Selection Sort

The strategy taken in a selection sort is to find the smallest item and put it in the first position (by swapping the current item 0 with the newly found smallest item). Then to find the second small item and put it in the second position, etc. In other words, it fills in each slot in the list, each time doing a sequential search through the unsorted remainder of the list.

Example

```java
public void SelectionSort (int[] A) {
    for (int i=0; i< A.length; i++) {
        int j = findIndexOfSmallestInt(A, i, A.length-1);
        swapIntegers(A, i, j);
    }
}

private int findIndexOfSmallestInt(int[] A, int low, int high) {
    int winner = low;
```
for (int i=low+1; i<=high; i++) {
        winner = i;
    }
}
return winner;

private void swapIntegers(int[] A, int i, int j)
{
    int temp = A[i];
    A[i] = A[j];
    A[j] = temp;
}

- The time complexity of selectionSort depends on the complexity of the other two, so let’s start with them.
- The time complexity of swapIntegers is $O(1)$ because it does not depend on the size of the list.
- The time complexity of findIndexOfSmallestInt is $O(N)$. It is called on segments of the array ranging from length $N$ down to 1, so that averages out to arrays of length $N/2$.
- The time complexity of selection Sort is $O(N^2)$. There are $N$ calls to findIndexOfSmallestInt and they take time $N/2$ on average and so the time is $N \times N/2$, but were move constants from Big-Oh notation, so we say it is $O(N^2)$

Complexity: $O(n^2)$
**Insertion Sort**

The insertion sort, like the selection sort, builds the list one at a time. But instead of searching for the lowest, then second to lowest items. It simply puts the items in order as the items are encountered.

**Example**

![Insertion Sort Diagram](image)

Video: [https://www.youtube.com/watch?v=ROalU379l3U](https://www.youtube.com/watch?v=ROalU379l3U)

```java
public void InsertionSort (int[] A)
{
    for (int i=1; i< A.length; i++)
        insert(i, A);
}

private void insert(int i, int[] A)
{
    int temp = A[i];
    while( i > 0 && temp < A[i-1] ) {
        A[i] = A[i-1];
        i--;
    }
    A[i] = temp;
}
```

Insert is O(N). It is called on segments of the array ranging from length 1 up to N, so that averages out to arrays of length N/2. Insertion Sort is O(N²) because it uses a for loop with N iterations, that calls a method of O(N).

Complexity: O(n²)
**Bubble Sort**
It works by sequentially going through your array and comparing two values at a time, swapping them if necessary. It then repeats the process until no swaps are required.

Video: [https://www.youtube.com/watch?v=lyZQPjUT5B4](https://www.youtube.com/watch?v=lyZQPjUT5B4)

**Example**

```java
public void bubbleSort(int[] A){
    for (int i=A.length -1; i>0; i--)
        for (int j=0;j<i;j++)
            if(A[j]>A[j+1])
                swapIntegers(A,j,j+1)
}
```

Complexity: O(n²)

Also check out: [https://visualgo.net/sorting](https://visualgo.net/sorting)