Sorting 2: QuickSort & Comparator Problems

Agenda:

1. Sort 0-1 array
2. Pivot partition
3. QuickSort
4. Comparator problem

Sort 0-1

Given an array of 0s & 1s in random order.
Sort the array [all 0s in left and all 1s in right]

A = [0 1 0 0 1 1 0 1 0]

o/p : [0 0 0 0 0 1 1 1 1]

Idea 1 : use in-built sort → TC = O(nlogn)

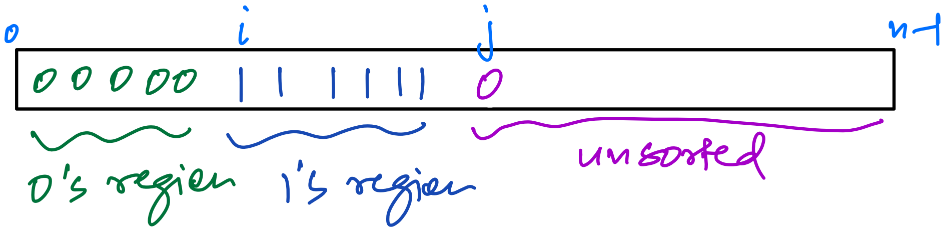
Idea 2 : use count sort

iterating 2 times

↳ count all 0s & 1s and finally populate the array

TC = O(N)
SC = O(1)

Idea3 : try to solve in single loop



0 to i-1 is 0's region

i to j-1 is 1's region

j to n-1 is unsorted

initially, $i=0$
$j=0$

A[j] == 1	A[j] == 0
// increase region of 1 j++	// swap 0 in 0's region & increase 0's & 1's region swap (A[i] , A[j]) i++ , j++

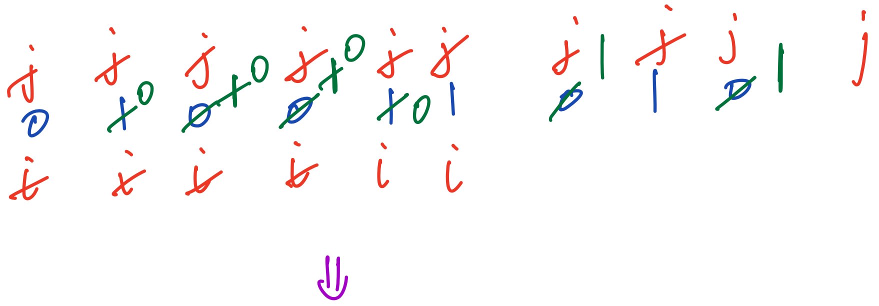
```
sort01 ( A[], n ) {
    i=0, j=0
    while (j<n) {
        if ( A[j] == 1 ) {
            j++
        }
```

$TC = O(N)$

$SC = O(1)$

$\boxed{\text{one iteration}}$

```
        else {
            swap(A[i], A[j])
            i++, j++
        }
    }
}
```



0 0 0 0 0 1 1 1 1

HW: sort 0's, 1's & 2's

Partition

Given an array, consider first element as pivot, rearrange array s.t. all elements < pivot are on left side of pivot & rest are on right side of pivot.

A = [54 26 93 17 77 31 44 55]

pivot

partitioning

< pivot	pivot	> pivot
26 17 31 44	54	93 77 55

divide & conquer

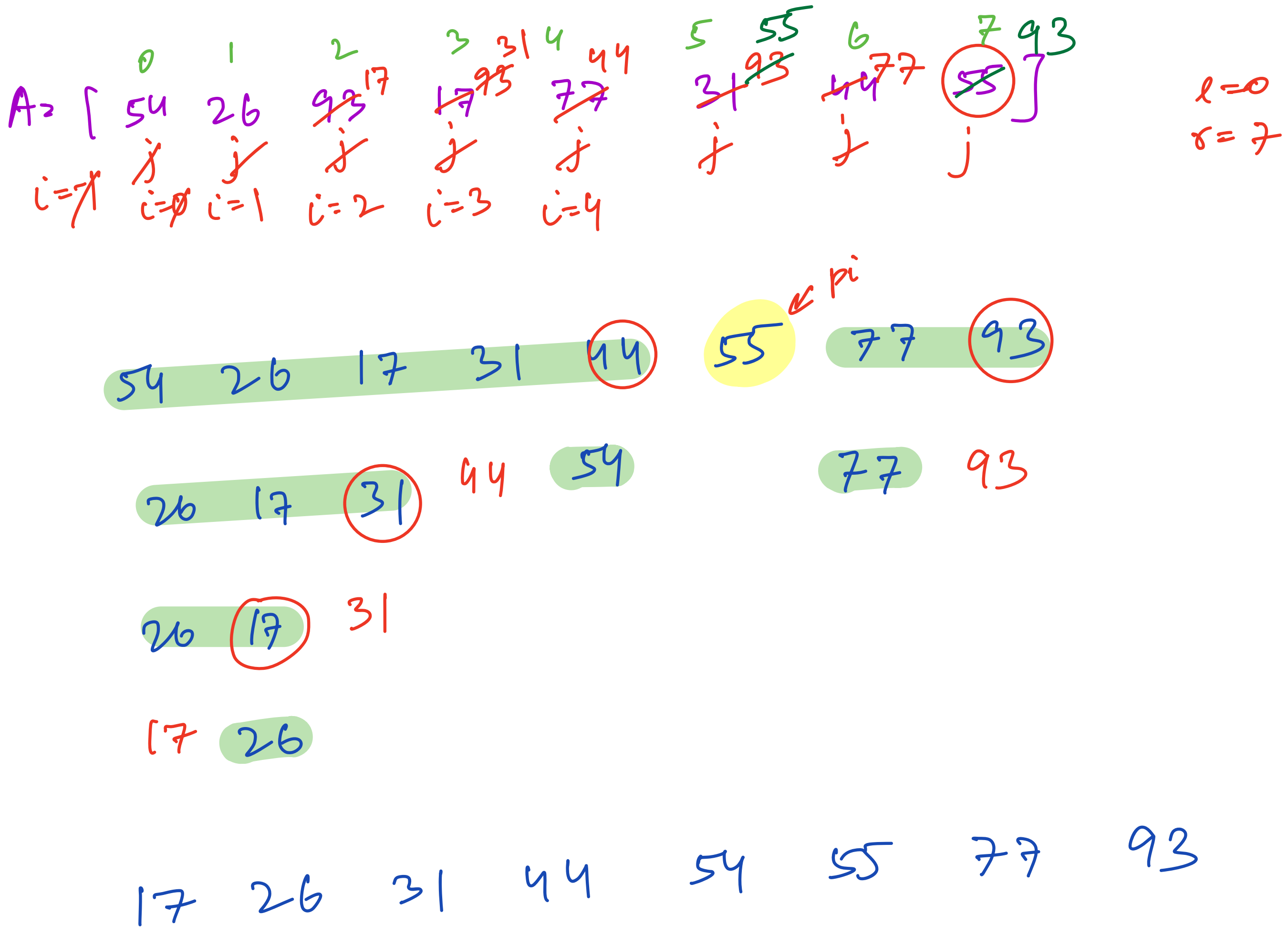
Arrange smaller elements on left

index 0 to i , all elements are smaller than pivot

```
                 0    n-1
int partition(A, l, r) {
    pivot = A[r]   // random
    i = l-1        // 0-i elements < pivot
    for (j = l to (r-1)) {
        if (A[j] < pivot) {
            i++
            swap(A[i], A[j])
        }
    }
    swap(A[i+1], A[r])
    return i+1
}
```

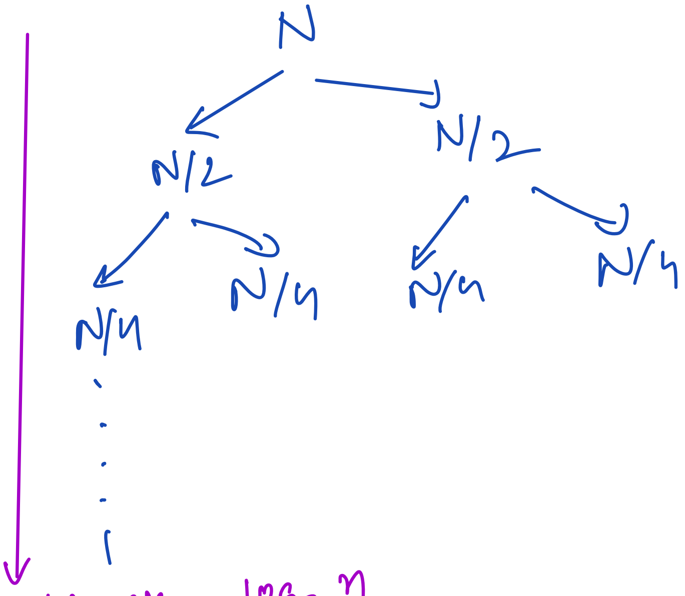
TC = O(N)

SC = O(1)



```
void quickSort(A[], l, r) {
                    0  n-1
    if (l < r) {
        pi = partition(A, l, r);
        quickSort(A, l, pi-1)
        quickSort(A, pi+1, r)
    }
}
```

Best Case

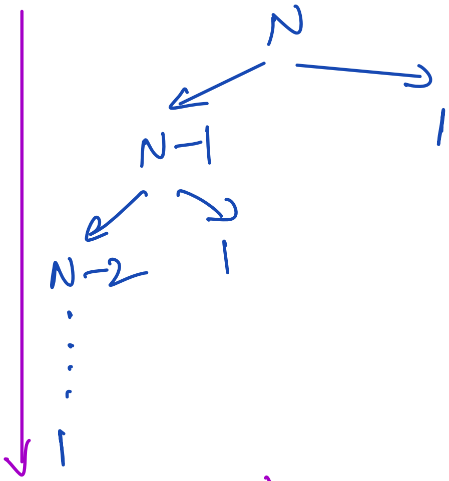


levels = $\log_2 n$

TC = $O(N \log N)$

SC = $O(\log N)$

Worst Case



levels = N

TC = $O(N^2)$

SC = $O(N)$

Randomized Quick Sort

Always choose pivot element at random.

Given N elements, probability of random element being minimum = $1/N$

Again, probability of minimum element = $1/N-1$

⋮

$1/N-2$

⋮

$$\frac{1}{N} \times \frac{1}{N-1} \times \frac{1}{N-2} \times \cdots = \frac{1}{N!}$$

$N = 10$, $\frac{1}{10!} = ?$ $2.7 \times 10^{-7} \approx 0$

Hence, on averge TC of quickSort = O(NlogN)

Comparator

Java, Py, JS, C#, Ruby

```
int compare (first, second) {

    return ──> -ive  =>  first should come before second
           ──> 0     =>  both are equal
           ──> +ive  =>  first should come after second

}
```

eg → sort in ascending ⇒ return (first - second)

descending ⇒ return (second - first)

```
bool compare (first, second) {        C++

    return ──> true   =>  first should come before second
           ──> false  =>  first should come after second
}
```

Question

Given an integer array, Sort the data w.r.t count of factors. If factors are same, sort based on magnitude.

A= [9 3 10 16 4]

#factors → 3 2 4 5 3

sorted → 3 4 9 10 16

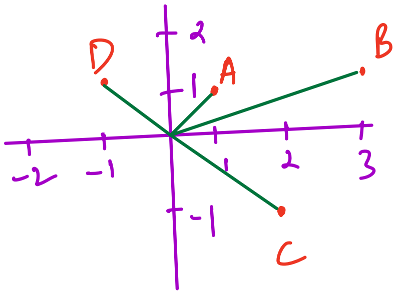
Java

```
Collections.sort( A, new comparator<Integer>() {
        @override
        public int compare( Integer f, Integer s ) {
            if( factors(f) == factors(s)) {
                return f - s;
            }
            else {
                return factors(f) - factors(s);
            }
        }
}
```

Question

Given an array of points in 2D-plane. $p[i] = (x_i, y_i)$

Return K closest points to origin (0,0).



K=2

o/p → (1,1) (-1,1)

compare → (x_1, y_1) (x_2, y_2)

distance from origin = $\sqrt{x^2+y^2}$

compare → $\sqrt{x_1^2+y_1^2}$ $\sqrt{x_2^2+y_2^2}$

compare → $(x_1^2+y_1^2)$ $(x_2^2+y_2^2)$

```
int compare ( f , s ) {
    return (f.x^2 + f.y^2) - (s.x^2 + s.y^2)

}
```

Question

Given an array of non-negative integers, arrange them s.t. we get largest no. & return it.

$A = [2, 5, 7]$

ans = "752"

$A = [10, 2] \xrightarrow[\text{in desc}]{\text{sort}}$ "102"

ans = "210"

$A = [3, 30, 34, 5, 9] \xrightarrow[\text{in desc}]{\text{sort}}$ "3430953"

ans = "9534330"

let's use custom sorting

```
int compare ( int f, int s ) {
        s1 = string(f) + string(s)
        s2 = string(s) + string(f)
        if (s1 > s2)
              return -1
        else  return 1
```

3

A = [3, 30, 34, 5, 9]

sorted = [9, 5, 34, 3, 30]

↓ create string

ans = "9534330"

Python

```
A = sorted( A, key = functools. cmp-to-key( compare ))

def compare( f, s ):
    return f - s
```