Searching Algorithms

By Lujia Zhang(Luke), Daniel Hartung, and Daniel Schnipke

Sources

Big Java, Fifth Edition (Horstmann); Sections 14.6 and 14.7

Summary

Definitions:

There are two main types of searches that are used; sequential and binary.

A sequential search operates by iterating through the list, and on each element checking to see if that element is the element that we are looking for. This is also called a linear search, because it progresses through the list in a linear manner, as well as for its runtime.

A binary search locates a value in a sorted array by determining whether the value occurs in the first or second half, then recursively repeating the search in one of the halves.

Uses:

Searching is used to see if a piece of data is in a list of specific pieces of data. Examples include searching a database for a person based on their fingerprint data, searching an inventory to see a company has a certain part in stock, or google.

The sequential search has the best case running time as O(1) when the number that we are looking is the first number in the array. The average runtime for a sequential search is half of the number of elements multiplied by the time it takes to do a single comparison, which is O(n). The worst case runtime for a sequential search would be the length of the entire list multiplied by the time it takes to do a single comparison, which is also O(n). It is also worth noting that whenever an item is not in the list, the sequential search will go through the entire list looking for it. This means that whenever an item is not contained in a list that it is being searched for it, it will take the worst case runtime.

Binary search also has the best case running time as O(1) when the number that we are looking for locates in the exact middle of the array. However, this only rarely happens. For the average runtime, the list is split in half repeatedly. Because of this nature, the search matches with the function of a logarithm with the base of 2, resulting in an average runtime of O(log(n)). Because the search always follows this pattern, it is unable to take longer, the binary search algorithm also has a worst case runtime of O(log(n)).

If you search the array only once, then it is more efficient to pay for an O(n) linear search than for an O(n log(n)) sort and an O(log(n)) binary search. But if you will be making many searches in the same array, then sorting it is definitely worthwhile.

Samples: See example code provided in the public repository.