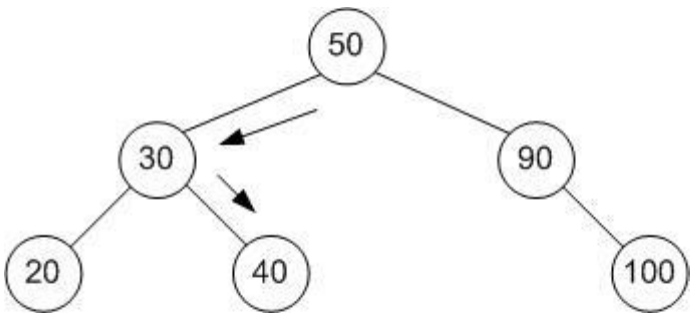
Problem 3.1

Question 1 (3.1.1)

The reason that there is no better algorithm in this case is intuitive. We assume that array is not sorted. If the key is equal to the first element of the list, only one comparison is needed. If the element does not exist in the array, then n comparison is required. Because the data structure we use is an array and the array is not sorted, we have to sequentially search for the element in array. And, because our movement within the array has to be sequential, there is no better search algorithm.

Question 2 (3.1.2)

We assume that the array is already sorted. We can think of an array a binary tree as follows:



Each node represents a part of an array containing only a single integer. The worst case happens when the deepest level of the tree (the single-item array at the bottom) is reached and its complexity is $C(n) = floor(\lg n + 1)$. If we checked each item by comparing it with the number that we search starting from the beginning and iterating until the end, it would be $C(n) = n$ worst time complexity and this is already worse than $\lg n + 1$. Every comparison divides by two the number of possible locations where the key can be. Therefore, $\lg n + 1$ is the best complexity that we can achieve if the data structure which we use is an array. If we used, for example, a hash table, then the complexity would be $O(1)$.

Question 3 (3.1.3)

If the only prior knowledge you have is that it contains n characters, then we have to systematically enumerate all possible candidates for the solution and then check whether each candidate is equal to the password that we are trying to crack.

We can safely assume that the characters in the password will be from ASCII number 33 through 127. That is 127-33=94 characters.

$$\underbrace{\boxed{94}\quad\boxed{94}\quad\boxed{94}\quad\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\quad\boxed{94}}_{n\ characters}$$

$$= 94^n$$

Therefor the worst time complexity is $O(94^n)$.