Lab 01 - COSC 3020

Wednesday, August 31, 2022 3:15 PM

Beckham Carver

Our insertion sort code is the following:

```
function insertionSort(arr) {
    for (var i = 1; i < arr.length; i++) {
        var val = arr[i];
        var j;
        for (j = i; j > 0 && arr[j - 1] > val; j--) {
            arr[j] = arr[j - 1];
        }
        arr[j] = val;
    }
}
```

Best case:
From the outer loop we can conclude that our minimum run time (best case) is O(n) because we must interact with each element of the array at least once, to check if is larger than the previous element. This makes our minimum total iterations belong to O(n) if the array is already sorted. This was also clarified in class

Worst case:
If our inner loop does not immediately return false, then it will result in some fraction of 'n' steps being taken. This is because it searches through the sorted portion of our array, which is a minimum of 1, and a maximum of (n-1). Our worst case for regular insertion sort is a descending list of non-duplicate numbers. This means that each inner loop will run for (n-j) steps. Because j is function of n we conclude that our worst case scenario belongs to $O(n^2)$.

Average case:
Furthermore our average case also belongs to $O(n^2)$. This is because when selecting a number from a defined set of numbers, it will on average be greater than half of the set of numbers. When inserting an item to our sorted array which is of size (n-j), on average it will need to search through half of the sorted array.

This means that every iteration of the inner loop will on average run for **(n-j) * 0.5** steps. In terms of asymptotic comparison, c1 * n - c2 is equivalent to n, which proves our average case also belong to $O(n^2)$.