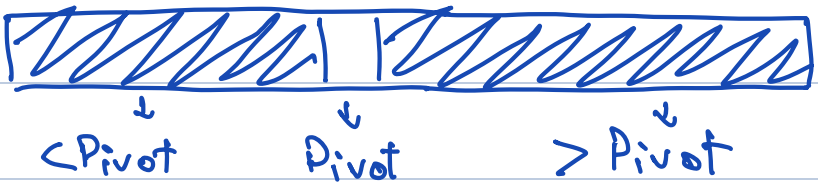
Sorting - 2

Today's Content:

- Pivot Partition
- Quick Sort
- Comparator Problems

Question: Given an integer array, consider the first element as pivot, rearrange the elements such that the elements < pivot are to its left side and elements > pivot are to its right side and return the index of the pivot.



ex: arr = [54, 26, 93, 17, 77, 31, 44, 55, 20]

Pivot

Pivot

0	1	2	3	4	5	6	7	8

O/P = [26, 17, 31, 44, 20, 54, 55, 77, 93]

return = 5

Approach

p c r

arr = [~~10~~, ~~15~~, ~~7~~, ~~8~~, ~~25~~, ~~20~~, ~~23~~, ~~8~~]

5 ~~8~~ ~~10~~ 10 ~~23~~ 23 25 13

~~10~~ 8 20

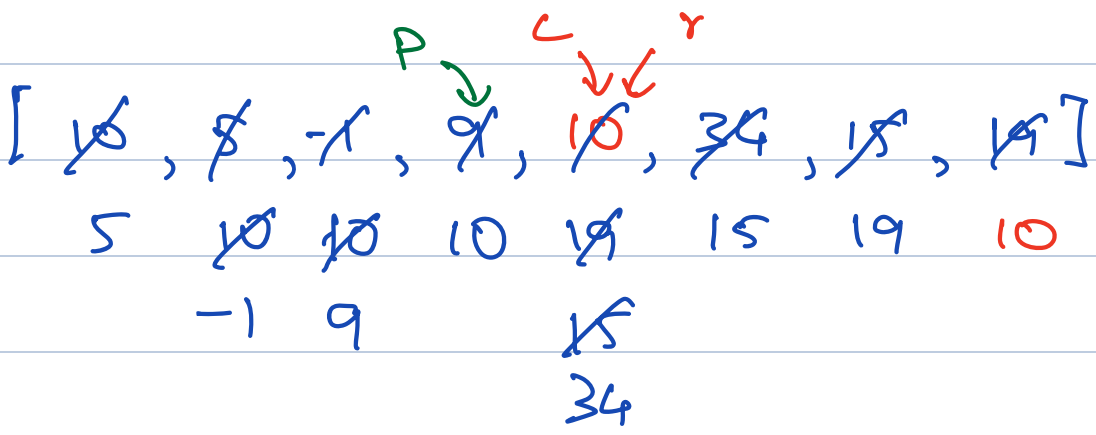
7

if arr[cur] < arr[pivot] : swap(c, p); c++; p++;

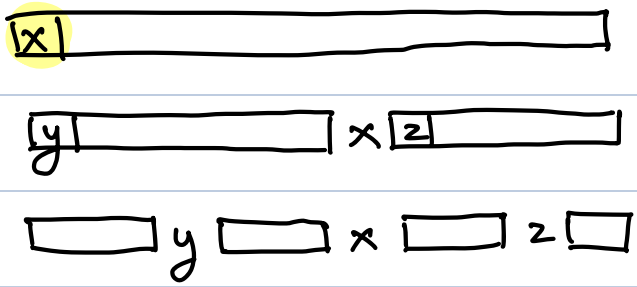
else : swap(c, r); r--;

```
func pivotPartition (int[] arr, int start, int end){
    p = start; c = start+1; r = end;
    while (c <= r) {
        if (arr[c] < arr[p]) {
            swap (arr[c], arr[p]);
            c++;
            p++;
        }
        else {
            swap(arr[c], arr[r]);
            r--;
        }
    }
    return p;
}
```

T.C = O(N) S.C = O(1)



Quick Sort



ex: [18 8 6 3 11 14 23 20 31 27]

8 6 3 11 14 18 23 20 31 27

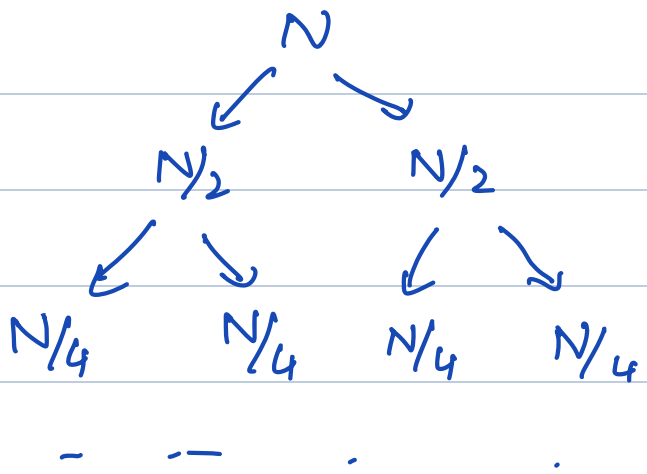
6 3 8 11 14 18 20 23 31 27

3 6 8 11 14 18 20 23 27 31

```
func quickSort(int[] arr, int start, int end){
    if(start >= end) return;
    pivot = pivotPartition(arr, start, end);
    quickSort(arr, start, pivot-1);
    quickSort(arr, pivot+1, end);
}
```

Complexity

1) Best case scenario:



T.C = levels * (TC of each recursion)

= $\log N * N = O(N \log N)$

S.C = levels + S.C of each recursion

= $\log N + 1 = O(\log N)$

2) Worst case scenario:

arr = [1, 2, 3, 4, 5] - N

↓

[1, 2, 3, 4, 5] - N

↓

[1, 2, 3, 4, 5] - N

↓

[1, 2, 3, 4, 5] - N ⇒ [2, 1, 4, 3, 5]

↓

[1, 2, 3, 4, 5] - N [1, 2, 4, 3, 5]

T.C = $O(N^2)$

S.C = $O(N)$

p — random p

Randomised Quick Sort

- Take random element as pivot.

```
func pivotPartition (int[] arr, int start, int end){
    p = random(start, end);
    swap (arr[start], arr[p]);
    p=start; c = start+1; r = end;
    while (c < r) {
        .....
```

Why randomised pivot?

- Chances of worst case drops to 1/N.
- Ammortised T.C of Quick Sort = $O(N \log N)$.

Comparator Problems

Comparator → Interface to help with custom sort.

```
class className implements Comparator {
    int compare (int i1, int i2){
        ↳ -1 → if i1 should be put first
        ↳ +1 → if i2 should be put first.
```

Question: Given an array of integers, sort the data in ascending order of count of factors. If count of factors are equal, then sort on the basis of their magnitude

ex: [9, 3, 10, 6, 4] ⇒ [3, 4, 9, 6, 10]
↓ ↓ ↓ ↓ ↓
3 2 4 4 3
↳ O/P.

[1, 5, 13, 4, 10]
↓ ↓ ↓ ↓ ↓
1 2 2 3 4

```
class compareFactors implements Comparator {
    @Override
    public int compare(int i1, int i2) {
        f1 = factors(i1);   // TODO
        f2 = factors(i2);   // TODO
        if (f1 == f2) {
            if (i1 <= i2) return -1;
            else return 1;
        }
        else if (f1 < f2) return -1;
        else return 1;
    }
}

main() {
    arr = ...;
    compareFactors c1 = new compareFactors();
    Collections.sort(arr, c1);
}
```

Python:

```
def compare(v1, v2):
    if (factors(v1)==factors(v2)):
        if (v1 <= v2) return -1
        else return 1
    elif (factors(v1) < factors(v2)): return -1
    else return 1

A = sorted(A, key=functools.cmp_to_key(compare))
```

Question: Given an array of points where p[i] = [xi, yi] representing a point on the X-Y plane and an integer k, return k closest points to the origin (0, 0).

Distance b/w 2 points $= \sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$

on X-Y plane $= \sqrt{x^2+y^2}$

ex: p = [[1, 3], [-2, 2]] k = 1

$\sqrt{10}$ $\sqrt{8}$

ex: p = [[3, 3], [5, -1], [-2, 4]] k = 2

$\sqrt{18}$ $\sqrt{26}$ $\sqrt{20}$

```
func compare (int[] v1, int[] v2){
    d1 = ( v1[0] * v1[0] + v1[1] * v1[1])
    d2 = ( v2[0] * v2[0] + v2[1] * v2[1])
    if (d1 <= d2) return -1;
    else return 1;
}

Collection.sort( points, compare)
return points[0: k-1] // Return 1st k points
```

Question: Given a list of non-negative integers, arrange them such that they form the largest number and return it.

ex: arr = [10, 2] → 102, 210

O/P = [2, 10]

ex: arr = [3, 30, 34, 5, 9]

O/P = [9, 5, 34, 3, 30]

arr = [10, 5, 2, 8, 200]

O/P = [8, 5, 2, 200, 10]

[8, 9]

[98, 789]

98789

78998

1) Convert to string → N + M
2) Concatenate → N + M
3) Convert to int → N + M
4) Compare → 1

1) Convert to string - N+M

2) Compare char[0] ... char[min(N, M) - 1]

↳ min(N, M)

```
func compare(int v1, int v2){
    s1 = String.toString(v1);
    s2 = String.toString(v2);
    for(int i=0; i < min(s1.length, s2.length); i++){
        c1 = s1.CharAt(i);
        c2 = s2.CharAt(i);
        if(c1 > c2) return -1;
        else if(c1 < c2) return 1;
    }
    //Keep checking last digit in min(N, M) with
      remaining digits of max(N, M).
}
```

999 968 999 999 309 3

3093

↓ ↑

399 , 3 339 393

v1 v2

- Contest on 26th July 2024 → 9 PM to 10:30
- Contest Discussion from 10:30 PM – 12 AM.
- Topics:
  1) Hashing
  2) Sorting
  3) Searching.