Sorting 2 : QuickSort & Comparator Problems

Agenda

- Pivot partition
- Quick Sort
- Comparator problems

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

partioning

< pivot	pivot	>= pivot

26 17 31 44 54 93 77 55

A = [10 13 7 8 25 20 23 5]

left = [7 8 5]

right = [13 25 20 23]

Arrange smaller elements on left

index 0 to i contains smaller elements than pivot

Code

```
                    0   n-1
int partition (A, l, r) {
    pivot = A[r]      // rightmost element as pivot
    i = l-1           // l-i elements < pivot

    for( j = l to r-1) {
        if(A[j] < pivot) {
            i++
            swap( A[i] , A[j])
        }
    }

    // [0-i) elements < pivot
       [i+1 - r-1]    > pivot
       a[r]           = pivot
```

TC = O(N)

O(r-l)

SC = O(1)

```
swap (A[i+1], A[r])
return i+1
}
```

	0	1	2	3	4	5	6	7
A = 1	54	26	~~93~~ 17	~~17~~ 93 31	~~77~~ 44	~~31~~ 93	~~44~~ 77	~~55~~ 93

$l = 0$

$r = 7$

~~i=1~~

~~i=0~~ ~~i=1~~ ~~i=2~~ ~~i=3~~ i=4

pivot

j

return i+1

54 26 17 31 44 55 77 93

pi

26 17 31 44 54 77 93

26 17 31

17 26

17 26 31 44 54 55 77 93

Quicksort

```
void quickSort (A[], l, r) {
    if (l < r) {
        pi = partition(A, l, r)
```

0 n-1

Divide & conquer strategy

```
        quicksort(A, l, pi-1)
        quicksort(A, pi+1, r)
    }
}
```

OR

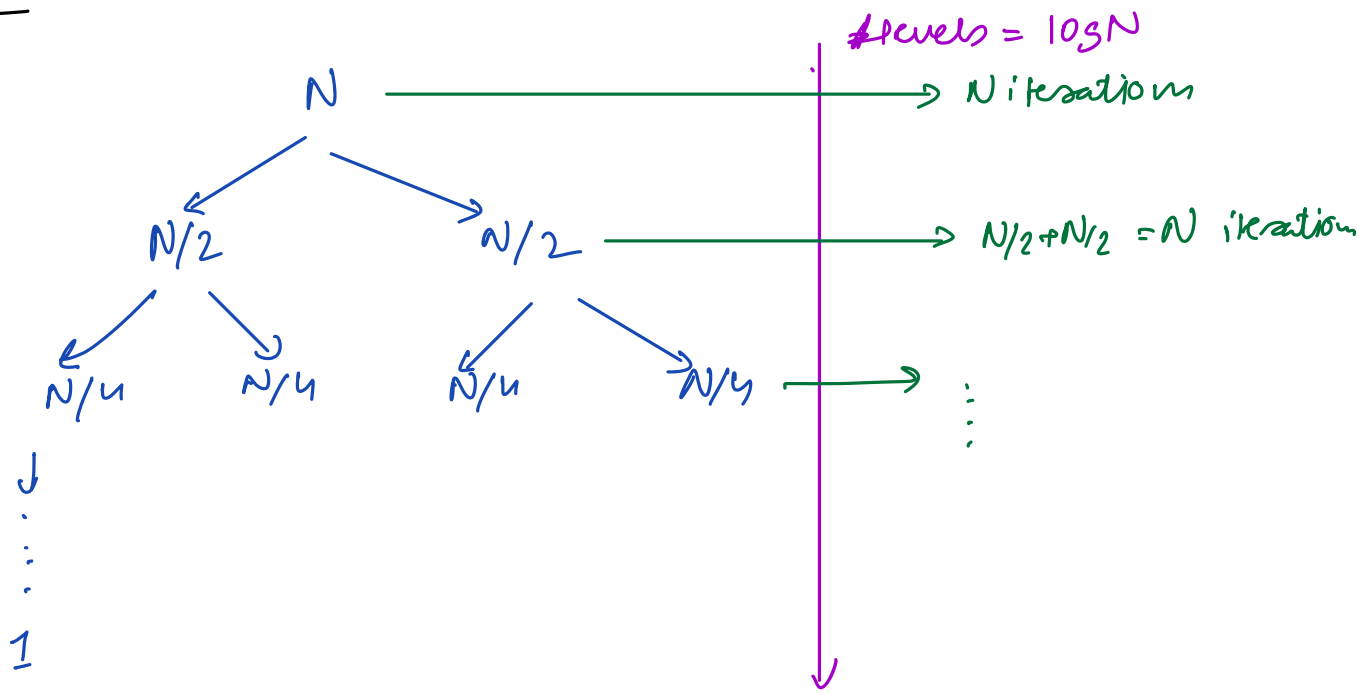
```
void quicksort(A[], l, r) {
                  0   n-1
    if(l >= r)  return

    pi = partition(A, l, r)
    quicksort(A, l, pi-1)
    quicksort(A, pi+1, r)

}
```

Time Complexity

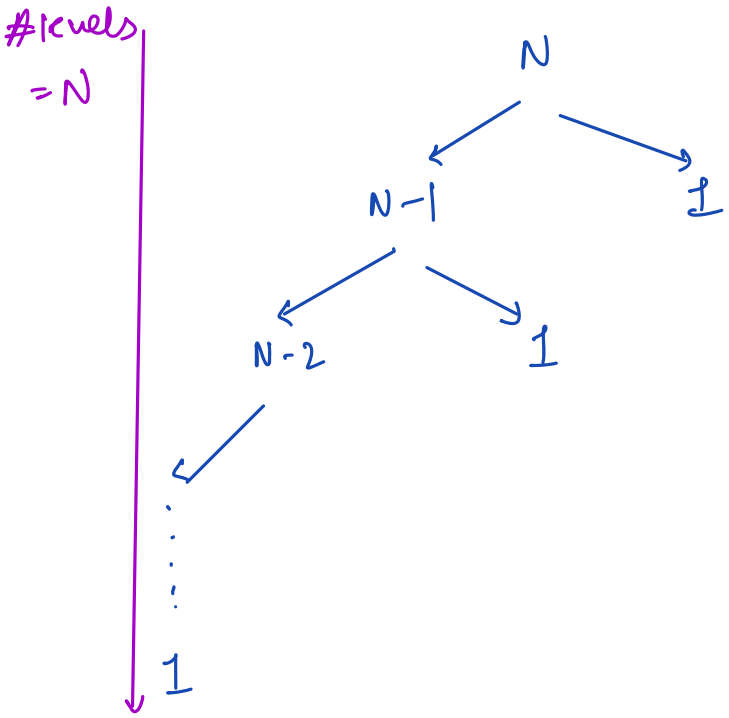
Best Case



$TC = O(N \log N)$

$SC = O(\log N)$

Worst Case



$TC = O(N^2)$

$SC = O(N)$

1 2 3 4 5

1 2 3 4 5

1 2 3 4

1 2 3

1 2

Randomized QuickSort

How to choose random ?

1. Get a random integer from [l, r] using "random" library.
2. Swap a[r] with a[random_index]
3. Run partition() with a[r] as pivot

```
int partition (A, l, r) {
    int random_index = randInt( l, r)   // random value from [l, r]
    swap ( a[r], a[random_index])
    pivot = A[r]   // rightmost
    .
    .
    .
    .
    .
}
```

$l \to 0$, $r \to n-1$

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

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

$\vdots$

$1/N-2$

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

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

Hence, on averge TC of quicksort $= O(N \log N)$

Comparator

compare 2 values & return a result indicating whether the values are equal, less than or greater than.

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 $\Rightarrow$ return (first - second)

sort in descending $\Rightarrow$ return (second - first)

C++

```
bool compare ( first , second ) {
    return → true  ⇒ first should come befor second
           → false ⇒ first should come after second
}
```

TC of sort = $O(N \log N) \times$ (TC of custom comparator)

Question

Given an array, sort the data w.r.t count of factors. If factor count is same, sort based on magnitude.

A = [	9	3	10	16	4]
# factors	3	2	4	5	3
sorted →	3	4	9	10	16

A = [	10	4	5	13	1]
# factors:	4	3	2	2	1
sorted →	1	5	13	4	10

code (C++)

```
vector<int> solve( vector<int> A) {
    sort( A.begin(), A.end() , compare)
    return A
}

bool compare ( int x , int y )
    int fx = factors(x)  -> TODO
    int fy = factors(y)  -> TODO

    if ( fx != fy ) {
        return (fx < fy);
    }
    return x < y
}
```

if fx < fy => x before y
else => x after y

(Java)

```
ArrayList<Integer> solve( ArrayList<Integer> A ) {
    Collections.sort( A , new Comparator<Integer>() {
        @ override
```

```
public int cmp (Integer x, Integer y){
        fx = factors(x)
        fy = factors(y)
        if ( fx != fy ){
            return fx - fy
        }
        return x - y
    }
});
return A;
}
```

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"

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

ans = "9534330"

$A = [10, 5, 2, 8, 200]$

ans = "8522001 0"

<u>let's use custom sorting</u>

x, y ⇒ create string "xy"

&

"yx"

x = 542 xy = "54260"

y = 60 yx = "60542" ✓

⇒ x should come after y

C++

```
string largestNum( vector<int> A){
    sort(A.begin(), A.end(), compare);
    string ans = "";
    for( auto x : A) {
        ans += to_string(x)
    }
    if (ans[0] == '0')  return "0"
    return ans
}

bool compare ( int x , int y ) {
    string xy = to_string(x) + to_string(y)
    string yx = to_string(y) + to_string(x)

    return (xy > yx)
}
```

Java

```
String largestNum( ArrayList<Integer> A) {
    Collections.sort(A, new Comparator<Integer>() {
        public int cmp( Integer x, Integer y ) {
            String xy = String.valueOf(x) + String.valueOf(y)
            String yx = String.valueOf(y) + String.valueOf(x)

            if ( xy.compareTo(yx) > 0) {
                return -1
            }
            else {
                return 1
            }
        });
    StringBuilder ans = new StringBuilder();
    for (int x : A) {
        ans.append( String.valueOf(x))
    }
    if( ans.charAt(0) == '0')
        return "0"
    return ans.toString();
```

3

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

$\downarrow$ sorting

$[9, 5, 34, 3, 30]$