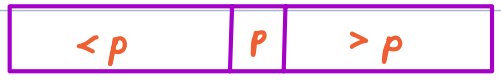
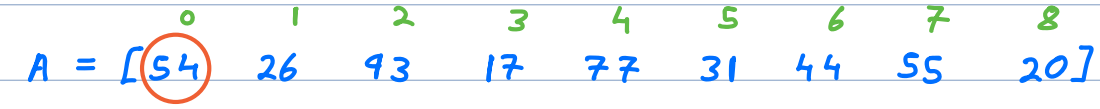
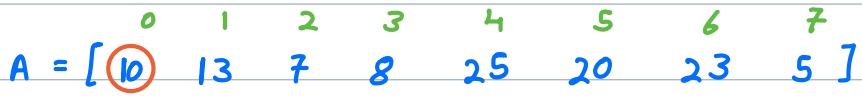
Q → Given an integer array, consider first element as pivot, rearrange the elements s.t

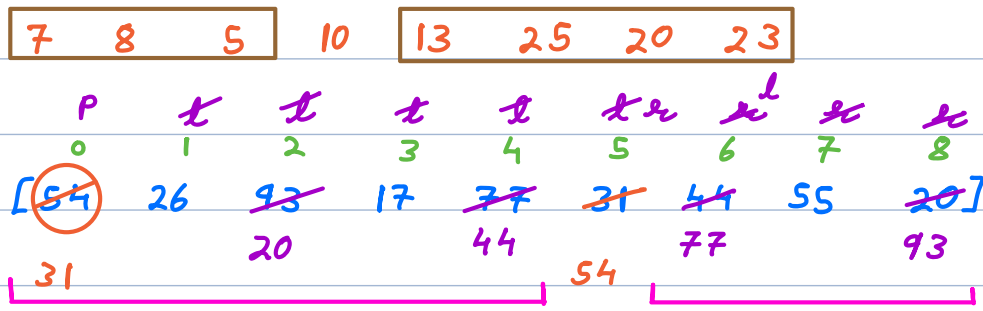
$\forall i$, $A[i] < p \rightarrow$ move left

$A[i] > p \rightarrow$ move right (distinct elements)










```
p = A[0]      // L
l = 1         // L+1
r = N-1       // R
while (l <= r) {
    if (A[l] < p)   l++
    else if (A[r] > p)  r--
    else {  swap(A, l, r)
            l++   r--
    }
```

compare via comparator

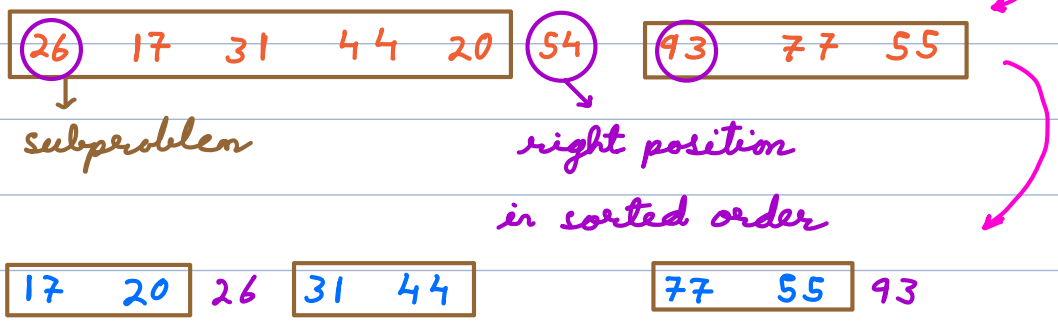
```
    }
}
```

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

swap (A, r, 0) // pi = r

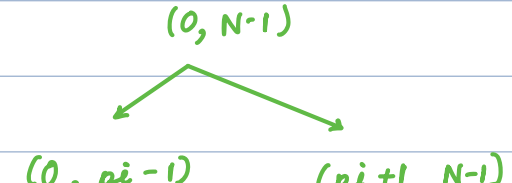
Quick Sort (Divide & Conquer)

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



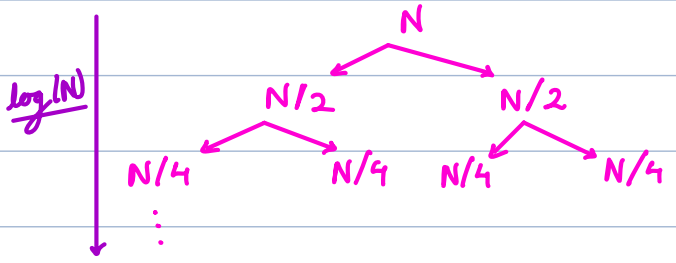

```
void quickSort (A, L, R) {          // L: 0, R: N-1
    if (L >= R) return
    pi = partition (A, L, R)
    quickSort (A, L, pi - 1)
    quickSort (A, pi + 1, R)
}
```

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



Time Complexity

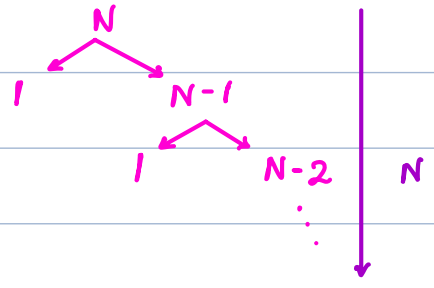
Best Case



TC = O(N log (N))

SC = O(log N)

Worst Case



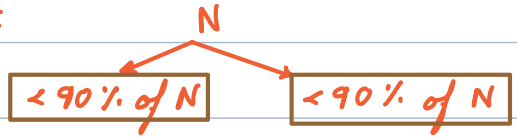
TC = O(N^2)

SC = O(N)

Random Pivot

1 2 3 . . . 9 10 11 . . . . 98 99 100

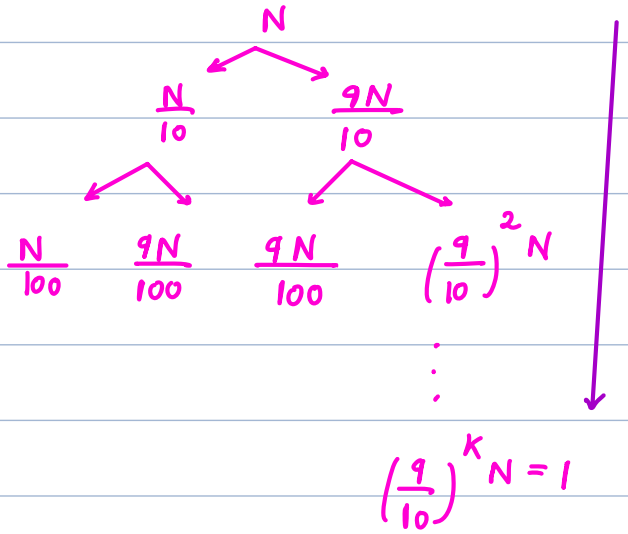
Probability of selecting pivot s.t



valid numbers → [11 90] → 90 - 11 + 1 = 80

⇒ probability = $\frac{80}{100}$ = 0.8 (80 % cases)

Worst scenario with 80 % probability



levels →

$(9/10)^K N = 1 \Rightarrow N = (10/9)^K$

$\Rightarrow K = \log_{10/9}(N)$

$N = 10^5 \Rightarrow \log_{10/9}(N) \approx 10^2$

TC = $O(N \log_{10/9}(N))$ → 10^7

SC = $O(\log_{10/9}(N))$ → 10^2 (better SC wrt Merge sort)

Comparators

It defines the parameter wrt which we organise data.

```
int compare (object u, object v) {
    if 'u' should be on left wrt v → return -ve
    if 'u' should be on right wrt v → return +ve
    if 'u' & 'v' are same → return 0
}
```

Q → Sort the given integer array wrt #factors in ascending order. If count of factors are same use their actual values to compare.

A = [9 3 10 6 4]

#factors → 3 2 4 4 3

o/p → [3 4 9 6 10]

A = [10 4 5 13 1]

#factors → 4 3 2 2 1

o/p → [1 5 13 4 10]

```
int compare (Integer u, Integer v) {
    uf = factors(u)    vf = factors(v)
    if (uf < vf) return -1

    else if (uf > vf) return 1
    else {
```

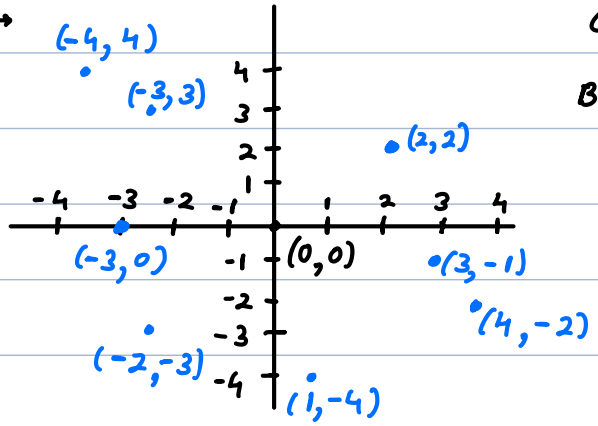
```
        if (u < v) return -1
        else if (u > v) return 1
        else return 0
    }
}
```

return u - v

H.W → check comparator syntax in your programming language.

Q → Given array of points find B closest points to origin.

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



Eg → B = 3

o/p → (2, 2) (-3, 0) (3, 1)

-4, 4 → $\sqrt{32}$ 2, 2 → $\sqrt{8}$ ✓

-3, 3 → $\sqrt{18}$ 3, 1 → $\sqrt{10}$ ✓

-3, 0 → $\sqrt{9}$ ✓ 4, -2 → $\sqrt{20}$

-2, -3 → $\sqrt{13}$ 1, -4 → $\sqrt{17}$

$x < y$

$\Rightarrow \sqrt{x} < \sqrt{y}$

```
int compare (Point u, Point v) {
    sdu = u.x * u.x + u.y * u.y
    sdv = v.x * v.x + v.y * v.y
    return sdu - sdv
}
```

Q → ... an integer array with +ve numbers. Arrange the array s.t it forms largest number & return it as a string.

A = [10 2 55 100]

↳ "55 2 10 100" (Ans)

(large)
MSD ——— LSD

A = [10 5 2 8 200]

↳ "8 5 2 200 10" (Ans)

53 > 402 → 53 40 2

53 < 532 → 53253 ✗ ⇒ (string comparison)

53532 ✓

53 < 536 → 536 53

```
// int → Strings
int compare (String u, String v) {
    x = u + v   // concatenation
    y = v + u
    if (x < y) return 1
    else if (x > y) return -1
    else return 0
}
```

---

u = 53 (left of 534)

v = 534

x = u+v = 53534 ✓

y = v+u = 53453