1. The probability of inserting the element to the higher level is always $\frac{1}{2}$. The base level 1 has n element, so the second level will have $\frac{1}{2}n$ elements. Thus, at level i, there are $(\frac{1}{2})^{i-1} \cdot n$. To sum up the number of elements at each level, $\sum_{x=1}^{i} (\frac{1}{2})^{x-1} n = \frac{n(1-\frac{1}{2}^{i})}{1-\frac{1}{2}}$

$$= \frac{(1-\frac{1}{2}^{i})n}{\frac{1}{2}} \in O(n)$$

2. Let the total number of lists in a leap list be x. Assume that there is at least one elemet at the highest level. Thus, $(\frac{1}{2})^{x-1} \cdot n = 1$

$$\Rightarrow (\frac{1}{2})^{x-1} = \frac{1}{n}$$

$$x-1 = \log_{\frac{1}{2}} \frac{1}{n}$$

$$x = \log_{\frac{1}{2}} \frac{1}{n} + 1$$

$$x = \log_2 n + 1$$

3. When searching for a key, half number of elements of each level have to be compared. The search process is similar to Binary search tree, so the time complexity will be $O(\log_2 n+1) \in O(\log_2 n)$

4. At worst case, the key is bigger than any elements in the leap list, so every element and height will be compared, thus the time complexity is $O(n+h)$