Searching and Sorting

Alejandra Hernandez

College of DuPage

# 

# 

# Searching and Sorting

Searching and Sorting algorithms have been in existence for a long while. The tasks of these algorithms involve gathering all th**e** data such as, being, int, char, bool, floating point values into a list or an array. Then rearranging them so that they become sorted in any order that the programmer pleases. When it comes down to searching, the programmer then creates a data structure to find a specific element within the list or array.

**Real World Example**

We search and sort all the time. One of the ways users commonly use a search algorithm is when opening a file finder, and searching for an application, picture, or whatever document you have on your drive. When looking for a specific file in your drive, there is an algorithm going through your entire drive, looking at each item, comparing elements to others to find what it is the user is searching for.

Sorting is a critical part of searching algorithms. Searching a sorted list can decrease the time it takes for the program to find what the user is looking for. It increases the efficiency and effectiveness of the searching algorithm because it doesn’t have to work as hard. On the other hand, a programmer may want to just sort data for visualisation purposes such as data analysis.

**Bubble Sort**

A bubble sort is a very simple comparison sorting algorithm. Bubble sort begins with taking the first two elements and comparing their value. The element with the lesser value gets placed before the current element. The algorithms then increments by one element comparing then next to elements and so on. The algorithm will repeat until every element is sorted.

**Insertion Sort**

The insertion sort algorithm is another comparison based algorithm. Similar to bubble sort it starts off by comparing the first two elements. The key difference is that insertion sort builds a sorted list as it compares while bubble sort has to iterate many times. Insertion will compare the next element to every element before it to find its place.

**Shell Sort**

A shell sort algorithm is an algorithm much like insertion sort. The key difference is that shell sort, instead of comparing the values next to each other, it starts off by comparing an element somewhere in the middle of the list. As the algorithm iterates, the gap becomes smaller and smaller until the list is completely sorted. This is much more efficient than insertion sort in the case that maybe the smallest value would be towards the end of the list causing the algorithms to waste time.

**Linear Search**

A linear search goes through each element in the list until the algorithm finds said element.

**Binary Search**

A binary search algorithm works only on a sorted list. It starts in the middle of the list and compares the value to the element in the middle. It then narrows the interval to whichever half and continues until the value is found.

**Conclusion**

Out of the sorting algorithms explained above, Shell sort is the most efficient sorting algorithm. Shell sort can relocate any item into any position by swapping places with it. Making it faster than insertion sort. Bubble sort would be the most inefficient algorithm for the amount of iterations it must go through until the list is fully sorted.

# 

# References

Brilliant. (2020, October 11). Sorting Algorithms. https://brilliant.org/wiki/sorting-algorithms/.

GeeksforGeeks. *Sorting Algorithms*. https://www.geeksforgeeks.org/sorting-algorithms/.

*Searching and Sorting*. Javanotes 8.1, Section 7.4 -- Searching and Sorting. http://math.hws.edu/javanotes/c7/s4.html.

Settergren, R. (2017, October 12). *In computer programming, why is sorting important? When are the sorting algorithms used in real coding?* Quora. https://www.quora.com/In-computer-programming-why-is-sorting-important-When-are-the-sorting-algorithms-used-in-real-coding.

tutorialspoint. *Data Structure and Algorithms - Shell Sort*. Tutorialspoint. https://www.tutorialspoint.com/data\_structures\_algorithms/shell\_sort\_algorithm.htm.