$ in the worst case would run in linear time, and in this particular case m slots would have to be visited.

More specifically we can consider the load factor α. From theorem 11.6, the expected number of probes made during an unsuccessful search is at most $\frac{1}{1-\alpha}$, where $\alpha = \frac{n}{m}$, where n is the number of keys and m is the size of the table. So if $n = m-1$ then for large n and m, the expression becomes $\frac{1}{1-\alpha} = \frac{1}{1-\frac{m-1}{m}} = \frac{1}{\frac{1}{m}} = m$. So m slots are visited.

(b) Let the sequence of $n+1$ keys be multiples of 7, that is $7, 14, 21, ..., 7(n+1)$ for $n+1$ such keys. Let the hash function be given by $h'(k,i) = (h(k)+i)$ $mod\ 7$; $h(k) = k\ mod\ 7$, where k is a key from the aforementioned sequence, and $i = 0, 1, 2, 3, ..., (m-1)$, where $m-1$ is the last position in the hashtable of size m. Then selecting linear probing, we can ensure that for the $(n+1)$-th $Insert$ achieves the worst case running time of $O(m)$. This is because every key k from the sequence will have $k\ mod\ 7 = 0$ since each k is a multiple of 7. And so with linear probing, the $(n+1)$-th key will first check the 0th position, but since that is occupied, the 1st position would then be checked, and that too will be occupied, and so on, till finally the $(n+1)$-th position is checked and this is empty so it is finally inserted here.

(c) Since we have $n+1$ keys, there will be $n+1$ calls made to $INSERT$. Note that the 1st $Insert$ will take 1 step, the 2nd $Insert$ will take 2 steps, and so on, till the $(n+1)$-th $Insert$ which will take $n+1$ steps. Summing all these steps we get $1 + 2 + ... + (n+1) = \frac{(n+1)(n+2)}{2}$. So on average

it will take $\frac{\frac{(n+1)(n+2)}{2}}{n+1} = \frac{n+2}{2}$ steps to insert a key. Written in big Theta notation, it would be $\Theta(n)$.

(d) The function $h'(k,i) = (h(k) + i)\ mod\ m$; $h(k) = \frac{k}{7}$, where m is the size of the hashtable. This will ensure that $Insert$ runs in constant time, $O(1)$ because $h(k)$ gives us the set of natural numbers $1, 2, ..., (n+1)$ with the sequence of $n+1$ keys specified in part (b). And $h'(k,i)$ ensures that these keys are inserted in order into the hashtable. As an example, $h'(7,0) = 1$, $h'(14,0) = 2$, $h'(21,0) = 3$, ..., $h'(7(n+1),0) = n+1$. So the keys are inserted into the table sequentially, and in constant time, as the first spot provided by $h'(k,i)$ for a given key, k, will always be empty, assuming the hashtable is large enough to accommodate all $n+1$ keys.