Agenda

1) Quick sort
2) Complexity analysis of quicksort and merge sort
3) 1 question on custom comparison (if time permits)

Q.1 Partition an array

Given an array, partition it based on last element (ele) such that all elements < ele are coming on left of it and all elements >= ele are coming on right of it.

$$A = \left[\begin{array}{cccccccc} 8 & 9 & 3 & 1 & 5 & 6 & 10 & \boxed{7} \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}\right] \qquad ele = 7$$

$$\underbrace{3 \quad 1 \quad 5 \quad 6}_{<7} \quad 7 \quad \underbrace{8 \quad 9 \quad 10}_{>=7}$$

$$A = \left[\begin{array}{cccccccc} 8 & 5 & 1 & 3 & 7 & 2 & 9 & \boxed{6} \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}\right] \qquad ele = 6$$

$$\underbrace{5 \quad 1 \quad 3 \quad 2}_{<6} \quad 6 \quad \underbrace{8 \quad 7 \quad 9}_{>=6}$$

Expected TC : O(n)

without creating space

ele = 7

i, j

i

A = [3 1 5 6 $\boxed{7}$ 9 10 8]

0 1 2 3 4 5 6 7

j

i to j-1 => >= ele

```
if ( A[j] >= ele ) {
        j++;
}
```

```
else {
        swap(A[i], A[j]);
        i++; j++;
}
```

swap (A[i], A[n-1]);

why
=

ele = 7

i, j

i

A = [3 1 5 6 $\boxed{7}$ 9 10 8]

0 1 2 3 4 5 6 7

j

i to j-1 => >= ele

A[i] >= ele

```
if ( A[j] >= ele ) {
        j++;
}
```

```
else {
        swap(A[i], A[j]);
        i++; j++;
}
```

swap (A[i], A[n-1]);

```
void   partition ( int[] A ) {

   int  n= A.length ;

   int  ele= A [n-1];

   int  i =0, j = 0 ;

   while ( j < n-1 ) {

       if (A[j] >= ele ) {

            j++;

       }
       else {

            // swap  A[i], A[j]

            int  temp= A[i];

            A [i]= A [j];

            A[j] = temp;

            i++;

            j++;

       }
   }

   // swap  A[n-1], A[i]

   int  temp= A [i];

   A [i] = A [n-1];

   A [n-1] = temp;

}
```

i

[2 5 3 1 $\boxed{6}$ 6 8 9 7]

0 1 2 3 4 5 6 7 8

j

ele = 6

i , j-1 => >= ele

the last comes at what index after partition : i

Quicksort

- ↳ divide and conquer sorting algo
- ↳ recursion

[8 5 1 3 7 2 9 6]
0 1 2 3 4 5 6 7

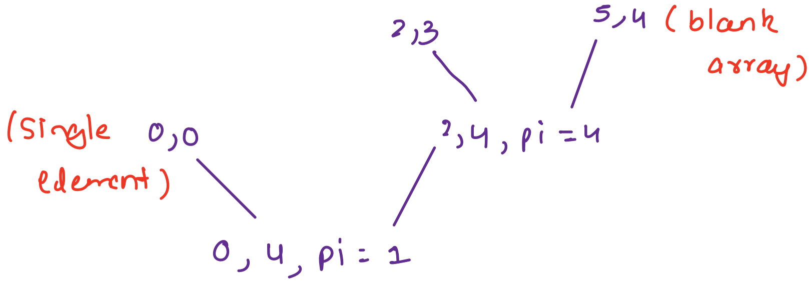
i) partition array based on pivot element => last element

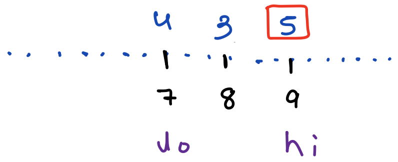
sort(lo, pi-1) sort(pi+1, hi)

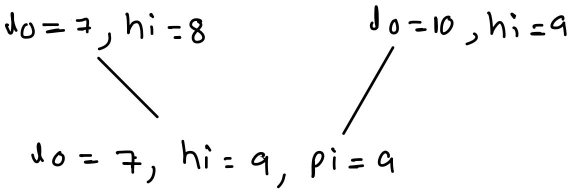
1 2 3 5 7 8 9
[~~5 1 3 2~~ $\boxed{6}$ ~~8 9 7~~]
0 1 2 3 4 5 6 7
lo pi hi

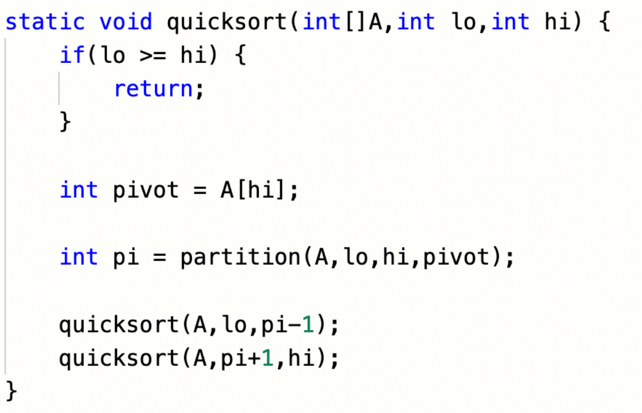
```
void quickSort ( int [ ] A , int lo , int hi) {
                                    if (lo >= hi) { return }

    int pivot = A[hi];

     int pi = partition( A, lo, hi, pivot);

     quicksort (A, lo, pi-1);

     quicksort (A, pi+1, hi);

}
```

A =	1	2	4	3	$\boxed{5}$
	0	1	2	3	4
			lo		hi

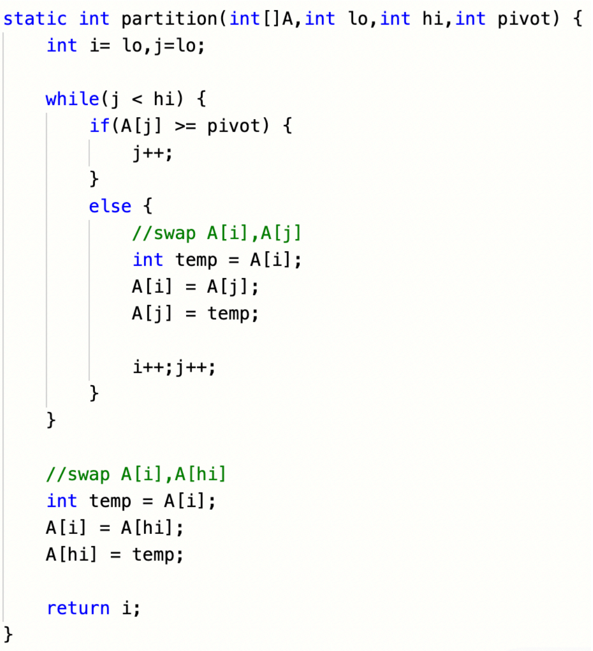








```
static void quicksort(int[]A,int lo,int hi) {
    if(lo >= hi) {
        return;
    }

    int pivot = A[hi];

    int pi = partition(A,lo,hi,pivot);

    quicksort(A,lo,pi-1);
    quicksort(A,pi+1,hi);
}
```

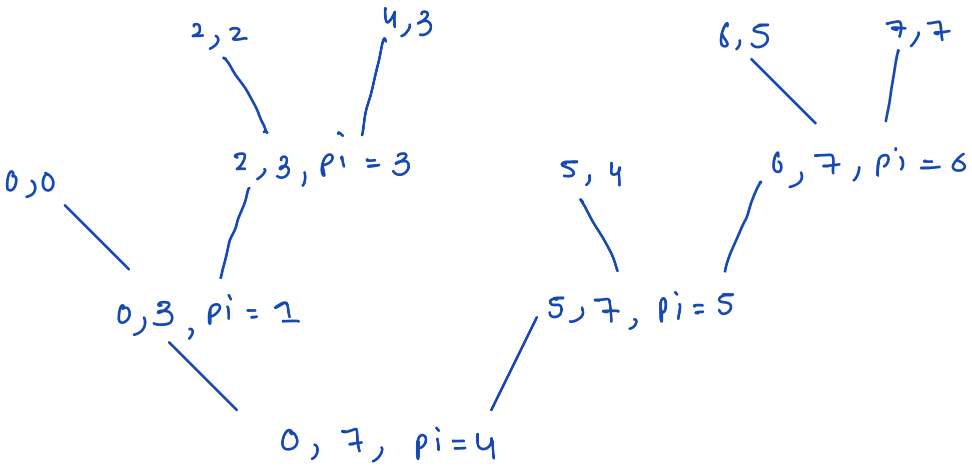


```
static int partition(int[]A,int lo,int hi,int pivot) {
    int i= lo,j=lo;

    while(j < hi) {
        if(A[j] >= pivot) {
            j++;
        }
        else {
            //swap A[i],A[j]
            int temp = A[i];
            A[i] = A[j];
            A[j] = temp;

            i++;j++;
        }
    }

    //swap A[i],A[hi]
    int temp = A[i];
    A[i] = A[hi];
    A[hi] = temp;

    return i;
}
```

[1 2 3 5 6 7 8 9]

0 1 2 3 4 5 7 7

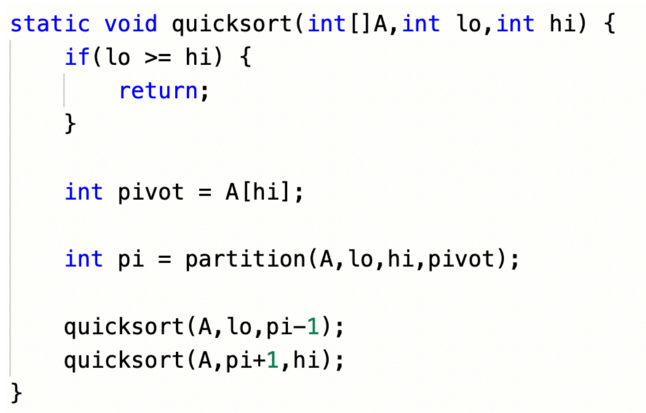
```
if (A[j] >= pivot) {
        j++;
}
else {
        swap A[i], A[j]
        i++, j++;
}
```



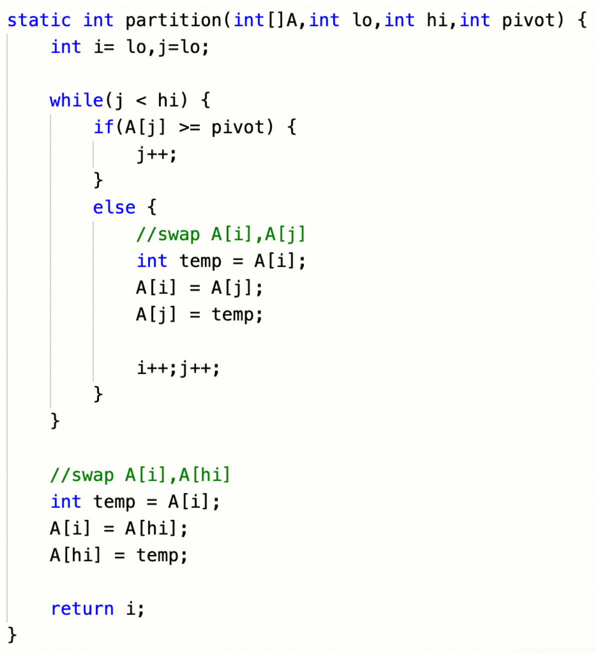
(lo, hi, pi)

TC and SC analysis

i) Quicksort

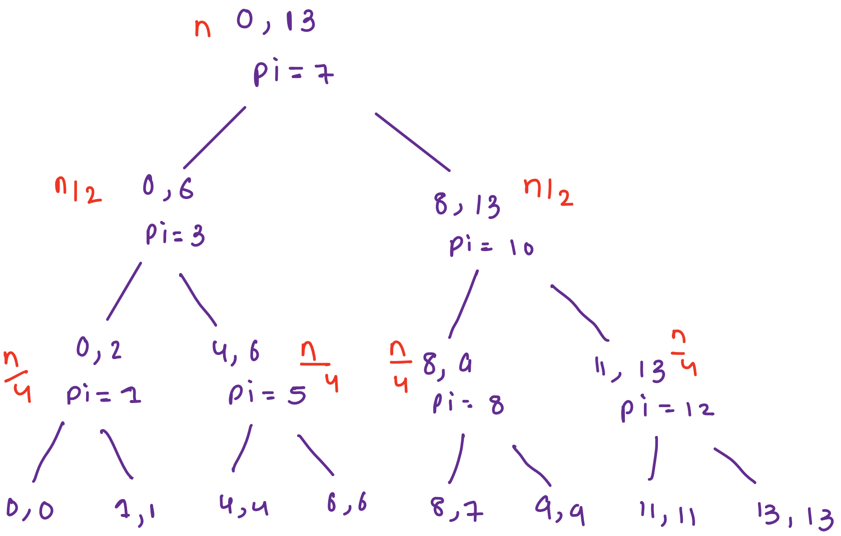


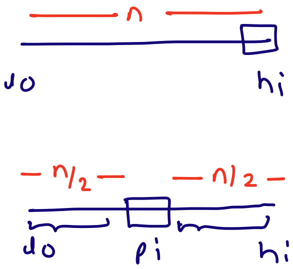
```
static void quicksort(int[]A,int lo,int hi) {
    if(lo >= hi) {
        return;
    }

    int pivot = A[hi];

    int pi = partition(A,lo,hi,pivot);

    quicksort(A,lo,pi-1);
    quicksort(A,pi+1,hi);
}
```



```
static int partition(int[]A,int lo,int hi,int pivot) {
    int i= lo,j=lo;

    while(j < hi) {
        if(A[j] >= pivot) {
            j++;
        }
        else {
            //swap A[i],A[j]
            int temp = A[i];
            A[i] = A[j];
            A[j] = temp;

            i++;j++;
        }
    }

    //swap A[i],A[hi]
    int temp = A[i];
    A[i] = A[hi];
    A[hi] = temp;

    return i;
}
```

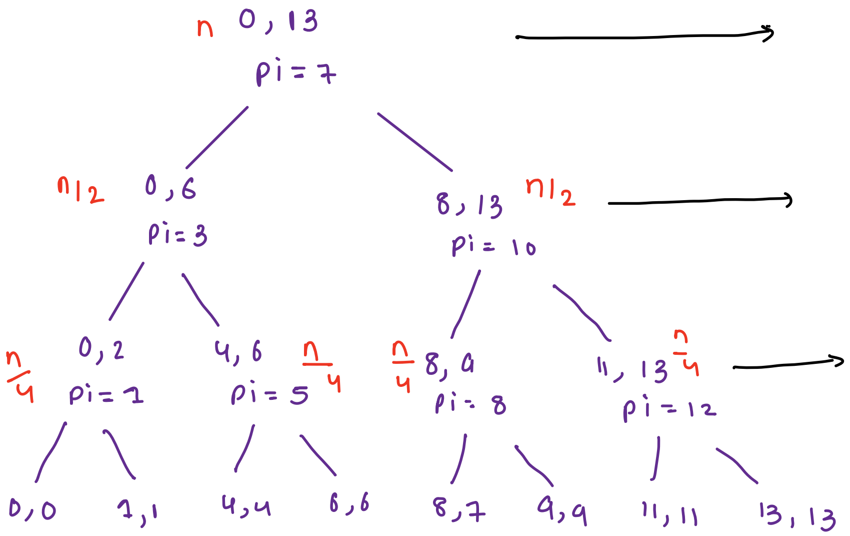
Best scenario =





pi turns out to be mid of lo to hi

n → length of original array



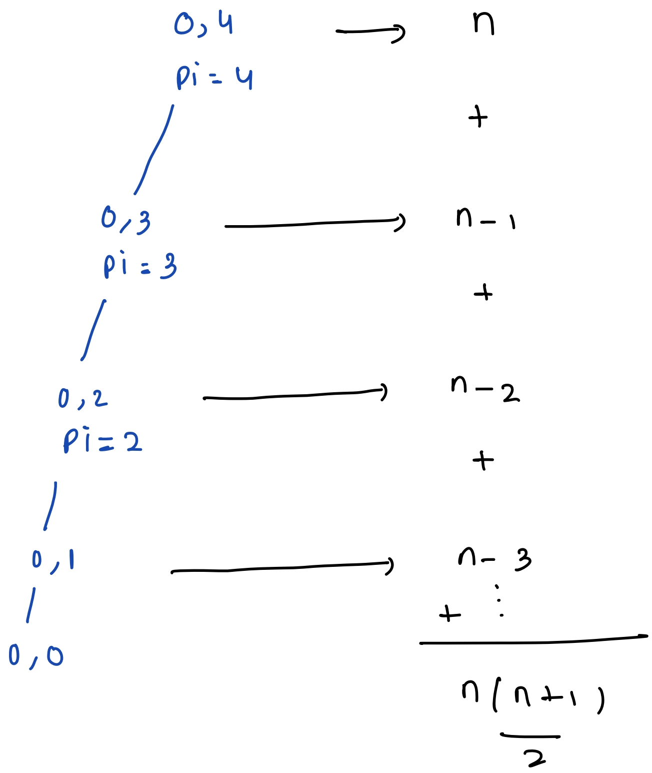
$$\left.\begin{array}{c} 1 * n \\ + \\ \cancel{2} * \frac{n}{\cancel{2}} \\ + \\ \cancel{4} * \frac{n}{\cancel{4}} \end{array}\right\} 3$$

$$\begin{array}{c} + \\ \vdots \\ \hline n \log_2 n \end{array}$$

	T C	S C (call stack)
Best	$O(n \log_2 n)$	$O(\log_2 n)$
worst	$O(n^2)$	$O(n)$

TC of Quicksort : $O(n \log_2 n)$ to $O(n^2)$

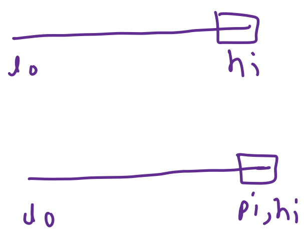
SC of Quicksort : $O(\log_2 n)$ to $O(n)$



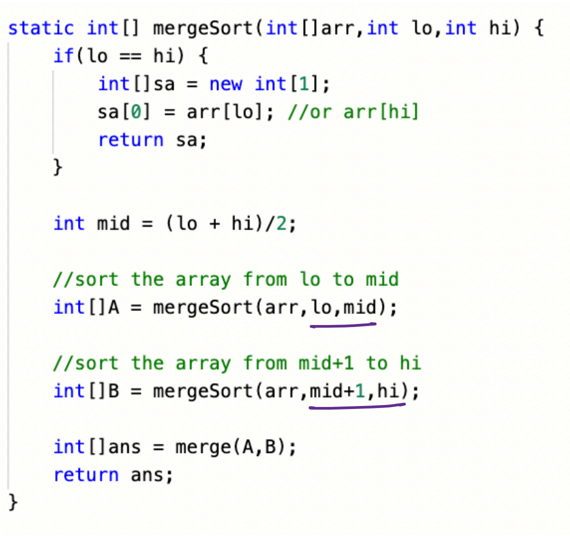
worst case

when pi turns out to be on any of the corner every time.

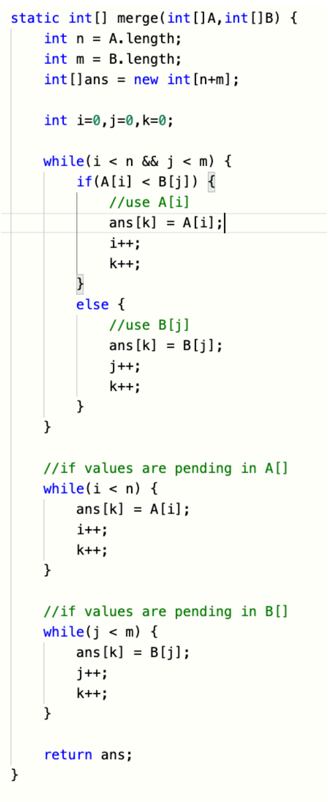
1 3 4 6 8



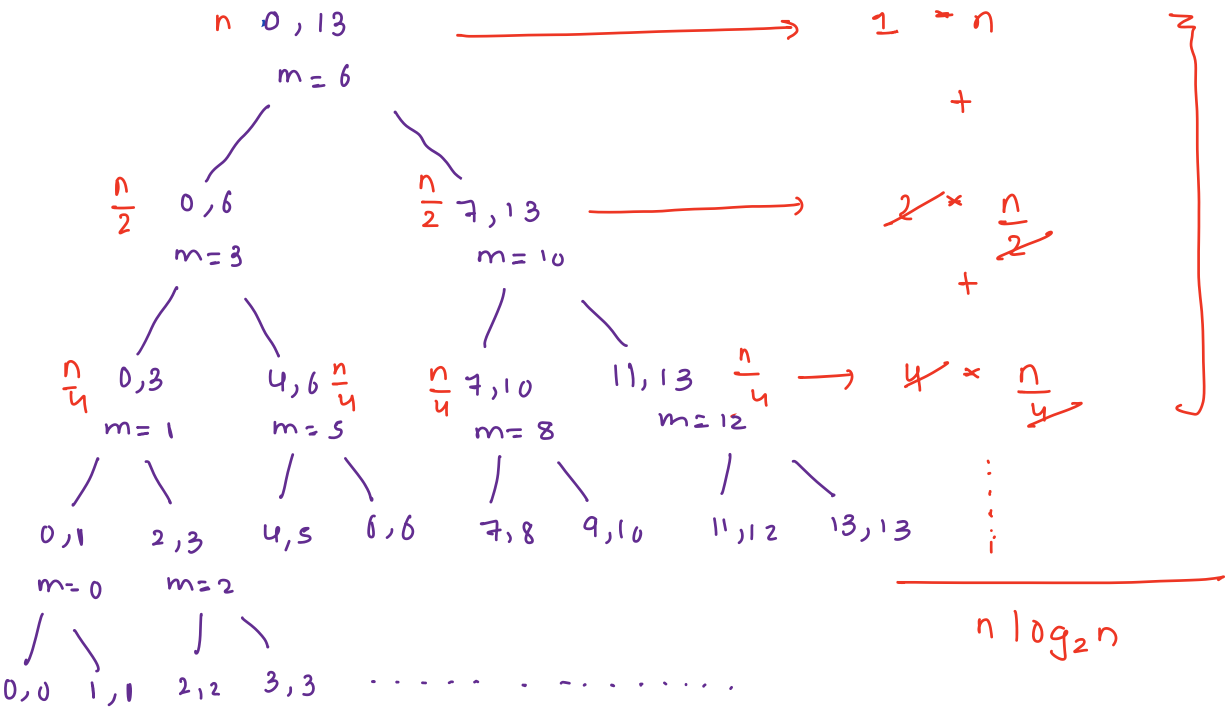
merge sort



```java
static int[] mergeSort(int[]arr,int lo,int hi) {
    if(lo == hi) {
        int[]sa = new int[1];
        sa[0] = arr[lo]; //or arr[hi]
        return sa;
    }

    int mid = (lo + hi)/2;

    //sort the array from lo to mid
    int[]A = mergeSort(arr,lo,mid);

    //sort the array from mid+1 to hi
    int[]B = mergeSort(arr,mid+1,hi);

    int[]ans = merge(A,B);
    return ans;
}
```



```java
static int[] merge(int[]A,int[]B) {
    int n = A.length;
    int m = B.length;
    int[]ans = new int[n+m];

    int i=0,j=0,k=0;

    while(i < n && j < m) {
        if(A[i] < B[j]) {
            //use A[i]
            ans[k] = A[i];
            i++;
            k++;
        }
        else {
            //use B[j]
            ans[k] = B[j];
            j++;
            k++;
        }
    }

    //if values are pending in A[]
    while(i < n) {
        ans[k] = A[i];
        i++;
        k++;
    }

    //if values are pending in B[]
    while(j < m) {
        ans[k] = B[j];
        j++;
        k++;
    }

    return ans;
}
```

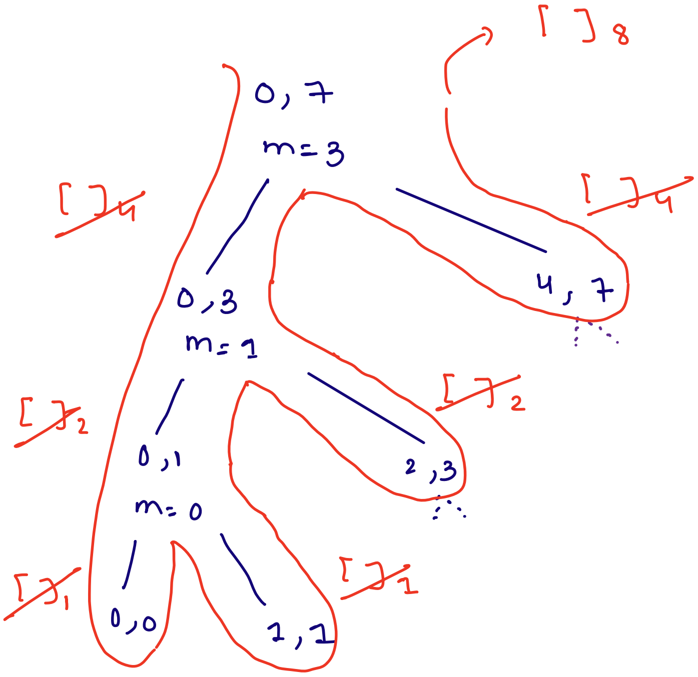


TC of merge sort : $O(n\log_2 n)$

SC of merge sort : → storage created by you : n

→ call stack space : $\log_2 n$

$O(n)$



contest detail

1) tomorrow, 8am to 8pm

2) 90 min, 2 Ques (1 Hashing, 1 recursion)

3) Discussion → 10pm [1 hr]

Doubts

[94 , 30 , 3]

ascending array

a b

94, 30 → "94" "30" (as bs)

"9430" "3094"

30 < 94

30	3	94

a b

30 , 3 → "30" "3" (as bs)

"303" "330"

30 < 3

94, 3 → "94" "3"

"943" "394"

3 < 94