Sorting 2

Good Evening :)

QuotePixel.com

Agenda

01. Partition of Array
02. Quicksort
03. Comparators
    - → Solve questions on comparators
    - 01. Sort no. based on parameter
    - 02. B closest point to Origin
    - 03. Largest Number.

Partition of Arr

Q1 Given an integer array, consider first element as pivot, rearrange the elements such that

$\forall i$, $A[i] < p \rightarrow$ move left

$A[i] > p \rightarrow$ move right

(Distinct ele)

A[] = { 54 26 93 17 77 31 44 55 20 }
(0 1 2 3 4 5 6 7 8)

Ans[] = { 26 17 31 44 20 54 93 77 55 }

A[] = { 10 13 7 8 25 20 23 5 }
(0 1 2 3 4 5 6 7) — i = 0, j = 7

Ans[]: { 7 8 5 10 13 25 20 23 }

A[] = { 54 26 ~~93~~ 17 ~~77~~ 31 ~~44~~ ~~55~~ ~~20~~ }
(0 1 2 3 4 5 6 7 8) — R = 5, L = 6

P, 20, ~~55~~ 44, 55, 77, 93

Q1 if (A[L] < P) L++;

if (A[L] > P) {

swap(A, L, R)

R--;

}

A[ ] = { ~~54~~ 26 20 17 44 ~~31~~ 55 77 93 }

54

31 R L

02 ⇒ swap (pivot & R) at the end

```
pivot = A[0]      // A[L]
int i = 1         // L+1
int j = n-1       // R
while (i ≤ j){
    if (A[i] < p) i++;
    else if (A[i] > p){
        swap(A, i, j)
        j--;
    }
}
swap(A, 0, j)     // putting pivot ele at its correct position
```

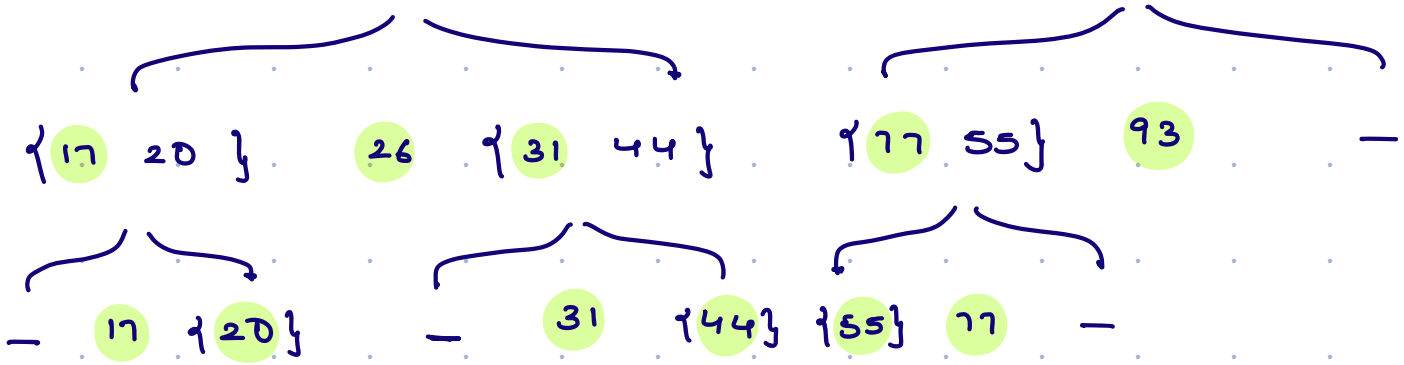
```
swap(A, int i, int j)
{
    int t = A[i]
    A[i] = A[j]
    A[j] = t;
}
```

Quick sort → Divide & Conquer Technique

→ Take a pivot element & we place it at its correct position

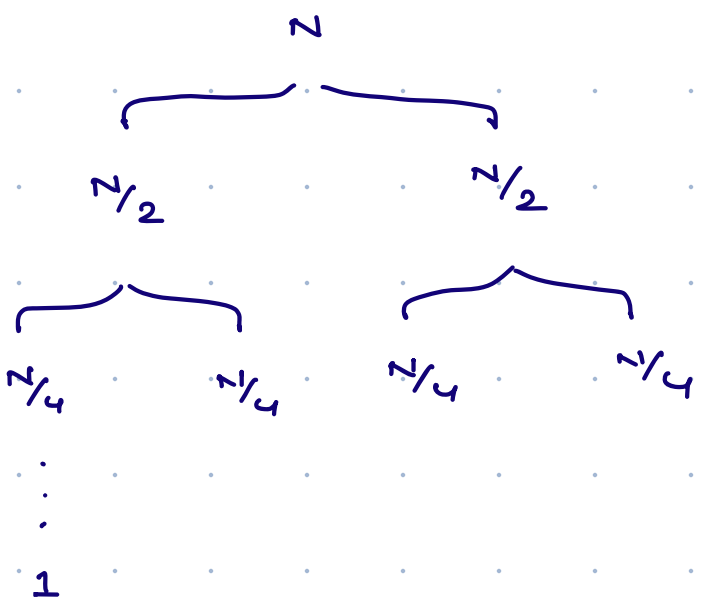
	0	1	2	3	4	5	6	7	8
A[ ] = {	54	26	93	17	77	31	44	55	20 }

{ 26 17 31 44 20 } 54 { 93 77 55 }




```
                                  0          n-1
                                  ↑          ↑
void quicksort (A, L, R)
|
|   if (L ≥ R) return;
|
|   pi = partition (A, L, R)
|
|   quicksort (A, L, pi-1)
|
|   quicksort (A, pi+1, R)
|
}
```

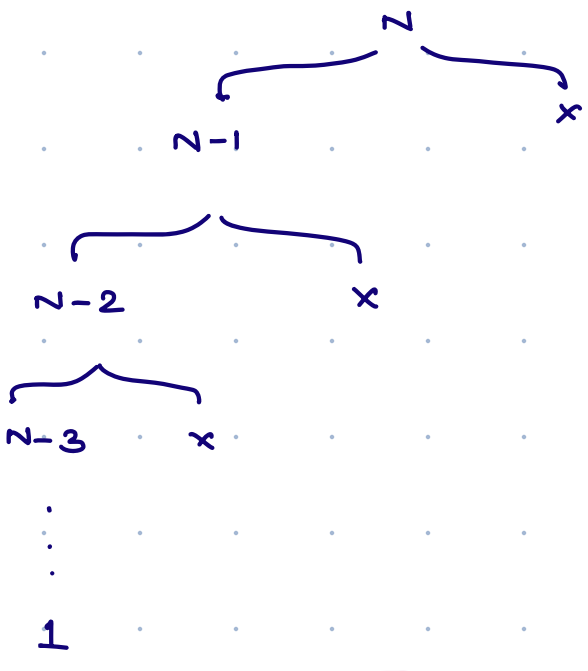
Best Case



TC : O(nlogn)

SC : O(logn)

Worst case



TC : O(n²)

SC : O(n)

	TC	SC	Stability
Merge Sort	O(N log N)	O(n)	Stable
Quick sort	O(N log N)	O(log N)	Unstable

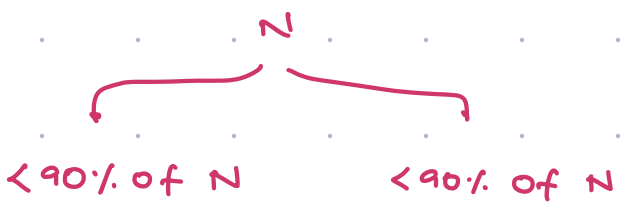
* **Randomised Quick sort**

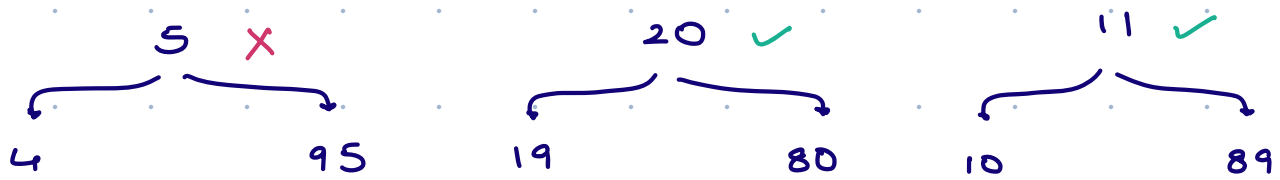
→ We pick the pivot randomly & not at start or at the end.

Random pivot

1 2 3 4 . . . . . 98 99 100

Probability of selecting a pivot such that

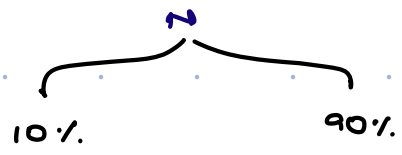


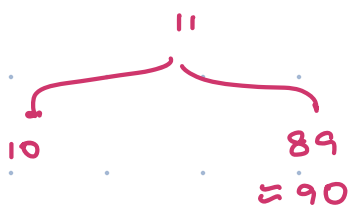


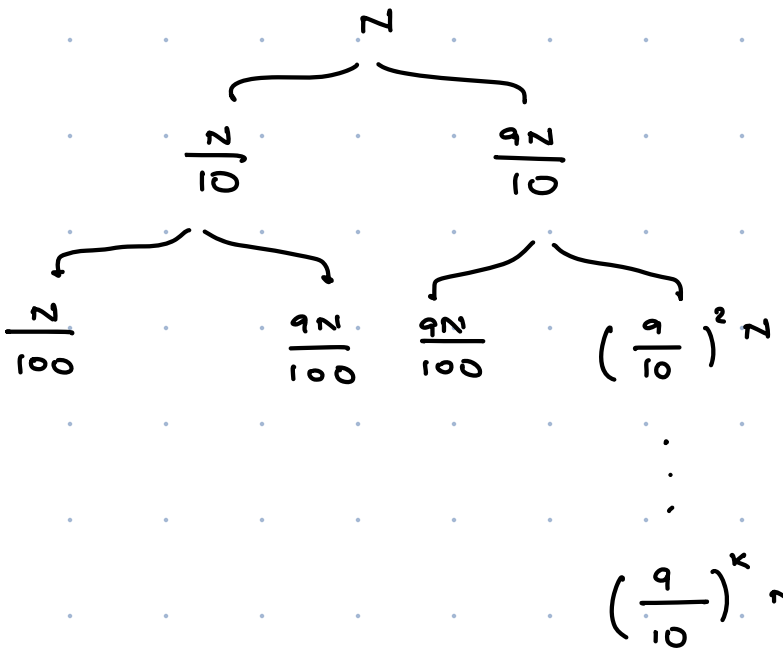
valid pivots = [11 90] ⇒ 90 − 11 + 1 = 80

probability = $\frac{80}{100}$ = 80%

1. Worst case







levels

$\left(\frac{9}{10}\right)^k N = 1$

$N = \left(\frac{10}{9}\right)^k$

$k = \log_{\frac{10}{9}} N$

$N = 10^5$

$k = \log_{10/9}(N) = 10^2$

TC : $O(N \log_{\frac{10}{9}} N) \leq 10^7$

SC : $O(\log_{\frac{10}{9}} N) \leq 10^2$

(better SC than merge sort)

10:19 PM → 10:29 PM

* **Comparator** → Interface which sorts the data in desired order

A[] = { 10 23 40 45 7 }

Arrays.sort(A) → { 7 10 23 40 45 }

→ Ascending order

→ Comparator takes two argument & returns

01. -ve value
02 +ve value
03. Neutral value

} Java, python, C#, Ruby, JS

```
int compare (Integer a, Integer b){
    01. If we want a to be present in beginning = -ve
    02 If we want a to move on right of B = +ve
    03. If (a == b) return 0;
}
```

C++

01. If we want a before b → return true
02 If we want a on right side of b → return false.

A[] = {10 3 17 48 20}

Arrays.sort(A) → {3 10 17 20 48}

Q - I want the descending order in A

Arrays.sort(A, new desc()); → {48 20 17 10 3}

```
public class desc implement Comparator<Integer>{
    public int compare(Integer a, Integer b)
    {
        if (a > b) return -1;
        else if (a < b) return 1;
        else return 0;
    }
}
```

* Sort the data based on ascending order of count of factors. If count of factors, then sort it on the basis of values.

A[ ] = { 9 3 10 6 4 }

cf = 3 2 4 4 3

Ans[ ] = { 3 4 9 6 10 }

A[ ] = { 10 4 5 13 1 }

cf = 4 3 2 2 1

Ans[ ] = { 1 5 13 4 10 }

```
int factors (int n)
{
    int c=0
    for(i=1; i*i<=n; i++){
        if (n%i==0){
            if (i==n/i) c++;
            else c+=2;
        }
    }
    return c;
}
```

Arrays.sort(A, new cf());

```
public class cf implements Comparator<Integer>{

    public int compare (Integer a, Integer b){

        int fa = factors(a)
        int fb = factors(b)

        if (fa < fb) return -1;
        else if (fa > fb) return 1;
        else {
            if (a<b) return -1;
            else if (a>b) return 1;
            else return 0;
        }
    }
}
```

Some constant factor ↑

TC : O(nlogn * $\sqrt{k}$)

SC : based on sorting Algo

* C++

```
if ( fa < fb || (fa == fb && a<b)) return true:
else return false;
```

Zomato's Issue :

You are developing a feature for **Zomato** that helps users find the **nearest restaurants** to their current location. It uses GPS to determine the user's location and has access to a database of restaurants, each with its own set of coordinates in a two-dimensional space representing their geographical location on a map. The goal is to identify the **"B" closest restaurants** to the user, providing a quick and convenient way to choose where to eat.

Given a list of restaurant locations (each represented by its x and y coordinates on a map) and the user's current location (also in x and y coordinates), write a function to find the "B" closest restaurants to the user's location. The distance between the user's location and a restaurant is calculated using the Euclidean distance formula.

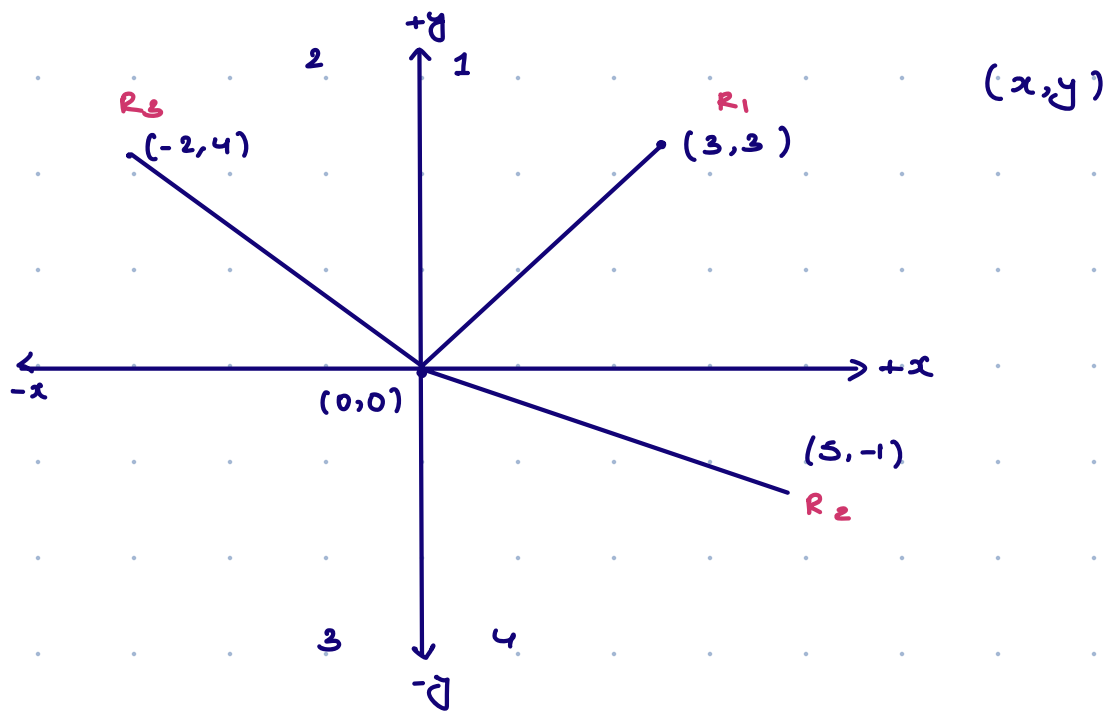
$$\text{Distance} = \sqrt{(x_1-x_2)^2 + (y_1-y_2)^2}$$

$$= \sqrt{(x_1)^2 + (y_1)^2}$$

R_1	3	3	$\sqrt{18}$
R_2	5	-1	$\sqrt{26}$
R_3	-2	4	$\sqrt{20}$

B = 2

B closest restaurants = R_1 & R_3



Idea → Sort the array on the basis of distance from origin & return first B closest restaurants

```
Arrays.sort(A, new dist());
```

```
public class dist implements Comparator<int[]>{
    public int compare(int[] a, int[] b){
        x1 = a[0]        x2 = b[0]
        y1 = a[1]        y2 = b[1]

        int da = x1^2 + y1^2

        int db = x2^2 + y2^2

        if (da < db) return -1;
        else if (da > db) return 1
        else return 0;
    }
}
```

```
for (i=0; i<B; i++){
    System.out.print(A[i][0] + "  " + A[i][1]);
}
```

03 Given an integer A with numbers. Arrange the array such that it forms largest no. when concatenated & return it as a string

A[] = { 10 2 55 100 }

↳ " 55 2 10100 "

A[] = {10 5 2 8 200}

↳ "8 5 2 200 10"

A[] = {53 402}

→ 53402

→ 40253

A[] = {3 3 34 5 9}

Try this → A[] = {90 909 900 9}

I'll discuss

```cpp
// Define a structure to represent an interval
struct Interval {
    int start;
    int end;
};

// Comparator function to sort intervals based on the ending time
bool compareInterval(const Interval &i1, const Interval &i2) {
    return (i1.end < i2.end);
}

int main() {
    // Create a vector of intervals
    std::vector<Interval> intervals = {{1, 3}, {2, 4}, {3, 5}, {0, 6}, {5, 7}, {8,

    // Sort the intervals based on the ending time
    std::sort(intervals.begin(), intervals.end(), compareInterval);

    // Print the sorted intervals
    std::cout << "Intervals sorted by ending time:\n";
    for (const auto &interval : intervals) {
        std::cout << "[" << interval.start << ", " << interval.end << "] ";
    }
```