16/17-dec-2021

Thursday, 16 December 2021 8:25 PM

Merge Sort :-

https://leetcode.com/problems/sort-an-array/

two merge Sorted array :-

[1, 3, 5, 7] + [2, 4, 6, 8] = [1, 2, 3, 4, 5, 6, 7, 8]

A B C

A = [1, 3, 5, 7]

i

B = [2, 4, 6, 8]

j

i = 1 2 3 4

j = 0 1 2 3

k = 1 2 3 4 5 6 7

C = [1, 2, 3, 4, 5, 6, 7, 8, 10, 12]

```
if (a[i] < b[j]) {
    c[k] = a[i];
    i++;
    k++;
} else {
    c[k] = b[j];
    j++
    k++;
}
```

a = [1, 3, 5, 7] ai = 0 1 2 3 4

b = [2, 4, 6, 8] bi = 0 1 2 3 4

k = 0 1 2 3 4 5 6 7 8

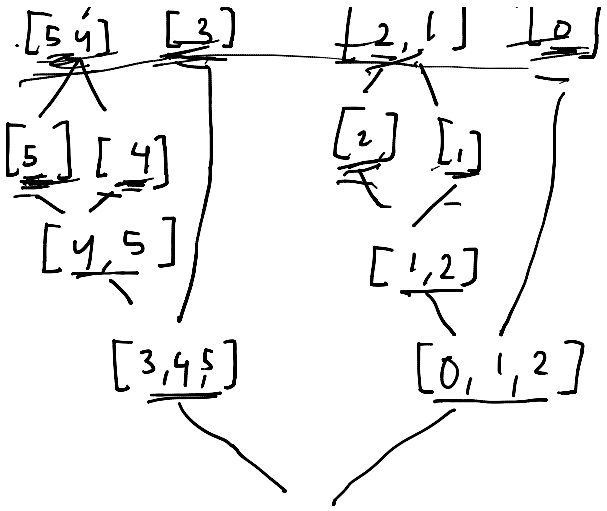
C = [1, 2, 3, 4, 5, 6, 7, 8]

```
while (ai < a.length && bi < b.length) {
    if (a[ai] < b[bi]) {
        c[k] = a[ai];
        ai++;
        k++;
    } else {
        c[k] = b[bi];
        bi++;
        k++;
    }
}
while (ai < a.length) {
    c[k] = a[ai];
    ai++;
    k++;
}
while (bi < b.length) {
    c[k] = b[bi];
    bi++;
    k++;
}
return c;
```

[5, 4, 3, 2, 1, 0]

[5, 4, 3] [2, 1, 0]

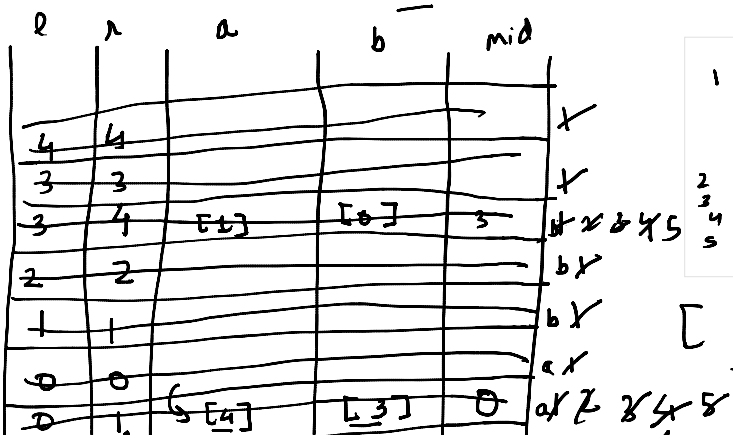
[5, 4] [3] [2, 1] [0]



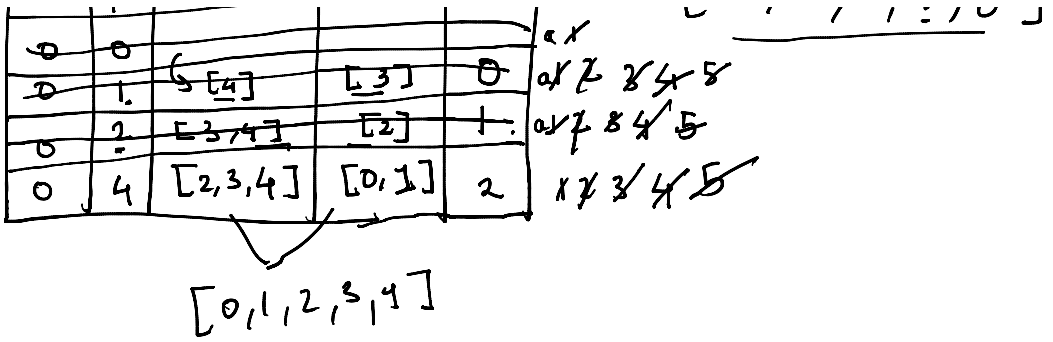

```
int[] mergeSort(arr, l, r){
  if(l==r) return new int[]{arr[l]};
  mid = (l+r)/2;
  int[]a = mergeSort(arr, l, mid);
  int[]b = mergeSort(arr, mid+1, r);
} return mergetwoSortedArr(a, b)
```

5 = 2
4 = 2

mid



[4, 3, 2, 1, 0]



Quick Sort :-

recursion Partion Index 1, 5 4, 6, 2, 3

Partition Index :-

[7, 3, 12, 16, 59]

Pivot = 16

i = 5

j = 4

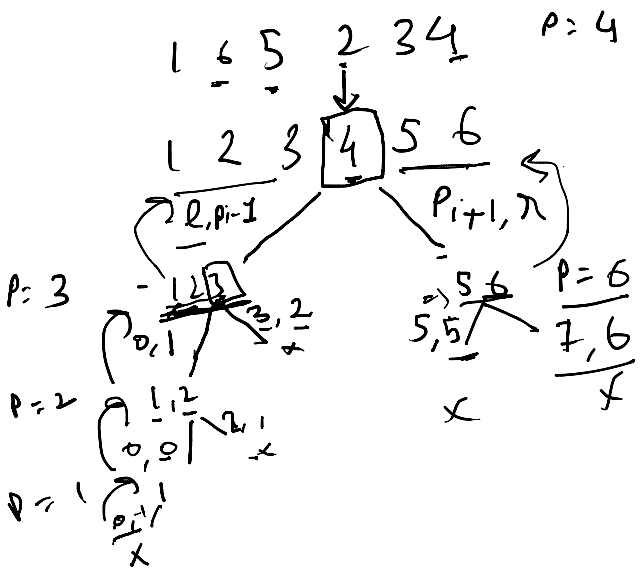
if (arr[i] <= pivot) {
swap(i,j); j++;
i++;
}

else {
i++;
}

quickSort :- (Take the last element as pivot)

qS(arr, l, r)

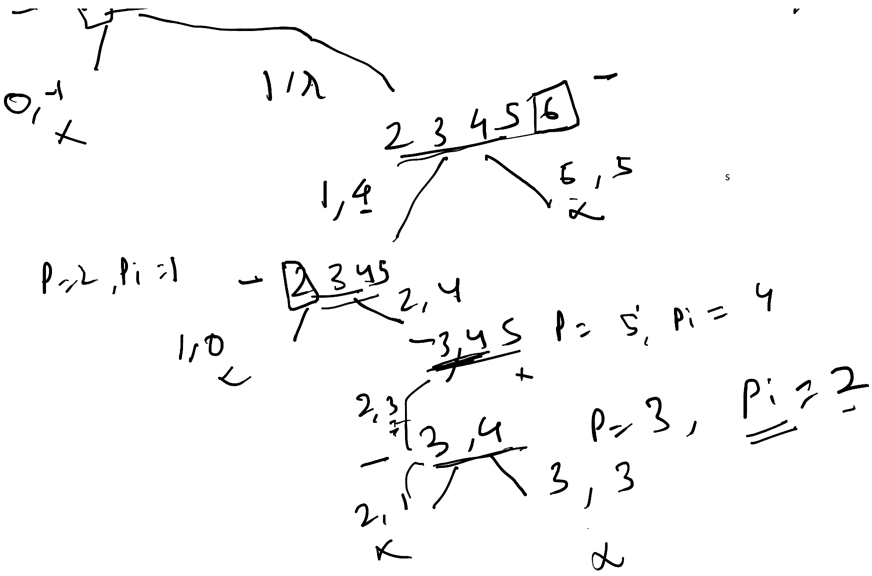
if (l >= r) return



6 5 4 3 2 1

P = 1 / Pi = 0

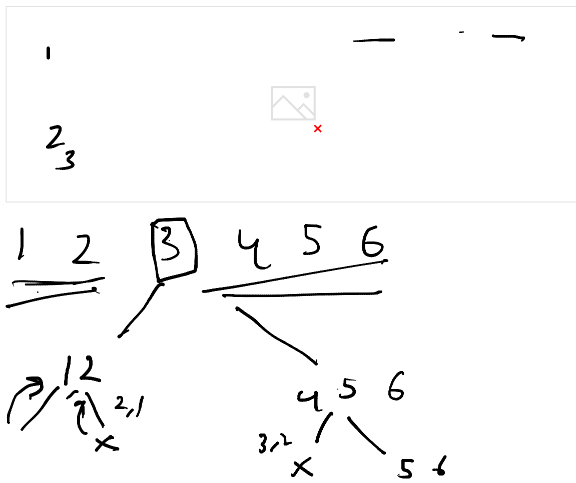
P = 6 / Pi = 5




```
void quickSort(arr, l, r) {
    if (l >= r) return;
    pivot = arr[r];
    pi = partitionIndex(arr, pivot, l, r);

    quickSort(arr, l, pi-1);
    quickSort(arr, pi+1, r);
}
```

l	r	Pivot	pi	
6	5			1 ✓
4	4			1 ✓
4	5	6	5	1 2 3
3	2			1 ✓
3	5	4	3	1 2 3
2	1			1 ✓
0	0			1 ✓
0	1	2	1	1 2 3
0	5	3	2	1 2 3



Time Complexity:

$n + n = 2n = O(n)$

O(n):

[10]

n = s.nextInt();

$n + n = 2n = O(n)$

for (0 to n-1) {

→ O(n)

2 × (n)

for (0 to n-1) { => O(n) 2x(n)
  sout("Hello")
}

5 times = $\frac{10}{2} = \frac{n}{2} = \frac{1}{2} * (n)$

for (i= ; i < n: i=i+2) { => $\frac{n}{2}$ i = 0, 2, 4, 6, 8
}

$\frac{n}{2}$ => $\frac{n}{2}$, n, logn, 2n

O(n) =

$\frac{n}{4} = \frac{n}{n} =$

$\frac{n}{n} = 1$

$O(n^2)$

for (i=0 to n-1) {
  for (j=0 to n-1) {
  }
}

n = 5

0 = 0 1 2 3 4
1 = 0 1 2 3 4
2 = 0 1 2 3 4
3 = 0 1 2 3 4
4 = 0 1 2 3 4

$= 25 = 5^2 = n^2$

for (i=0 to n-1) {
  for (j=i to n-1) {
  }
}

n = 5 $O(n^2)$

0 => 0 1 2 3 4 : 5
1 => 1 2 3 4 : 4
2 = 2 3 4 : 3
3 = 3 4 : 2
4 = 4 : 1
15

$O(n^2) = \frac{(n)(n+1)}{2}$

for (i=0 to n-1) {
  for (j=i to i) {
  }
}

n = 5

log n = O(n)

3 2

log (n)

for (i=2 ; i<n ; i=2*i) {
}

n, 2^{2n}, $\sqrt{n}$, logn

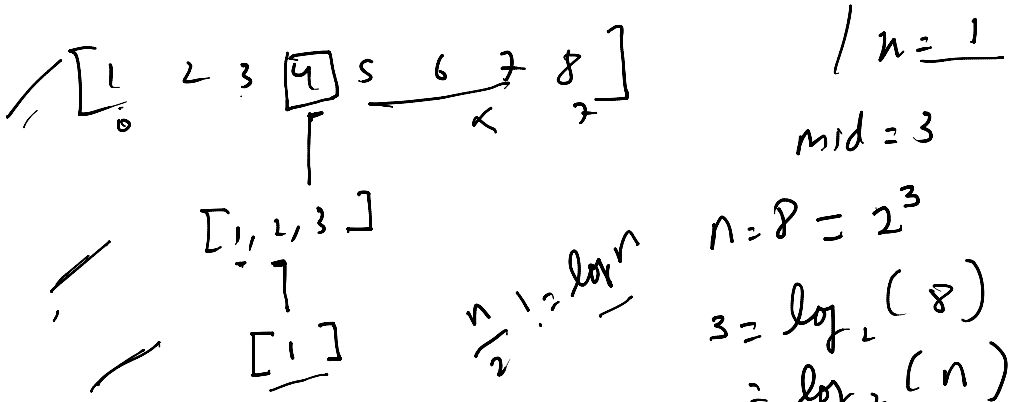
2, 4, 8, 16, 32 = 5 times

$32 = 2^5$

$\log_2(32) = 5$

$n=2 \quad \log_2(2) = 1$

$\log_2(2^k = n) = k$



$n = 1$

$mid = 3$

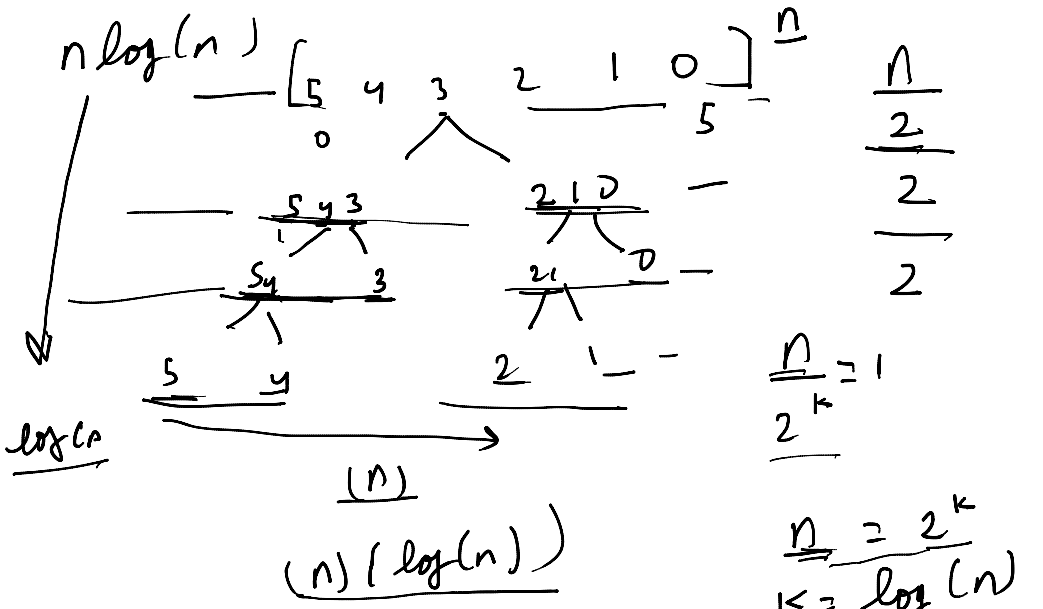
$\frac{n}{2} = \log n$

$n = 8 = 2^3$

$3 = \log_2(8)$

$= \log_2(n)$

$n \log(n)$



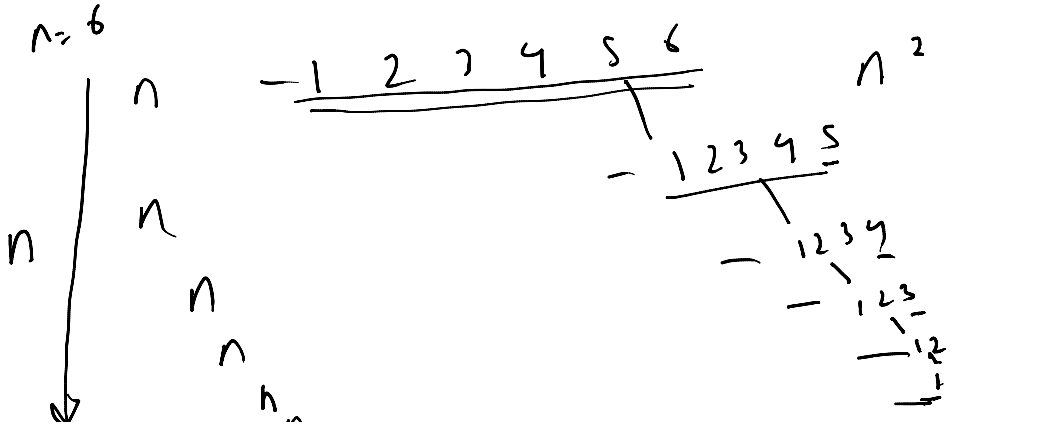
$n = 2^k$

$k = \log(n)$

Merge Sort = always its $(n)(\log(n))$

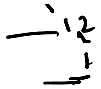
Quick Sort = $\Rightarrow n \log(n)$

$n = 6$



n

n_n



Best Case = n^2

Worst Case = n^2

Avg " = $n \log(n)$

$O(1)$