UPES

UNIVERSITY OF TOMORROW

Design & Analysis of Algorithm (Lab)

Name: Ananya

SAPID: 590013832

B-33

Submitted to: Mr.Aryan Gupta

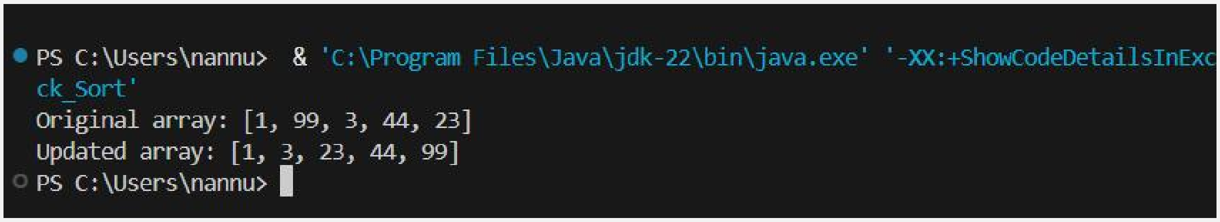
https://github.com/ananya438/DAALAB_ANANYA-590013832

QUICK SORT

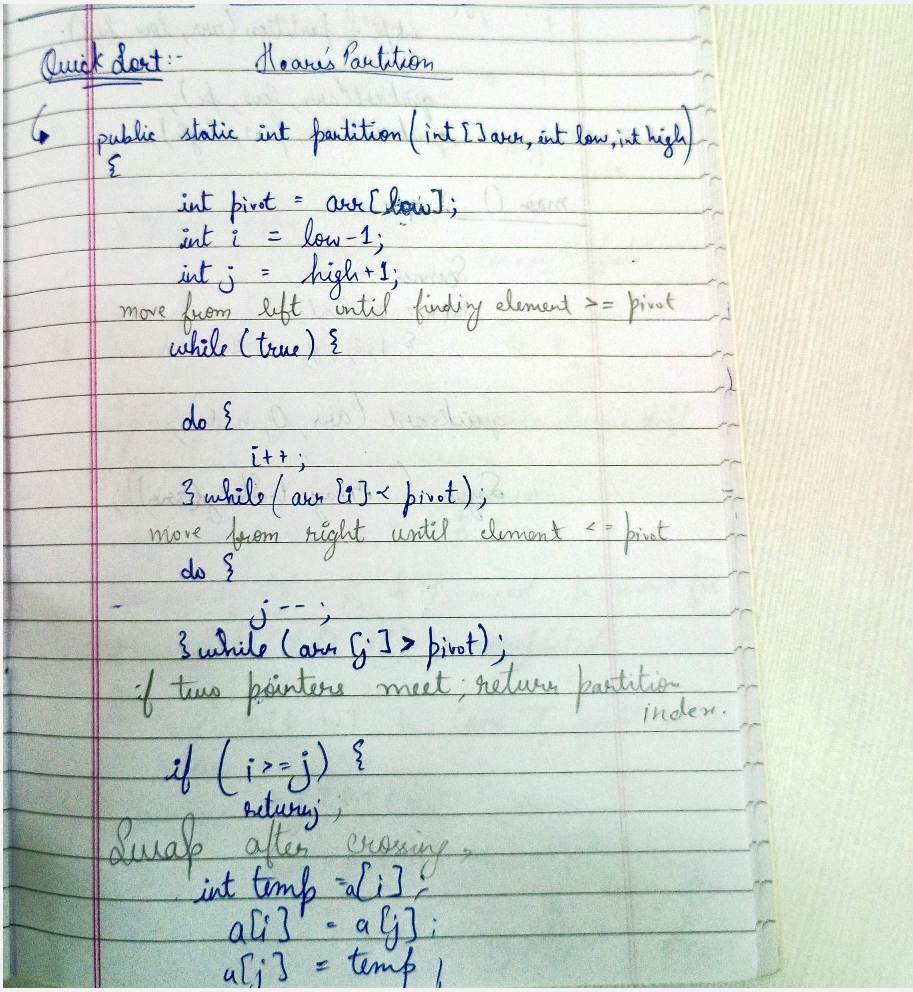
```
import java.util.*;
public class Quick_Sort{
    public static int partition(int [] arr,int low,int high){
        int pivot=arr[low];
        int i=low-1;
        int j=high+1;
        while(i<j){
            do{
                i++;
            }while(arr[i]<pivot);
            do{
                j--;
            }while(arr[j]>pivot);
        if(i<j){
            int temp=arr[i];
            arr[i]=arr[j];
            arr[j]=temp;
        }
    }
        return j;
    }
    public static void quickSort(int [] arr,int low,int high){
        if(low<high){
            int pi = partition(arr,low,high);
        quickSort(arr, low, pi);
        quickSort(arr, pi+1, high);
        }
    }


    public static void main(String[] args) {
        int arr[]={1,99,3,44,23};
        int n = arr.length;
        System.out.println("Original array: " + Arrays.toString(arr));
        quickSort(arr,0, n-1);
        System.out.println("Updated array: " + Arrays.toString(arr));
    }
}
```

O/P:



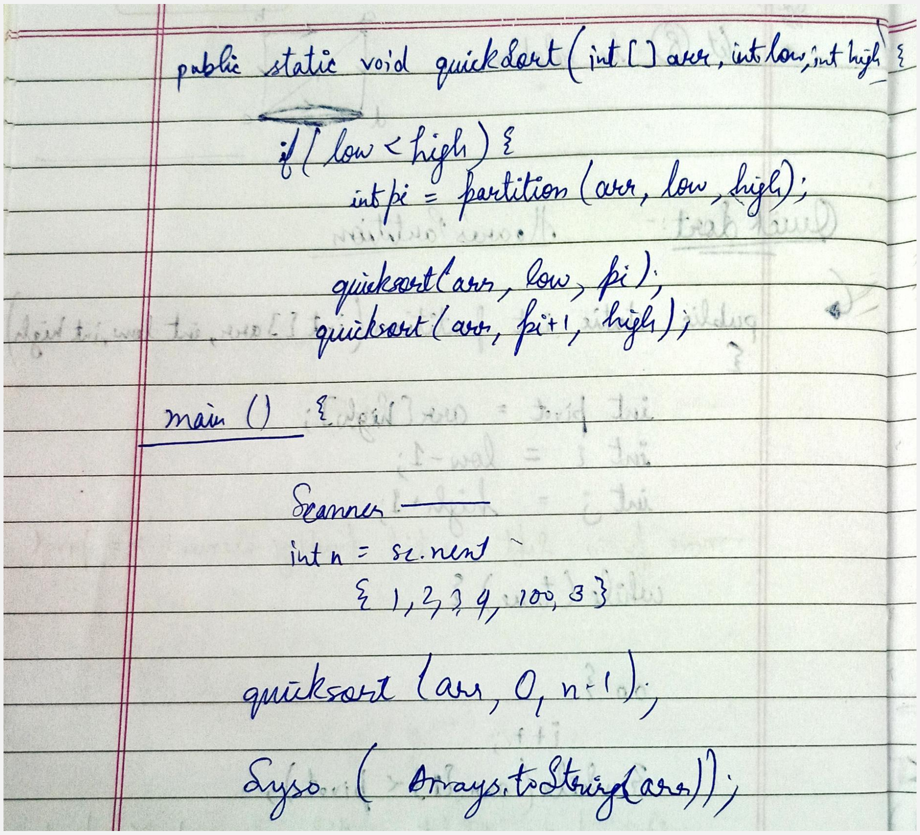
```
● PS C:\Users\nannu>  & 'C:\Program Files\Java\jdk-22\bin\java.exe' '-XX:+ShowCodeDetailsInExc
  ck_Sort'
  Original array: [1, 99, 3, 44, 23]
  Updated array: [1, 3, 23, 44, 99]
○ PS C:\Users\nannu> 
```



Quick Sort:- Hoare's Partition

```
public static int partition(int [] arr, int low, int high)
{
      int pivot = arr[low];
      int i = low-1;
      int j = high+1;
   move from left until finding element >= pivot
     while (true) {

      do {
           i++;
      } while (arr [i] < pivot);
    move from right until element <= pivot
      do {
           j--;
      } while (arr [j] > pivot);
   if two pointers meet; return partition
                                         index.
     if (i>=j) {
          return j;
 Swap after crossing.
        int temp = a[i];
            a[i] = a[j];
            a[j] = temp;
```



```
public static void quickSort(int [] arr, int low, int high {

        if ( low < high) {
               int pi = partition (arr, low, high);

               quicksort(arr, low, pi);
               quicksort (arr, pi+1, high);

main () {

        Scanner ______
        int n = sc.next
               { 1, 2, 3, 9, 100, 3 }

    quicksort (arr, 0, n-1);

    Syso ( Arrays.toString(arr));
```

* Quick Sort:

Recurrence:

$$T(n) = 2T(n/2) + O(n)$$

O(n) comes from partitioning

$$T(n) = O(n \log n)$$

Worst Case :-

(Already Sorted or Reverse Sorted with bad pivot choice

: Partition divides array into one element & rest (n-1)

Recurrence:

$$T(n) = T(n-1) + O(n)$$

Solving

$$\therefore T(n) = O(n^2)$$

Can be avoided using random pivot or median of three pivot solution.