05 June 2022
23rd Wk • 156-209
Sunday

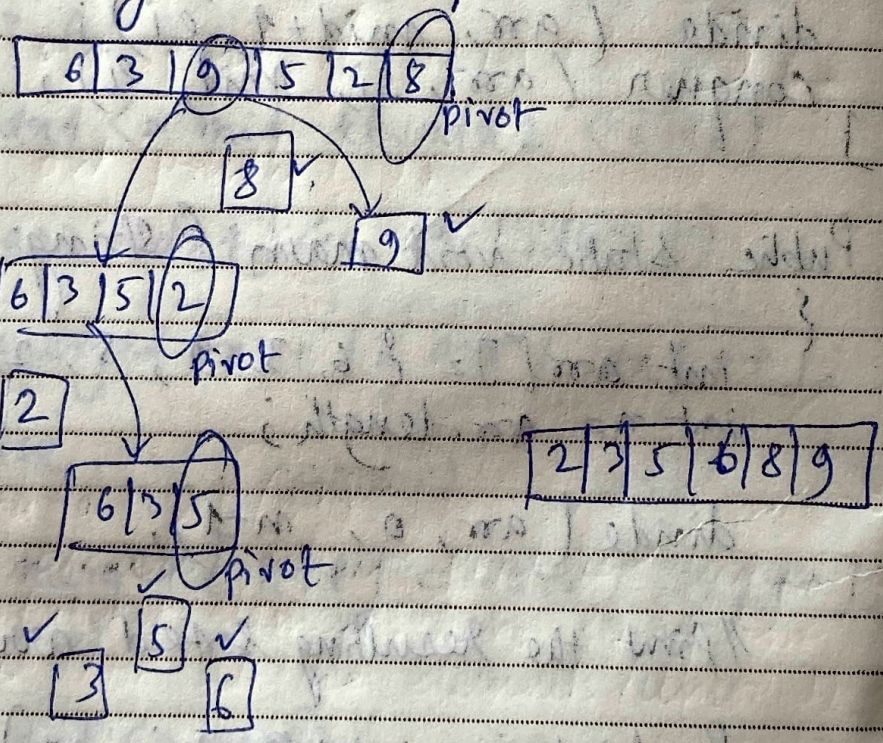
Quick sort

There are 2 keywords - 'pivot' & 'partition' in Quick sort Array.

Usually, there are four ways to choose the pivot - (i) random, (ii) median, (iii) 1st element, (iv) last element we choose any of these as our pivot and proceed accordingly.

In our case, we shall proceed by choosing the last element of our array as the pivot.

E.g →



✓ So, basically in every step, we choose the last element as the pivot. Now, we compare all other elements with the pivot. If they are smaller, place them in left side and if they are greater, place them in right side.

✓ Similarly, we again get an array. Apply the same recursive quick sort method to it. In the end, we shall get an sorted array.

2022 JULY

June 24th Wk • 157-208

06 Monday

```
class QuickSort
{
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++;                    // swap elements
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
    i++;                            // fit/make pivot in the right/correct position
    int temp = arr[i];
    arr[i] = pivot;
    arr[high] = temp;
    return i; // pivot index
}
public static void quickSort(int arr[], int low, int high)
{
    if (low < high)
    {
        int pidx = partition(arr, low, high);

        quickSort(arr, low, pidx-1);
        quickSort(arr, pidx+1, high);
    }
}
```

```
Public static void main (String args[])
{
    int[] arr = { 6, 3, 9, 5, 2, 8 };
    int n = arr.length;
    quickSort (arr, 0, n-1);
    for (int i=0; i<n; ++)  // Print the result array
    {
        System.out.print ((arr[i] + " ");
    }
    System.out.println ();
}
```

Time Complexity :-

Worst case scenario :- $O(n^2)$.

worst case occurs when pivot is always the smallest or the largest element. Then, every time, the whole loop runs for all the elements (unsorted) and thus $O(n^2)$.

Average Case Scenario :- $O(n \log n)$, which is similar to the worst case scenario of merge sort.

E.g → [1|2|3|4|5] n ∴ $n + (n-1) + (n-2) + \dots + 1 = \frac{n(n+1)}{2}$

[1|2|3|4] ← $n-1$ ≈ $O(n^2)$

[1|2|5] $n-2$ ∴ we can see if the array is completely sorted in ascending or descending order & we choose the last element as pivot, worst case occurs.

[1|2]

1 2

1

Comparison between Quick Sort & Merge Sort

✓ The usage of Quick sort or merge sort depends upon the situation.

In case, where we have the privilege to use extra memory space, we can use merge sort, where the worst case scenario has a time complexity of $O(n \log n)$.

Now, if we need to keep the memory usage, we can use quick sort. Also, worst case scenario occurs very rarely. So, in case of average time complexity it's said as merge sort and with low memory usage.

✓ In merge sort, we were making a new array and so it takes more memory space. whereas, in quick sort, no new array is created and so it takes little less memory than it's counterpart.