Given an array \( a \), rearrange the items so that they are in non-decreasing order. That is, \( a[i + 1] \geq a[i], \forall i : 0 \leq i \leq size - 2 \). The sorted array must have the same elements that were in the original array.

**Bubble Sort**

```java
for (int i = n-1; i > 0; i--)
    for (int j = 0; j < i; j++)
        if (a[j] > a[j+1]) a.swap(j, j+1);
```

A sorting algorithm is **stable** if all elements with the same value have the same relative ordering as before the sorting.

- Is Bubble Sort stable?
- What kind of proof would you use to prove Bubble Sort works?
- What is the runtime of Bubble Sort?

**Insertion Sort**

```java
for (int i = 1; i < n; i++)
    int j = i;
    while (j > 0 && a[j] < a[j-1])
        a.swap(j, j-1);
    j--;
```

**Selection Sort**

```java
for (int i = 0; i < n-1; i++)
    int smallest = i;
    for (int j = i+1; j < n; j++)
        if (a[j] < a[smallest]) smallest = j;
a.swap(i, smallest);
```
IN EFFECTIVE SORTS

**XKCD # 1185: Ineffective Sorts.** StackSort connects to StackOverflow, searches for 'sort a list', and downloads and runs code snippets until the list is sorted.

```c
void MergeSort(T a[], int l, int r) {
    if (l<r) {
        int m = floor((l+r)/2);
        MergeSort(a,l,m);
        MergeSort(a,m+1,r);
        Merge(a,l,r,m);
    }
}
void Merge(T a[], int l, int r, int m) {
    int i=l, j=m+1, k=0;
    while (i <= m || j <= r) {
        if (i <= m && (j > r || a[i] < a[j])) {
            temp[k] = a[i];
            i++; k++;
        } else {
            temp[k] = a[j];
            j++; k++;
        }
    }
    for (k=0; k< r+1-l; k++) a[k+1] = temp[k];
}
```