```
n=an-length
import java.util.Arrays;
public class Sort {
public static void sortA(int[] arr) {
 for(int i = 0; i < arr.length; i += 1) {
   System.out.print(Arrays.toString(arr) +
   int minIndex = i;
   for(int j = i; j < arr.length; j += 1) {
     if(arr[minIndex] > arr[j]) { minIndex = j; }
   \} N + (n-1) + (n-2) ···
   int temp = arr[i];
   arr[i] = arr[minIndex];
   arr[minIndex] = temp;
   System.out.println(Arrays.toString(arr));
public static void sortB(int[] arr) {
 for(int i = 0; i < arr.length; i += 1) {
   System.out.print(Arrays.toString(arr) + " -> ");
   for(int j = i; j > 0; j -= 1) {
     if(arr[j] < arr[j-1]) {
       int temp = arr[j-1];
       arr[j-1] = arr[j];
       arr[j] = temp;
   30+1+2 .... (n-1)
   System.out.println(Arrays.toString(arr));
```

```
jshell> Sort.sort (new int[] { 56, 17, 64, 22, 34, 11 });
[56, 17, 64, 22, 34, 11] -> [11, 17, 64, 22, 34, 56]
[11, 17] 64, 22, 34, 56] -> [11, 17, 64, 22, 34, 56]
[11, 17, 64, 22, 34, 56] -> [11, 17, 22, 64, 34, 56]
[11, 17, 22, 64, 34] 56] -> [11, 17, 22, 34, 64, 56]
[11, 17, 22, 34, 64, 56] -> [11, 17, 22, 34, 56, 64]
[11, 17, 22, 34, 56, 64]
```

```
jshell> Sort.sort \underline{B} (new int[]{ 56, 17, 64, 22, 34, 11 }); [56, 17, 64, 22, 34, 11] -> [56, 17, 64, 22, 34, 11] [56, 17, 64, 22, 34, 11] -> [17, 56, 64, 22, 34, 11] [17, 56, 64, 22, 34, 11] -> [17, 56, 64, 22, 34, 11] -> [17, 56, 64, 22, 34, 11] -> [17, 22, 56, 64, 34, 11] -> [17, 22, 34, 56, 64, 11] -> [17, 22, 34, 56, 64] [17, 22, 34, 56, 64]
```

Which is which?

A: sortA insertion, sortB selection **B:** sortA selection, sortB insertion

Selection Sort: Repeatedly find the minimum element and move it to the **end** of a **sorted prefix** of the array.

Insertion Sort: Repeatedly take the next element and insert it into the **correct ordered position within** a **sorted prefix** of the array.

Worst case complexity?

A: O(n)

B: O(n²)
C: O(n³)

D: O(n * log(n))

E: Something else

 $\binom{n(n+1)}{2} + \dots +$

Worst case complexity?

A: O(n)

B: O(n²) C: O(n³)

D: O(n * log(n)) E: Something else $\Lambda (\Lambda - 1)$ + ...

Best case complexity?

A: O(n)

B: O(n²)

C: O(n3)

D: O(n * log(n))

E: Something else

Best case complexity?

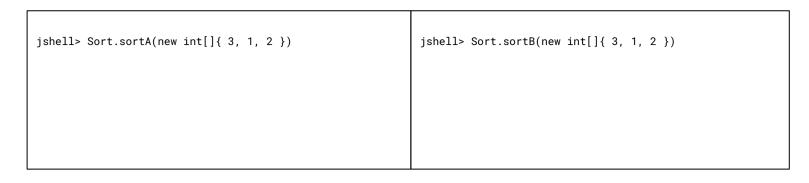
A: O(n)

B: O(n²)

C: $O(n^3)$

D: O(n * log(n))

E: Something else



Selection Sort: What is an improvement you can make to the selection sort algorithm on the front page?

Insertion Sort: What is an.**improvement** you can make to the insertion sort algorithm on the front page?