Quick Sort

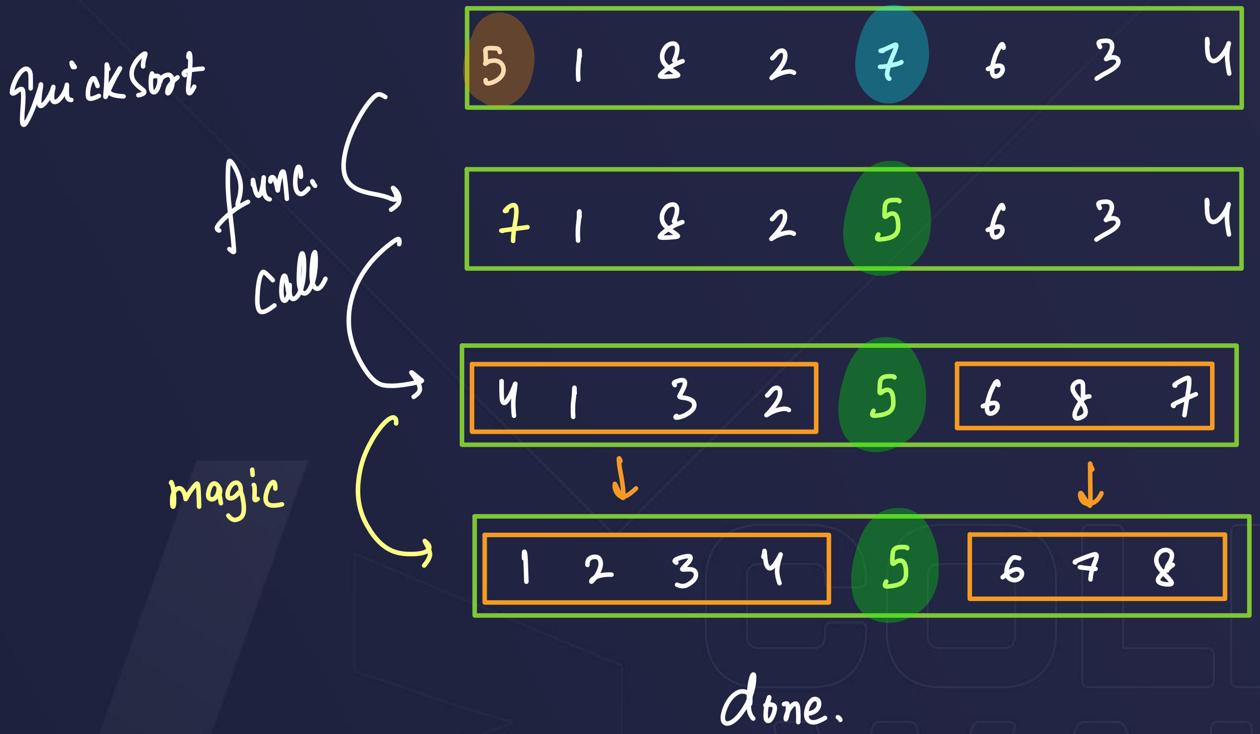
Lecture-33

Raghav Garg

Today's checklist

1) Quick Sort Algorithm
2) Example Explanation
3) Quick Sort Time and Space Complexity
4) Randomised Pivot point
5) Stability
6) Applications of Quick Sort
7) Difference between Quick Sort and Merge Sort

QuickSort Algorithm



How to choose the pivot point

1) As of now, we will take the first element as pivot.

starting idx

2) To set the pivot → · find pivot idx & then swap.

→ just find the number of smaller elements than arr[stIdx]

0	1	2	3	4	5	6	7
5	1	8	2	7	6	3	4

st (index 0) end (index 7)

Count = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4

$\boxed{\text{pivot idx} = \text{count} + st}$

Partition Algorithm

0	1	2	3	4	5	6	7
4	1	3	2	5	7	8	6
st				pi			end

5	6	7
7	8	6
st		end

count = 1

pivot Idx = count + st = 6

Code and dry run

```
void quicksort(int arr[], int si, int ei){
1   if(si>=ei) return; // base
    // 5,1,8,2,7,6,3,4
2   int pi = partition(arr,si,ei);
    // 4 1 3 2 5 7 8 6
3   quicksort(arr,si,pi-1);
4   quicksort(arr,pi+1,ei);
}
int main(){
```

0 1 2 3 4 5 6 7

1 2 3 4 5 6 7 8

si ↓ ei

4 1 3 2 5 6 8 7

si pi ei

0 1 2 3 4 5 6 7

1 2 3 4 5 6 8 7

si ei

6 7

8 7

si ei

7 8

si pi ei

Time and Space complexity

```
void quicksort(int arr[], int si, int ei){
    if(si>=ei) return; // base
    // 5,1,8,2,7,6,3,4
    int pi = partition(arr,si,ei);
    // 4 1 3 2 5 7 8 6
    quicksort(arr,si,pi-1);
    quicksort(arr,pi+1,ei);
}
```

```
int partition(int arr[], int si, int ei){
    int pivotElement = arr[si];
    int count = 0;
    for(int i=si+1;i<=ei;i++){
        if(arr[i]<=pivotElement) count++;
    }
    int pivotIdx = count + si;
    swap(arr[si],arr[pivotIdx]);
    int i = si;
    int j = ei;
    while(i<pivotIdx && j>pivotIdx){
        if(arr[i]<=pivotElement) i++;
        if(arr[j]>pivotElement) j--;
        else if(arr[i]>pivotElement && arr[j]<=pivotElement){
            swap(arr[i],arr[j]);
            i++;
            j--;
        }
    }
    return pivotIdx;
}
```

Let the QS code be run 'x' times

Two in each partition → $2x(ei-si)$

Time and Space complexity

```
void quicksort(int arr[], int si, int ei){
    if(si>=ei) return; // base
    // 5,1,8,2,7,6,3,4
    int pi = partition(arr,si,ei);
    // 4 1 3 2 5 7 8 6
    quicksort(arr,si,pi-1);
    quicksort(arr,pi+1,ei);
}
int main(){
```

→ O(1) ✗

→ O(n) ✗

→ O(log n) ✓

```
int partition(int arr[], int si, int ei){
    int pivotElement = arr[si];
    int count = 0;
    for(int i=si+1;i<=ei;i++){
        if(arr[i]<=pivotElement) count++;
    }
    int pivotIdx = count + si;
    swap(arr[si],arr[pivotIdx]);
    int i = si;
    int j = ei;
    while(i<pivotIdx && j>pivotIdx){
        if(arr[i]<=pivotElement) i++;
        if(arr[j]>pivotElement) j--;
        else if(arr[i]>pivotElement && arr[j]<=pivotElement){
            swap(arr[i],arr[j]);
            i++;
            j--;
        }
    }
    return pivotIdx;
}
```

Time and Space complexity

Problem
↓
arr[si] → pivot element
pi → swap

Time Complexity :

Avg. Case → $O(n \cdot \log n)$

Worst Case → $O(n^2)$

Worst Case :

5 4 3 2 1 or 1 2 3 4 5

1 4 3 2 5

1 4 3 2 5

1 2 3 4 5

Randomized Pivot point

pivot Element → arr[si]

pivot Element → $arr\left[\frac{si+ei}{2}\right]$

$\boxed{5 \quad 4 \quad 3 \quad 2 \quad 1}$

↓

$\boxed{1 \quad 2} \quad 3 \quad \boxed{4 \quad 5}$

$[\,] \quad 1 \quad \boxed{2} \quad 3 \quad [\,] \quad 4 \quad \boxed{5}$

→ T.C. → $\boxed{O(n \log n)}$

Stability of Quick Sort → Not Stable

Stable Sorts
↓
B.S, I.S., M.S

Unstable Sorts
↓
S.S, Q.S

2 1 8 5 5' 6 3 4

2 1 4 3 5' 5 6 8

1 2 3 4 5' 5 6 8
↓
1 2 3 4 5 5' 6 8

Application of Quick Sort

↓

1) Internal sorting uses variation of quick sort

2) Quick Select

3) Whenever there is no need of stability, we use quick sort.

Merge Sort vs Quick Sort

B.S, S.S, I.S

	Merge Sort	Quick Sort
T.C.	$O(n \log n)$	$O(n \log n)$
S.C.	$O(n)$	✓ $O(\log n)$ → In Place sorting
Stability	✓	✗

Stability ↓

Linked Lists

Inversion Count

Quick Select

Ques : Write a Program to find Kth smallest element in an array using QuickSort.

Selection Sort → $O(K^*n)$ ← Bubble & Insertion

Merge Sort → $O(n\log n)$

Quick Select → $O(n)$ (avg. case) → Worst Case → $O(n^2)$

K = 5

5	1	8	2	7	6	3	4

1	2	8	5	7	6	3	4

1 2 | 8 5 7 6 3 4 |

1 2 | 4 5 3 6 7 8 |

1 2 | 4 5 3 6 | 7 8

1 2 | 4 3 5 6 | 7 8

return

Quick Select Time Complexity Discussion

$n=8$

```
int kthSmallest(int arr[], int si, int ei, int k){
    int pi = partition(arr,si,ei);
    if(pi+1==k) return arr[pi];
    else if(pi+1 < k) return kthSmallest(arr,pi+1,ei,k);
    else return kthSmallest(arr,si,pi-1,k);
}
```

Rough

Total no. of ops $\rightarrow n + \frac{n}{2} + \frac{n}{4} \dots 1$

$$= 1+2+4\dots\frac{n}{2}+n = 2n-1 \sim n$$

$$= \boxed{T.C. = O(n)}$$

PW SKILLS

THANK YOU

Classwork → k^{th} smallest → $\boxed{\text{LeetCode}}$