CS50

Linear Search

Overview

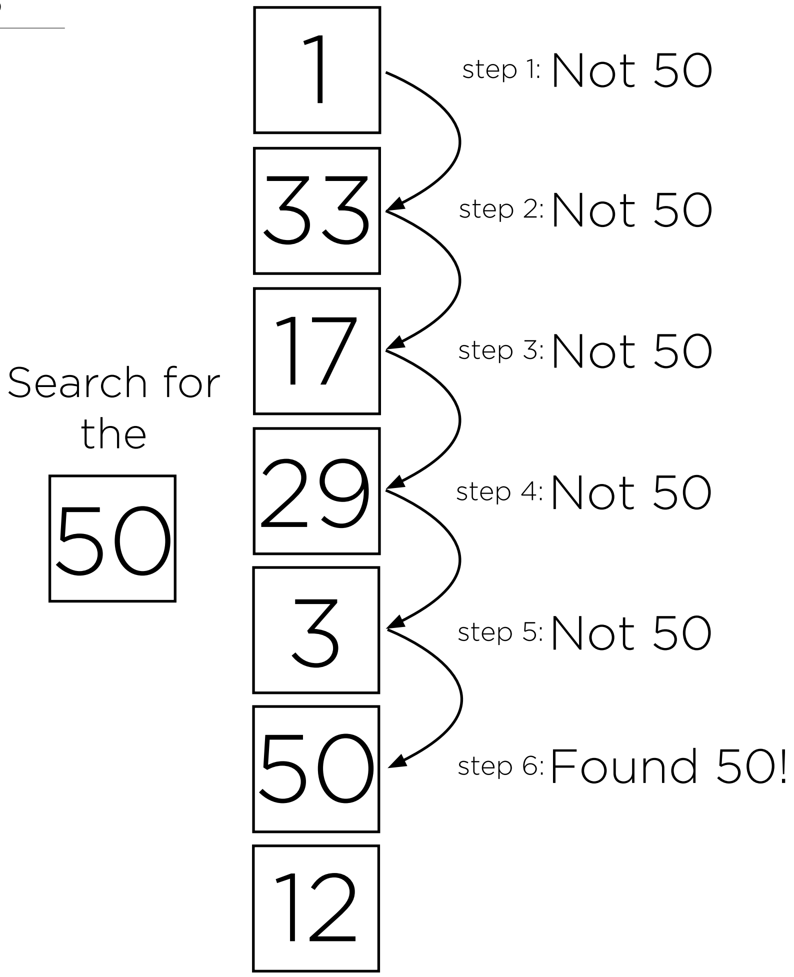
There are many different **algorithms** that can be used to search through a given list. One such algorithm is called **linear search**. This algorithm works by simply checking every element in the list in order. It will start at the beginning of the list and increment through the list until the desired element is found. In this way, linear search would require checking every **element** before reaching the conclusion that the element does not exist in the list.

Key Terms

- algorithm
- linear search
- element
- computational complexity
- constant

Efficiencies and Inefficiencies

While the linear search algorithm is correct, it is almost never the most efficient. The worst, or least efficient, case would be when the desired element is the last element in the list or not in the list at all (both have the same efficiency). We would have to look through every element in the list to find the one we're looking for. This would take *n* steps, where *n* is the length of the list. The **computational complexity** of linear search would be O(*n*). That may seem pretty daunting, especially if we have a list that has millions of elements. However, the best case scenario is much better; if the desired element is the first in the list, we would find it in one step. Although linear search is not usually the most efficient, linear search can be useful in certain situations. Take for instance a list that you know nothing about, like a stack of papers that are out of order. It is just as efficient to search this stack linearly as it would be to search for elements randomly. In this case, we cannot search for elements in a more efficient way since we don't know anything about the list. There is no information about the list's organization for us to leverage. Now you can see why it would seem rather convenient to sort a list before searching it. Granted, sorting also takes up time and space, but this additional step will save you some time if you plan on searching a list multiple times or if you have a very large list.



Examples of Linear Search

Recall the phone book example. While looking for Mike Smith, linear search meant going through each page of the phone book one at a time. Even the algorithm where we flipped through two pages at a time can be considered linear. In fact, any algorithm in which the there is a **constant** being multiplied by the total number of elements in the list to determine the search time, is considered linear. For example, if you are turning three pages of the phone book at a time (remembering to make the correction if you overshoot), the search time would be *n/3*. Since 1/3 is being multiplied to *n*, this algorithm would be considered linear. Linear search should typically be avoided, as there is usually a better algorithm that can implemented to make the search more efficient. Sorting a list first, is one such way to search more quickly. Once a list is sorted we are able to leverage some of the concepts learned earlier to make a faster, efficient, and more elegant program.