Sorting Algorithms in C+

Maxwell Miller
Fowler School of Engineering
Chapman University
Orange, California 92866
Email: MaxMiller@Chapman.edu

***Abstract*—There are five main types of sorting algorithms that are commonly used in C++. In this paper we will find that each one has its own benefits and draw backs.**

1. Introduction

Sorting algorithms are one of the most important parts of modern computing. Choosing the correct algorithm can drastically change a programs run time, which when working with large data sets can save hours, even days, of computing. There are five algorithms computer scientists use the most; Bubble Sort, Insertion Sort, Selection Sort, Quick Sort and Merge Sort.

December 11, 2019

1.1. Bubble Sort

Bubble Sort is the most basic of the sorting algorithms. It compares every neighboring element, from left to right, swapping them if the left element is bigger then the right element. Check through the data as many times as there are elements too insure it is fully sorted. Bubble Sort is the perfect algorithm to choose when you need to program a sort fast and don't care about efficiency or speed. It can be written in as little as seven lines of code. This algorithm is not suggested for large data sets.

1.2. Insertion Sort

Insertion Sort reads through the data once checking if each element is properly sorted. If it finds an element that is smaller then the previous element it is inserted into the correct place in the data. This means that if the elements are in exactly reverse order it will take the maximum runtime. But this also means that Insertion Sort is the fastest sort if the data is mostly sorted already, or you are trying to put a new element in an already sorted data set. This algorithm is not suggested for large data sets.

1.3. Selection Sort

Selection Sort finds the smallest number in the data set, removes it, and inserts it into a new data set until the original is empty. Since this algorithm only needs to move each element once, this sort is useful when you are bottle-necked by system memory. Though, modern machines are rarely ever bottle-necked by their memory. This algorithm is not suggested for large data sets.

1.4. Quick Sort

Quick Sort starts by finding a pivot and splitting the data around it, making it a Divide and Conquer Algorithm. Once it finds a pivot it program places anything that is larger then it to the right, and anything smaller to the left. It recursively runs on each side of the data until it is fully sorted. This algorithm is useful for random and larger data sets since it is fast and works best when the numbers start in a random order. Quick Sort is suggested for sorting arrays since it doesn't need to repetitively create and delete new arrays to hold data.

1.5. Merge Sort

Merge Sort starts by splitting the data in half, recursively calling itself on each half, then merges each half back together sorting the data. This makes it a Divide and Conquer Algorithm like Quick Sort. It is useful to use for larger data sets since it is fast. Merge Sort is suggested for sorting linked lists since it can insert data at any point without needing to create and rebuild a new array.

2. Conclusion

In today age of computing we have multiple ways of sorting a set of data. Each one goes about the problem of sorting data differently and have their own strengths and weaknesses. If you just need to quickly sort a list of numbers, its probably safe to use Bubble Sort. If your trying to keep a data set organized when inserting new elements, Insertion Sort is the way to go. Though when you are trying to sort a very large data set and need it sorted quickly you should use Quick Sort if you are using an array and Merge Sort if you are using linked lists.