PARALLEL ALGORITHMS ASSIGNMENT-III

GROUP-2

SAI TEJASWEE REDDY PASHAM

SOWJANYA POLEPALLY

MANOJ KUMAR RAVILLA

(Q). Design a parallel sorting technique and implement it in spark.

(A). Following is a sequential Quick sort code: time complexity *O(n log n)*

```
public class Quick_Sort {
    private int array[];
    private int length;
    public void sort(int[] inputArr) {
        if (inputArr == null || inputArr.length == 0) {
            return;
        }
        this.array = inputArr;
        length = inputArr.length;
        quickSort(0, length - 1);
    }

    private void quickSort(int lowerIndex, int higherIndex) {
        int i = lowerIndex;
        int j = higherIndex;
        int pivot = array[lowerIndex+(higherIndex-lowerIndex)/2];

        while (i <= j) {
            while (array[i] < pivot) {
                i++;
            }
            while (array[j] > pivot) {
                j--;
            }
            if (i <= j) {
                exchangeNumbers(i, j);
                   i++;
                j--;
            }
        }
        if (lowerIndex < j)
            quickSort(lowerIndex, j);
        if (i < higherIndex)
            quickSort(i, higherIndex);
    }

    private void exchangeNumbers(int i, int j) {
        int temp = array[i];
        array[i] = array[j];
        array[j] = temp;
    }

    public static void main(String a[]){
        Quick_Sort sorter = new Quick_Sort();
        int[] input = {24,2,45,20,56,75,2,56,99,53,12};
        sorter.sort(input);
        for(int i:input){
            System.out.print(i);
            System.out.print(" ");
        }
    }
}
```

Output:

```
2 2 12 20 24 45 53 56 56 75 99
```

Pseudo Code for Parallel Quick Sort time complexities *O(n log n)* :

```
quicksort([8, 14, -8, -9, 5, -9, -3, 0, 17, 19]);
Take an Array of Elements a[], integer pivot, boolean values (greater,lesser,equal)
if (#a [] < 2) then a
else
    let pivot   = a[#a/2];
foreach(e : a[]){
if( e < pivot)
    lesser  = e ;
else if (e == pivot)
    equal   = e;
else if(e > pivot)
    greater= e;
}
    result  = {v : [lesser,greater]};
    result[0] ++ equal ++ result[1];
```

Output:

```
result = [-9, -9, -8, -3, 0, 5, 8, 14, 17, 19]
```

comparative evaluation of Quick Sort with other techniques:

- Bubble Sort is not suitable in any circumstance. Time required to perform bubble sort on ‘n’ numbers increase as square of ‘n’. Thus it is quite slow.
- Insertion Sort is suitable for small files, but again it is an $O(n^2)$ algorithm, but with a small constant. It works best when the file is already almost sorted.

- Quick Sort is an O(n*log(n)) algorithm on an average case and an O(n^2) algorithm in the worst case scenario. This algorithm is used when the list is large and time is premium.