Search and Sort Algorithms

Author: Rohith Perumalla

Date: 10/21/16

Subject: Computer Engineering

Citations:

Tutorialspoint.com. "Data Structure and Algorithms (DSA) Tutorial." www.tutorialspoint.com. Tutorials Point, n.d. Web. 13 Oct. 2016.

Summary:

Algorithms are processes that perform calculations or other problem-solving operations, especially by a computer. Algorithms are often characterized by their input, output, finiteness, feasibility, and independence. Search algorithms retrieve information stored within data structures; the appropriate search algorithm is often dependent on the data structure being searched. A few search algorithms include Linear Search, Binary Search, Interpolation Search, and Hash Tables. While search algorithms search for values within data structures other algorithms, sorting algorithms, sort the values in a data structure or list. Data Structures are sorted in order to optimize the use of other algorithms, like search algorithms. Some sorting algorithms include Bubble Sort, Insertion Sort, Selection Sort, Merge Sort, Shell sort, and Quicksort. Even though there are many types of algorithms most algorithms are created to be independent of the underlying language, meaning they work regardless of the language used.

Analysis:

An algorithm is used to effectively perform some sort of operation. There many types of algorithms some including searching algorithms, sorting algorithms, inserting algorithms, updating algorithms, and deleting algorithms. Two types of algorithms that are very commonly used are searching and sorting algorithms. Searching algorithms and sorting algorithms are very useful when handling data structures and looking for or accessing values. Some searching algorithms are Linear Search and Binary Search; some sorting algorithms are Bubble Sort and Insertion Sort. These algorithms can be used on any platform and language, unlike some computer science concepts.

Linear Search is a very simple basic search algorithm. Linear Search is sequential search, that begins at the beginning of the collection of data and goes through every value looking for a match and then returns if it has found a match or if it hasn't. Linear Searches can either work from the back end or front end of the list depending on how it is configured. Linear Searches may be more effective in certain situations where values in a collection are not sorted and cannot be sorted (i.e. Stacks or Queues); however, due to Linear Search being a sequential search and its necessity to look through every value to find a match Linear Search is not considered to be the most efficient and has a Big-O Notation of O(n).

Binary Search is a very efficient searching algorithm. Binary searches work with sorted collections, the collection has to be sorted otherwise it will not work. Binary search begins by looking at the middle value in the collection and evaluating if the value is greater than less than or equal to the value it is looking for; if the middle value is a match it returns the middle value. Otherwise, the algorithm repeats the process on the first half (if the middle value is greater than the value the algorithm is looking for) or the second half (if the middle value is less than the value the algorithm is looking for) of the array by dividing it in two and evaluating the middle value and it and repeats the process until it finds the value or it reaches the last individual value and returns that the value the algorithm was looking for is not in the collection. Binary Searches may be more effective in certain situations where there are mass amounts of sorted numbers in a collection; however, due to binary search needing a sorted collection if a collection cannot be sorted binary search will not work. Binary Search has a Big-O Notation of O(log n), but, if the collection is not already sorted then the time complexity would increase depending on the time complexity of the sorting algorithm's.

Bubble Sort is a simple sorting algorithm that sorts the values in a collection from least to greatest or from greatest to least depending on how it is structured. This sorting algorithm is a comparison based algorithm, where the algorithm looks at pairs of adjacent values and then swaps them if they are not in order until they are in order. Let's look at a collection with 4 integers: {a, b, c, d}. Bubble sort works by starting off with the first two elements (a & b) and then compares them, if the first value is greater than the second value they switch (if a>b then the collection is now {b, a, c, d}), and the algorithm continues evaluating the current value (a)

with the following values until it finds one greater than ‘a’. However, if ‘a’ was not greater than ‘b’ then they would not switch and then they algorithm would focus on ‘b’ and compare it to its adjacent values. Eventually after checking all the adjacent values of every element the collection will be sorted. Bubble Sort would be used in very specific situations, one being a situation where there is not enough storage space to hold a temporary variable that a few more efficient sorting algorithms need. However, this sorting algorithm is not the most efficient and is not recommended for large data sets since it has to compare every value with each other resulting in a Big-O notation of O(n^2).

Another sorting algorithm is Insertion Sort. insertion Sort is another comparison based sorting algorithm that starts at the base of the collection and works through it creating a maintained sub-collection at the base, as it works through the collection it compares the current value with the following values adding any out of order to the sub-collection sorting it as it works through the rest of the array. In the same example of the collection {a, b, c, d} if a < b the algorithm considers {a} the sub array and the focuses on ‘b’ and compares ‘b’ to ‘c’ and if ‘c’ < ‘b’, ‘c’ and ‘b’ switch and ‘c’ is then compared to the rest of the sub array and if ‘c’ < ‘a’ then the sub-collection is found as {c, a} and will continue to add all the other values until the algorithm goes through the entire collection and ends up with a sorted sub-collection containing all the values. Insertion Sort would be used in very specific situations, a situation where there is not enough storage space to hold temporary variable that a few more efficient sorting algorithms need. However, this sorting algorithm is not very efficient and is not recommended for large data sets since it compares every value with each other resulting in a Big-O time complexity of O(n^2).

Algorithms are methods of completing a process some are better than others but each have their own uses and benefits. algorithms can search, sort, insert, update, and delete. Searching and sorting are extremely useful and make performing tasks much simpler and faster. Overall algorithms are extremely useful and are essential in developing software and creating useful efficient products.