Practical 4: Elementary Sorting

1. **How many compares does insertion sort make on an input array that is *already sorted*?**

N compares.

Examples of the following:

Constant	Whole Number
Logarithmic	Log(n)
Linear	N
Quadratic	N^2

2. What is a stable sorting algorithm?

In a stable sorting algorithm, the output will always be the same and when there are elements of equal value, the original order in which they appeared will be maintained.

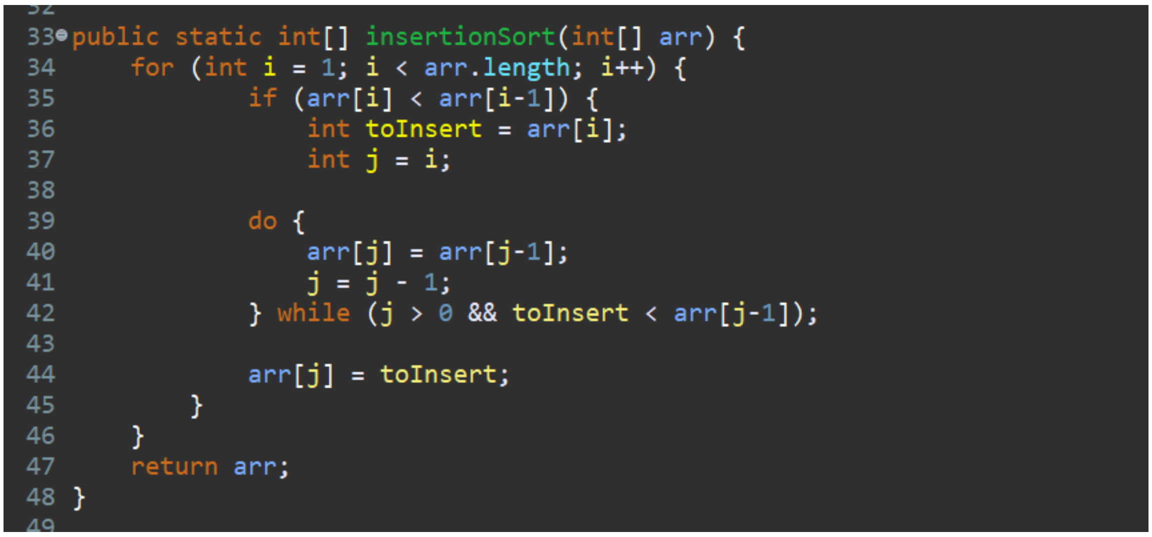
What is an external sorting algorithm?

A. Algorithm that uses tape or disk during the sort

Identify 6 ways of characterizing sorting algorithms?

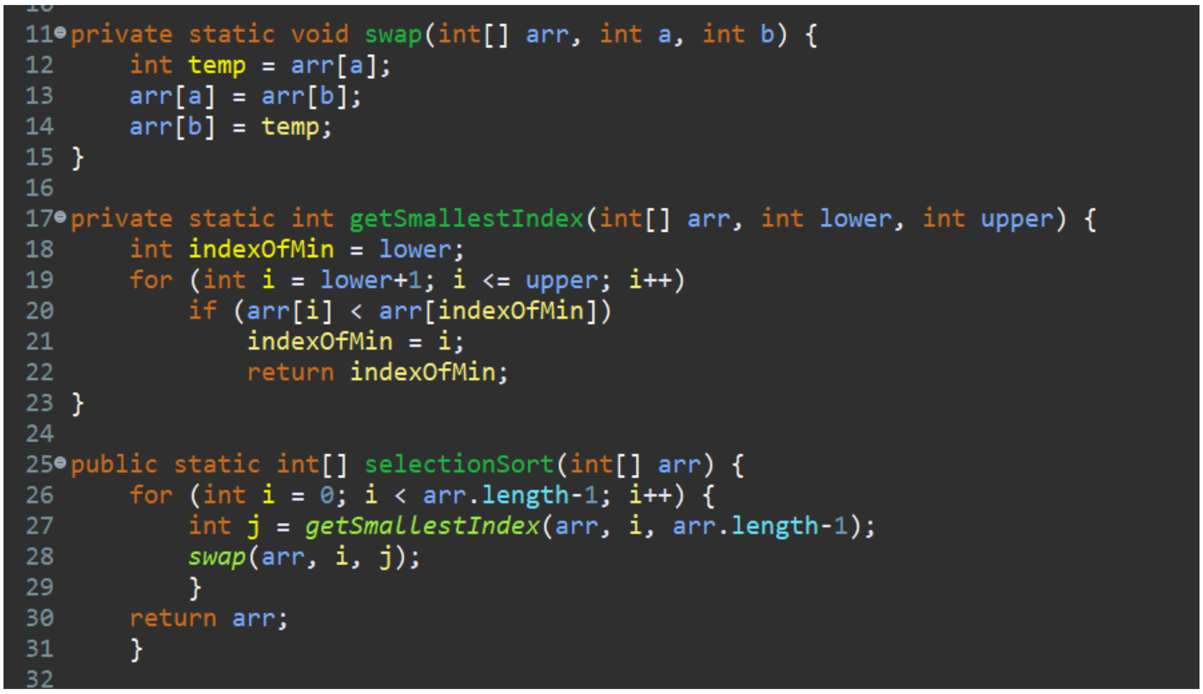
1. Comparison vs. Non-comparison
2. Time complexity
3. Space complexity (in-place vs. out-of-place)
4. Stability
5. Internal vs. External
6. Recursive vs. Non-recursive

Insertion Sort:



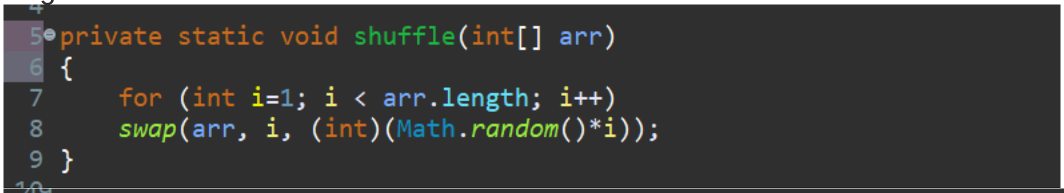
```java
public static int[] insertionSort(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
            if (arr[i] < arr[i-1]) {
                int toInsert = arr[i];
                int j = i;

            do {
                arr[j] = arr[j-1];
                j = j - 1;
            } while (j > 0 && toInsert < arr[j-1]);

            arr[j] = toInsert;
        }
    }
    return arr;
}
```

Insertion Sort:

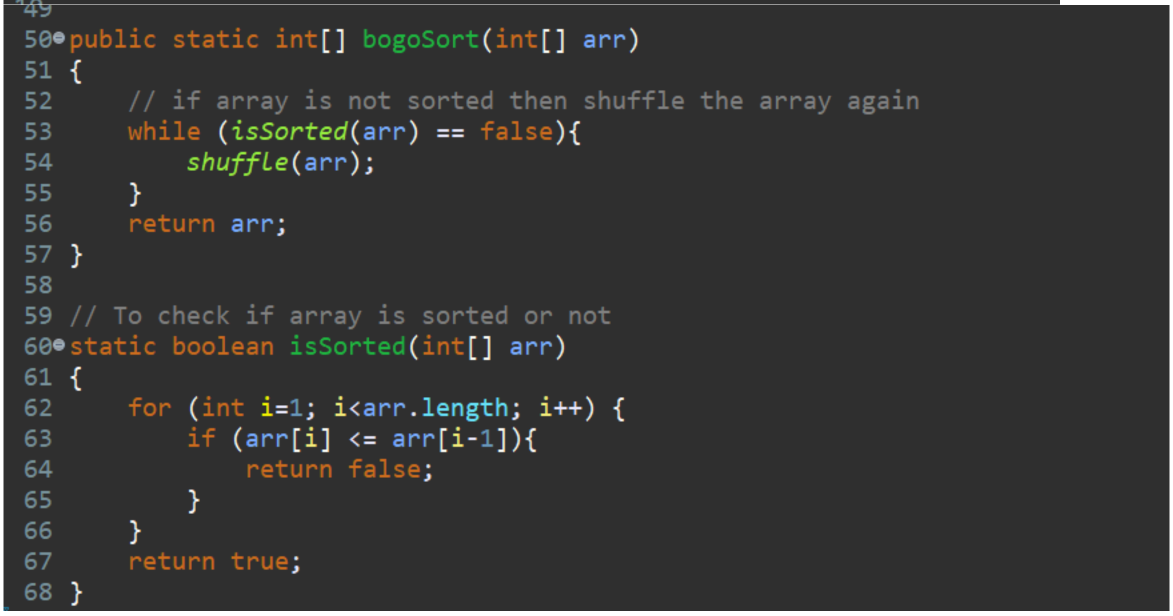


```java
private static void swap(int[] arr, int a, int b) {
    int temp = arr[a];
    arr[a] = arr[b];
    arr[b] = temp;
}

private static int getSmallestIndex(int[] arr, int lower, int upper) {
    int indexOfMin = lower;
    for (int i = lower+1; i <= upper; i++)
        if (arr[i] < arr[indexOfMin])
            indexOfMin = i;
            return indexOfMin;
}

public static int[] selectionSort(int[] arr) {
    for (int i = 0; i < arr.length-1; i++) {
        int j = getSmallestIndex(arr, i, arr.length-1);
        swap(arr, i, j);
        }
    return arr;
    }

```

Bogo Sort



```java
private static void shuffle(int[] arr)
{
    for (int i=1; i < arr.length; i++)
    swap(arr, i, (int)(Math.random()*i));
}
```



```java
public static int[] bogoSort(int[] arr)
{
    // if array is not sorted then shuffle the array again
    while (isSorted(arr) == false){
        shuffle(arr);
    }
    return arr;
}

// To check if array is sorted or not
static boolean isSorted(int[] arr)
{
    for (int i=1; i<arr.length; i++) {
        if (arr[i] <= arr[i-1]){
            return false;
        }
    }
    return true;
}
```

	10	100	1000	Time Complexity
Insertion Sort	0.0	1.0	5.0	O(n^2)
Selection Sort	0.0	0.0	6.0	O(n^2)
Bogo Sort	See below			

The times for bogo sort was incredibly inconsistent, the lowest time I saw (for 10 elements) was 27 ms and the highest was 2306 ms, the 100 element array did not work after 10 minutes (I tried it twice).

The time complexity of bogo sort at its best is O(n) as it iterates through once to see if its equal.

Its worst case is infinity O(∞) as there was nothing stopping it from doing the same combination twice.

I was completely stumped for the average case, so I looked it up. It is O(n! * n), because there is n! possible combinations, and each iteration takes n compares. This doesn’t factor the same combination appearing multiple times, but I accept it as the average case.

Bogo Sort is a joke of an algorithm but I can see how it shows the ineffectiveness of brute-force algorithms.