Problem 1 Theoretical Analysis

Written By: Chris(Junyu) Liang, Student ID: 1159696

Question 1

If there are `n` elements in the base level, and as `p` = $\frac{1}{2}$, there must be $\frac{1}{2}n$ elements on the second level. Similarly, there will be $\frac{1}{2} * \frac{1}{2} = \frac{1}{4}n$ elements on the third level. Thus, for a list of height 2, for example, it will have $n + \frac{1}{2}n$ elements in total. Therefore, if there are `i` levels in the list, there should be $\sum_{k=1}^{i} \frac{n}{2^{k-1}}$ elements in total. After solving, we get $2^{1-i}(2^i - 1)n$ elements in total. Since 2^{1-i} and $2^i - 1$ are both constants as `i` is constant for a leap list that has a height of `i` , we can conclude that the average space efficiency of a leap list is in $O(n)$.

Question 2

The height of a leap list should be $\log_2 n$, with `n` denotes the number of elements in the base list, and with `p` = $\frac{1}{2}$. Since `p` = $\frac{1}{2}$, the second level should have $\frac{1}{2}n$ elements, and the third level should have $\frac{1}{4}n$ elements, and so on. That is, an upper level should contain half of the number of elements of the level below it. Assuming the number of elements inserted to a leap list is divisible by 2, as there more levels being built, the number of elements will be halfed at each level, with the top level only containing 1 element, which cannot be divided. In conclusion, we can conclude that for each level above the base level, number of elements in that level will be halfed, which resulting in a total height of $\log_2 n$.

Question 3

Let `h` denote the number of levels in total (i.e. height) of the leap list. On average, I assume the searching key is in the leap list, but not at the top level, and values in each level are normally distributed. Hence, when performing a search, it will be like using a binary search, with the first or second list finds out approximately mid-point, and the level below finds the mid-point of the mid-point, since we immediately perform a drop down when the `next` value is grater than key in a level. Therefore, on average, leap lists should have a similar time efficiency as using binary search, which is in $O(log(n))$.

Question 4

In the worst case, the key we are finding is larger than all elements in the leap list, which is not in the list. In this case, we need to search $O(n)$ elements on every level of `h`. Thus, the total complexity will be $h * O(n) = O(hn)$. Since in question 2 I have concluded that the total height will be $log_2(n)$, in conclusion, the worst case complexity will be $O(hn) = O(nlog_2(n))$, where `h` $= log_2(n)$.