Bubble Sort and Binary Search

The sort I decided to write about is bubble sort because when I ran it against the selection sort, the time bubble took was more than that of selection but is more simple. Bubble sort is pretty simplistic in its operation, what it’ll do is start at index zero and run through the entire array, switching elements only when the current element is greater than that of its right neighbor. This will find the largest element in the array and put it in the final position of the array. This process takes ‘n’ number of iterations through a for loop where n is the length of the array. From here, it starts at index 0 again and repeats the process but will only do the for loop, where it compares right neighbor values, n-1 times since the last element is in the right spot. It’ll do that until it gets to the beginning of the array and by that time the array is sorted and the process is then done.

Binary search is done by taking an ordered array, this won’t work unless it’s ordered, and will find the middle of the array. For example, an array with length 50 would have a middle index of 25. It then checks to see of that middle value is the desired value and if it isn’t it’ll compare if the wanted value is larger or smaller than the value at the middle index. If the desired value is less than the middle value, it’ll find a new middle, in the case of our example the new middle would be 13, and repeat this checking process. If it’s larger than the middle value, the new middle would be found, our case the new middle would be 38. It repeats this process in whatever direction it determines to lead it to the desired value or exhaust all options.