1

Here are two implementations of the binary search algorithm: one using recursion and the other using an iterative approach.

```java
public static int recursiveBinarySearch(int[] array, int target, int left, int right) {
        if (left <= right) {
                int mid = left + (right - left) / 2;
                if (array[mid] == target)
                        return mid;
                if (array[mid] < target)
                        return recursiveBinarySearch(array, target, mid + 1, right);
                return recursiveBinarySearch(array, target, left, mid - 1);
        }
        return -1;
}
```

```java
public static int iterativeBinarySearch(int[] array, int target) {
        int left = 0;
        int right = array.length - 1;

        while (left <= right) {
                int mid = left + (right - left) / 2;

                if (array[mid] == target)
                        return mid;
                if (array[mid] < target)
                        left = mid + 1;
                else
                        right = mid - 1;
        }
        return -1;
}
```

Compare the two algorithms in terms of space and time complexity. Explain your answer.

✓ **Answer**

Where N is the length of the array:

	Recursive	**Iterative**
Space (number of ints)	$O(4\log_2 N) = O(logN)$	$O(3) = O(1)$

	Recursive	Iterative
Time (by comparisons)	$O(3\log_2 N) = O(\log N)$	$O(3\log_2 N) = O(\log N)$

Since `int` s are passed to the next function in the recursive method, java must allocate more memory for each call, as it is passed by value. The iterative approach does not have this drawback.

Both have the same number of comparisons, with 3 for each iteration, and a worst case of $log_2 N$ iterations.

The more optimized solution would be the `iterativeBinarySearch` as it uses less memory in the same time.

2

a

Is it possible to create an iterator on a custom object that does not implement the Iterable interface? Briefly explain why

✓ **Answer**

Yes, it is possible

If the class has a public way to programatically retrieve all the items within that class's data object in order, then you can write an external class that implements an iterator for the other class through its public methods.

If the class you want to create an iterator for doesn't have public methods to support retrieving all the information in order in any way, or does not contain multiple items to be iterated over, then you cannot create an iterator for that class.

b

Compare and contrast the Iterator and ListIterator interfaces in Java

✓ **Answer**

`Iterator` traverses only forwards while `ListIterator` can traverse in either direction.

`ListIterator` additionally can add, remove, and get the index of items, while `Iterator` cannot.

3

a

Given an array of 6 elements: 2, 24, 69, 8, 1, 15. Please explain step-by-step how selection sort and insertion sort will be performed on the array. You can choose either the recursive or the iterative way

i

Selection Sort

✓ **Answer**

1. Initialize your sorted index variable at `0`
2. Scan all values to the right of (including) that index for the smallest number
3. Swap the item at the index with the smallest number
4. If the index is at the end of the array, finish
5. Increment the index and go to step 2

Iteration: `0`
Array: `[2,24,69,8,1,15]`
Index: `0`
Smallest Index: `4`

Iteration: `1`
Array: `[1,24,69,8,2,15]`
Index: `1`
Smallest Index: `4`

Iteration: `2`
Array: `[1,2,69,8,24,15]`
Index: `2`
Smallest Index: `3`

Iteration: `3`
Array: `[1,2,8,69,24,15]`
Index: `3`
Smallest Index: `5`

Iteration: `4`
Array: `[1,2,8,15,24,69]`
Index: `4`
Smallest Index: `4`

Iteration: `5`
Array: `[1,2,8,15,24,69]`
Index: `5`
Smallest Index: `5`

Iteration: `6`
Array: `[1,2,8,15,24,69]`
Index: `6`
Done

Array: `[1,2,8,15,24,69]`

ii

Insertion Sort

✓ **Answer**

1. Initialize the sorted index at `0`, all items left of including the index, is sorted
2. Recurse backwards from the sorted index, searching for the first smaller node than the node after the sorted index
3. Move the node after the sorted index to after the smaller node, or to the beginning of the array if there are no smaller nodes
4. If there is no next node after the sorted index, finish
5. Increment the sorted index and go to step 2

Iteration: `0`
Array: `[2,24,69,8,1,15]`
Index: `0`
Smaller index: `None`

Iteration: `1`
Array: `[2,24,69,8,1,15]`
Index: `1`
Smaller index: `0`

Iteration: `2`
Array: `[2,24,69,8,1,15]`

Index: `2`
Smaller index: `1`

Iteration: `3`
Array: `[2,24,69,8,1,15]`
Index: `3`
Smaller index: `0`

Iteration: `4`
Array: `[2,8,24,69,1,15]`
Index: `4`
Smaller index: `None`

Iteration: `5`
Array: `[1,2,8,24,69,15]`
Index: `5`
Smaller index: `2`

Iteration: `6`
Array: `[1,2,8,15,24,69]`
Index: `6`
Done

Array: `[1,2,8,15,24,69]`

b

Why is the best case for Insertion Sort O(N)? Give an example of the scenario in which the best case occurs

✓ **Answer**

Insertion sort must always compare every pair of sequential nodes, leading to `N-1` comparisons at minimum when the list is already sorted. When the list is sorted as an input, no swaps are made either.

4

Write pseudo-code/code in Java for the following problem:
Reverse a section of a singly linked list: You have a singly linked list and two integers, left and right, with left ≤ right. Your task is to reverse the nodes of the list from position left to position right and then return the modified list.

For example:

List: 12 → 15 → 24 → 10 → 19

Left: 3

Right: 5

The resulting list should be: 12 → 15 → 19 → 10 → 24

✓ Answer

non-atomic in-place method

```
function reverse(array: SinglyLinkedList, left: int, right: int) {
        let a = array[left-1]
        let filo = new Stack();
        for (i = a; i.next ≠ None; i=i.next {
                filo.add(i)
        }
        for (i = filo.pop(); filo.hasNext(); i = filo.pop()) {
                a.next = i
                a = i
        }
        return array
}
```

pseudo-code in the style of javascript/typescript with java-like class annotations