Study Write-Up: Binary Search

Binary search is a very useful and efficient algorithm for searching through data. It halves the amount of computation needed to search through a set of data, just to demonstrate its efficiency. Fortunately, such an efficient algorithm is quite simple and easy to understand.

Most likely, one's first idea to search through a set of data is simply to look at each and every item in the set until the desired item is found, starting from the beginning. However, this is pretty obviously cumbersome and slow, especially if the desired item is near the end of the data set.

A more effective way to search is by taking a divide-and-conquer strategy. The idea is to search data how you might automatically perfectly search a phone book. A phone book is ordered from A to Z. So, you start by searching the halfway point in the phone book. Let's say it starts with an M, since M is the 13th letter in the alphabet. If the name we are looking for starts with anything from A to M, for example, "Fred," we find the halfway point between A names and M names. If, on the other hand, we are searching for a name farther down the alphabet than those that start with M, for example, "Sam," we find the halfway point between M names and Z names. Each time, we find the middle element of those elements left to be searched and repeat the process. We terminate the process once the sought-after value is found or when the remaining list of elements to search consists of only one value. By following this process, each time we compare, we reduce half of the search space. This halving process results in a tremendous savings in the number of comparisons made.

Here is an example of binary search in Ruby:

```ruby
def binary_search(array, value, from=0, to=nil)
    if to == nil
        to = array.count - 1
    end

    mid = (from + to) / 2

    if value < array[mid]
        puts "found at index #{binary_search(array, value, from, mid - 1)}"
    elsif value > array[mid]
        puts "found at index #{binary_search(array, value, mid + 1, to)}"
    else
        puts "found at index #{mid}"
    end
end

binary_search([100, 200, 300], 200)
```

All in all, binary search is a great algorithm that makes searching much more efficient. It is a prime example of making a computer work smarter rather than harder. As such, it is a good algorithm to use when searching through large sets of data for both you and your computer.