ADS week 6

P1/ a. at first we have an array of size n

A[n]

```
for i from 1 -> n
    for j from 0 -> n-1
        if A[j] > A[j+1]
            swap(A[j], A[j+1])
```

* the first loop shows how many times we are executing overall, since we are sorting one number at the time (putting the highest value at the end of the array this will take n time to do.

* the second loop iterates through each 2 elements, with the condition in the 3rd line that compares each 2 elements with each other. and the 4th line swaps them

b. the time complexity is obviously $O(n^2)$ as we can see from the nested for loop. this occurs during the worst and the averaage case scenarios

When a best case happens (array already sorted) time complexity is $O(n)$ because there's no action of swapping in the second loop.

c. the buble sort is stable because there's no swapping between equal elements so their position to each others remains the same.

same happens with insertion sort because it only swaps element when neccesary while shifting element to their correct position, if 2 elements are equal the one that appears first remains first

merg sort is alsce stable, it shift equal element that are in the left in their place before doing that with the dues in the right.

heap sort is the only unstable sorting algorithm because it uses a heap structure where the order of the equal values are not necessary maintained. when extracted.

d – insersion sort and Bubble sort are adaptative, because their Time complexity is faster in the bestcase $O(n)$ instead of $O(n^2)$ wast case

this is not the case for heapsort and merge sort that have a constant time complexity of $O(n \log n)$.