Quick Sort Answers

Ali Hu, Daniel Huynh, James Tan, Aiden Wang, David Lam

Definition - What is a quick sort?

Quick sort is a highly efficient sorting method which utilizes divide and conquer. It splits the list into halves based on if the element is greater or less then the pivot. The process recursives until the list is sorted.

Key Words: Pivot, partition, recursion, divide and conquer

Algorithm - fill in the blanks

1. All items greater than the pivot are moved to one side, less than, moved to the other side.
2. The list is now divided into two partitions.
3. With recursion, repeat with a new pivot for each side of the list, until the list is sorted.

Passes and Comparisons

1. How many passes and comparisons are required to sort the following cases?

a. "1, 2, 3, 4, 5"
   Passes:4 Comparisons:10
b. "1, 3, 2 ,5, 4"
   Passes:3 Comparisons:6

2. Which case has fewer comparisons and why?

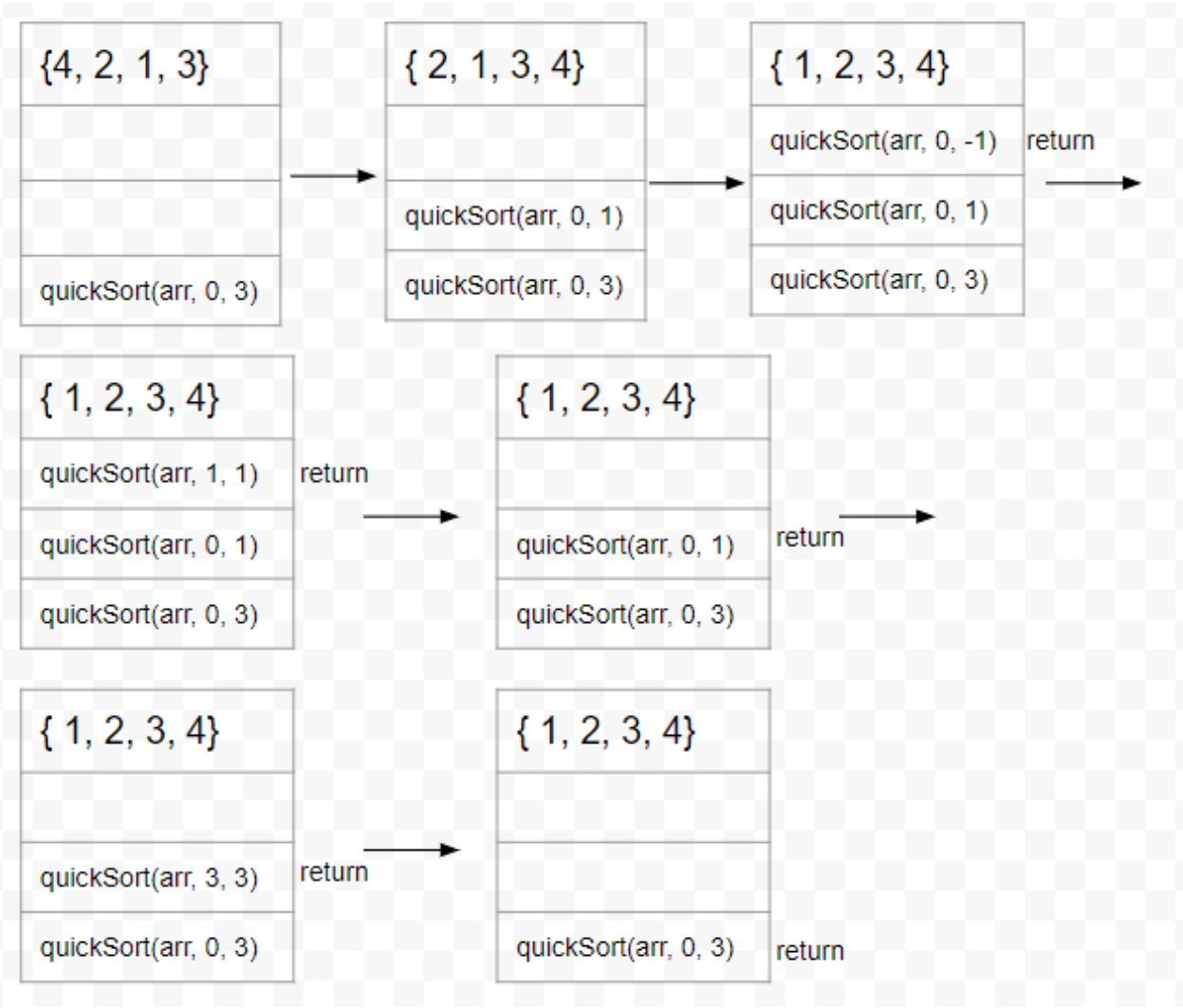
The second case has fewer comparisons. This is because when the last number - 4 - is chosen as the pivot, the right side of 4 is in the correct order, thus reducing the number of comparisons. Whereas for case one, there are no numbers to the right of the pivot, thus it requires more comparisons.

3. Fill in the following quicksort methods with pseudo or Java code

```java
public static void quickSort(int arr[],int low,int high){
        if(low<high){
            int pi = partition(arr,low,high);
            quickSort(arr,low,pi-1);
            quickSort(arr,pi+1,high);
        }
    }


public static int partition(int arr[],int low,int high){
        int pivot = arr[high];
        int i = low-1;
        for(int j=low;j<high;j++){
            if(arr[j]<=pivot){
                i++;
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
        int temp = arr[i+1];
        arr[i+1] = arr[high];
        arr[high] = temp;
        return i+1;
    }
```

4. Draw a stack diagram for quick sorting {4,2,1,3}



1. Write a program that sorts an array of 7 integers in ascending order using quicksort.

```java
/*
* Description: Program that sorts an array of integeres using quick sort
*/
package TestCode;
import java.util.*;
public class QuickSort {
    Run | Debug
    public static void main(String[] args) {

        int[] array = {5,3,2,4,1,6,8}; //array to be sorted
        quickSort(array, 0, array.length-1); //sort the array
        for(int i = 0; i<array.length; i++){ //print the array
            System.out.println(array[i]);
        }
    }

    //quick sort recursive method
    public static void quickSort(int[] array, int low, int high){
        if(low < high){ //base case
            int pivot = partition(array, low, high); //choose pivot and partition array
            quickSort(array, low, pivot-1); //sort the left half
            quickSort(array, pivot+1, high); //sort the right half
        }
    }

    //partition method
    public static int partition(int[] array, int low, int high){
        int pivot = array[high]; //choose pivot
        int i = low - 1; //index of smaller element
        for(int j = low; j<high; j++){ //loop through array
            if(array[j] <= pivot){ //if element is smaller than pivot
                i++; //increment index of smaller element
                int temp = array[i]; //swap elements
                array[i] = array[j];
                array[j] = temp;
            }
        }
        int temp = array[i+1]; //swap elements
        array[i+1] = array[high];
        array[high] = temp;
        return i+1; //return index of pivot
    }
}
```

2. Write a program that sorts an array of names in ascending order using quicksort.

```java
/*
* Descrption: sort a array of names in acdecning order using quick sort
*/
package TestCode;
public class SortName {
    Run | Debug
    public static void main(String[] args) {

        String[] names = {"James", "Aiden", "David", "Ringo", "Pete"}; //array of names
        quickSort(names, 0, names.length - 1);  //call the sort method
        for (int i = 0; i < names.length; i++) { //pring the names
            System.out.println(names[i]);
        }
    }

    //quick sort recursive method
    public static void quickSort(String[] names, int left, int right) {
        if (left < right) { //base case
            int pivot = partition(names, left, right); //get the pivot
            quickSort(names, left, pivot - 1); //sort the left side
            quickSort(names, pivot + 1, right); //sort the right side
        }
    }

    //partition the names
    public static int partition(String[] names, int left, int right) {
        String pivot = names[left]; //get the pivot
        while (left < right) {  //loop until left and right meet
            while (left < right && names[right].compareTo(pivot) >= 0) { //find the first element that is less than the pivot
                right--;
            }
            names[left] = names[right]; //swap the first element that is less than the pivot
            while (left < right && names[left].compareTo(pivot) <= 0) {  //find the first element that is greater than the pivot
                left++;
            }
            names[right] = names[left]; //swap the first element that is greater than the pivot
        }
        names[left] = pivot; //swap the pivot with the first element that is greater than the pivot
        return left; //return the index of the pivot
    }
}
```