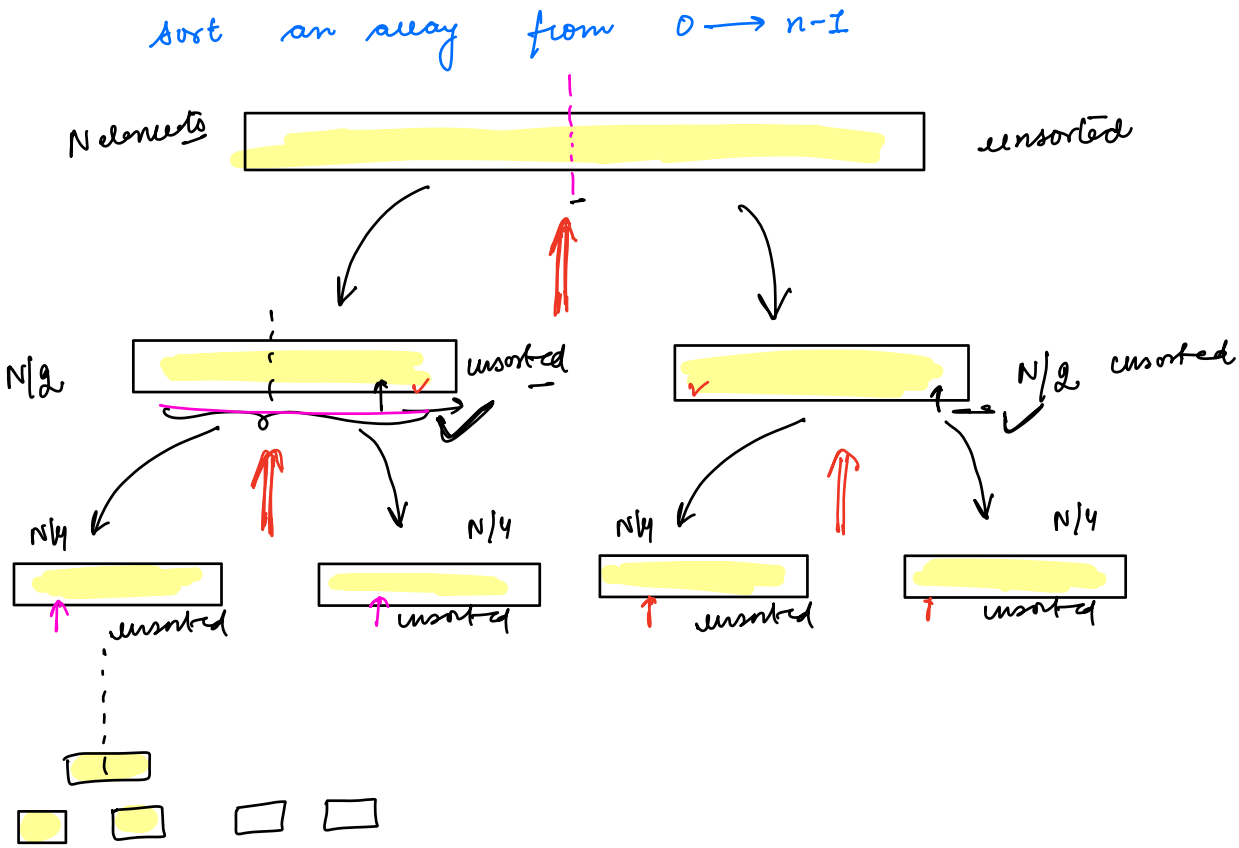
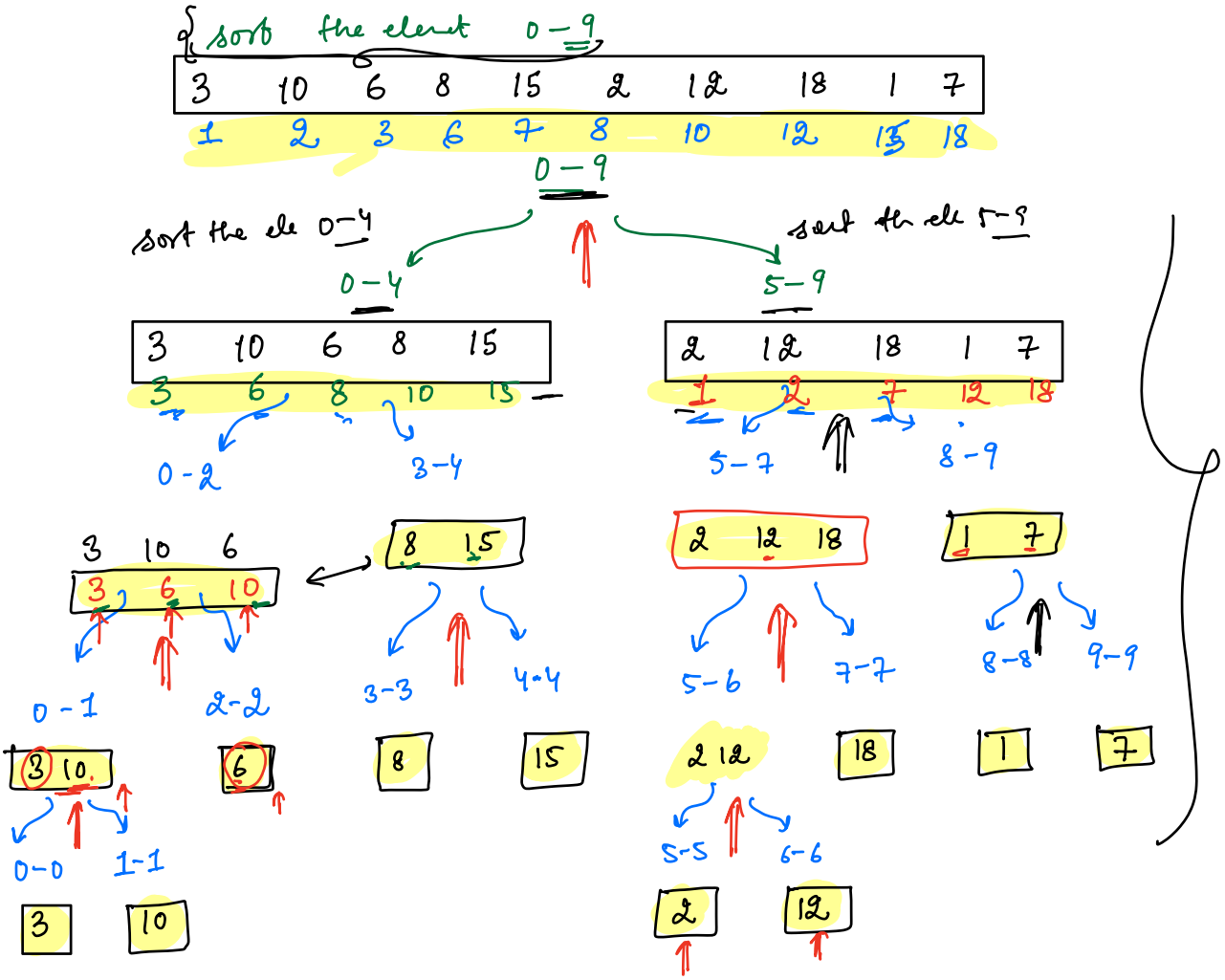

Sort an array from 0 → n-1
N elements
unsorted
N/2
unsorted
N/2
unsorted
N/4
N/4
N/4
N/4
unsorted
unsorted
unsorted
unsorted



Merge sort

```
void sort mergeSort ( int arr[], int l, int r)
{   // sort your arr from l-r
    if ( l == r) return;
    int mid = (l + (r-l)/2
    mergesort( arr, l, mid);      T(N/2)
    mergesort( arr, mid+1, r);    T(N/2)

    merge( arr, l, mid+1, r);
}
```

5-9 : 9+5/2, 5/2 = 2, 9-5/2, 4/2 = 2

(l+l)/2

l r

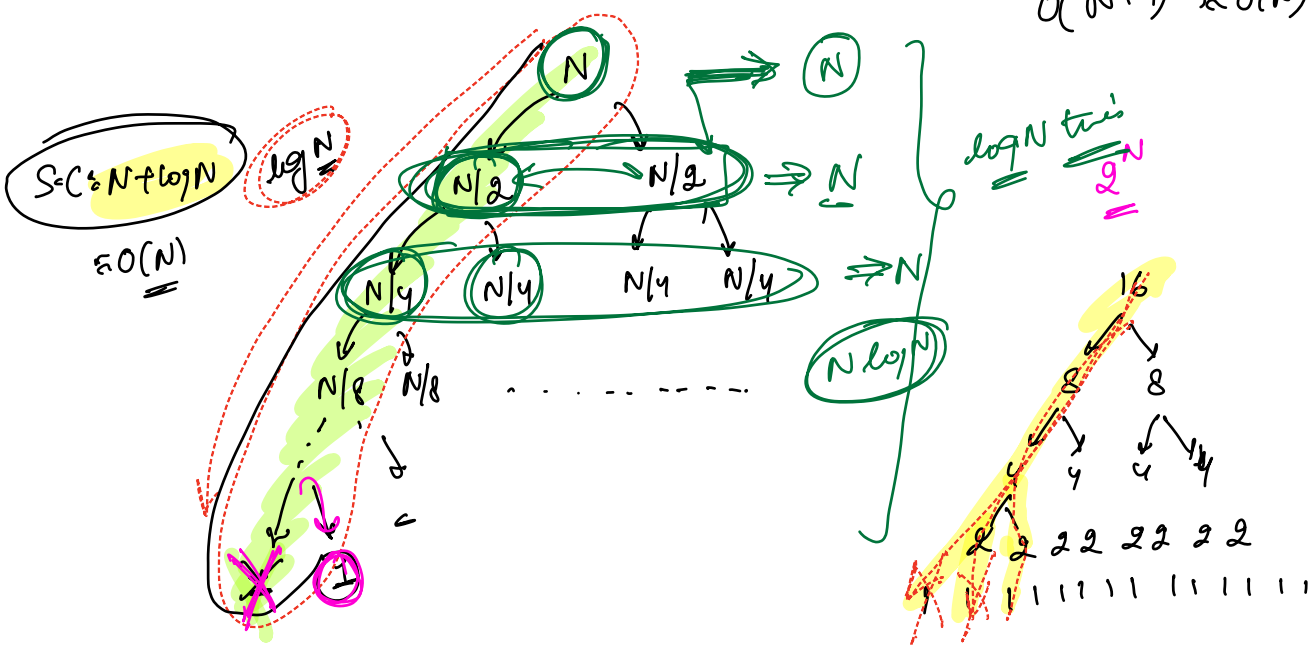
l, y, r
l-y-1
y-l

$T(N) = T(N/2) + T(N/2) + N$

$T(N) = 2T(N/2) + N \rightarrow N \log N$

$O(N+c)$ } $O(N)$

$O(N+1) \approx O(N)$



Q array of N integers, find total count of pairs (i,j)

$i<j$ && $a[i]>a[j]$ → inversion pair

later value – smaller index

0	1	2	3	4	5	6	7	8	9
10	3	8	15	6	12	2	18	7	1

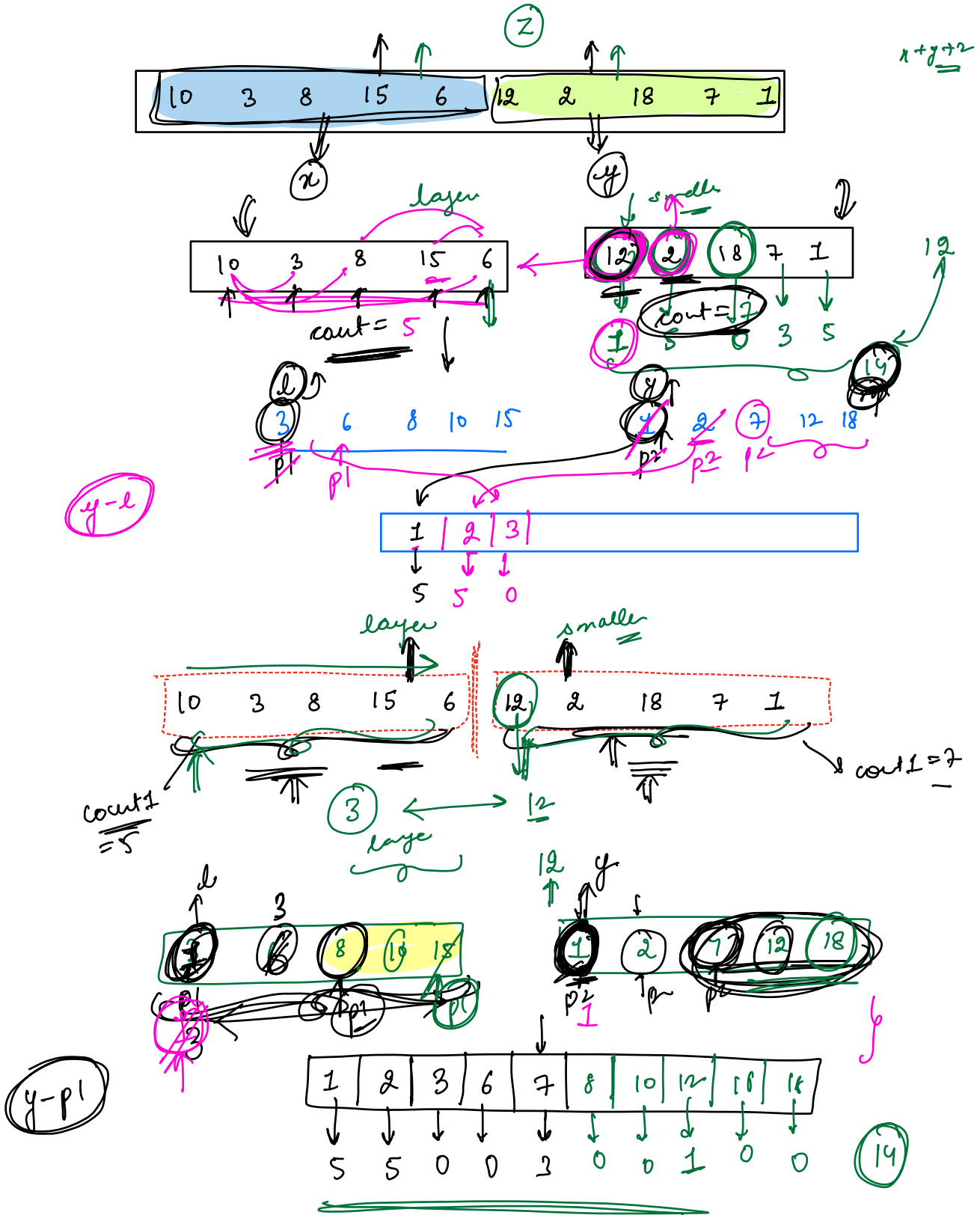
3, 9
15 1

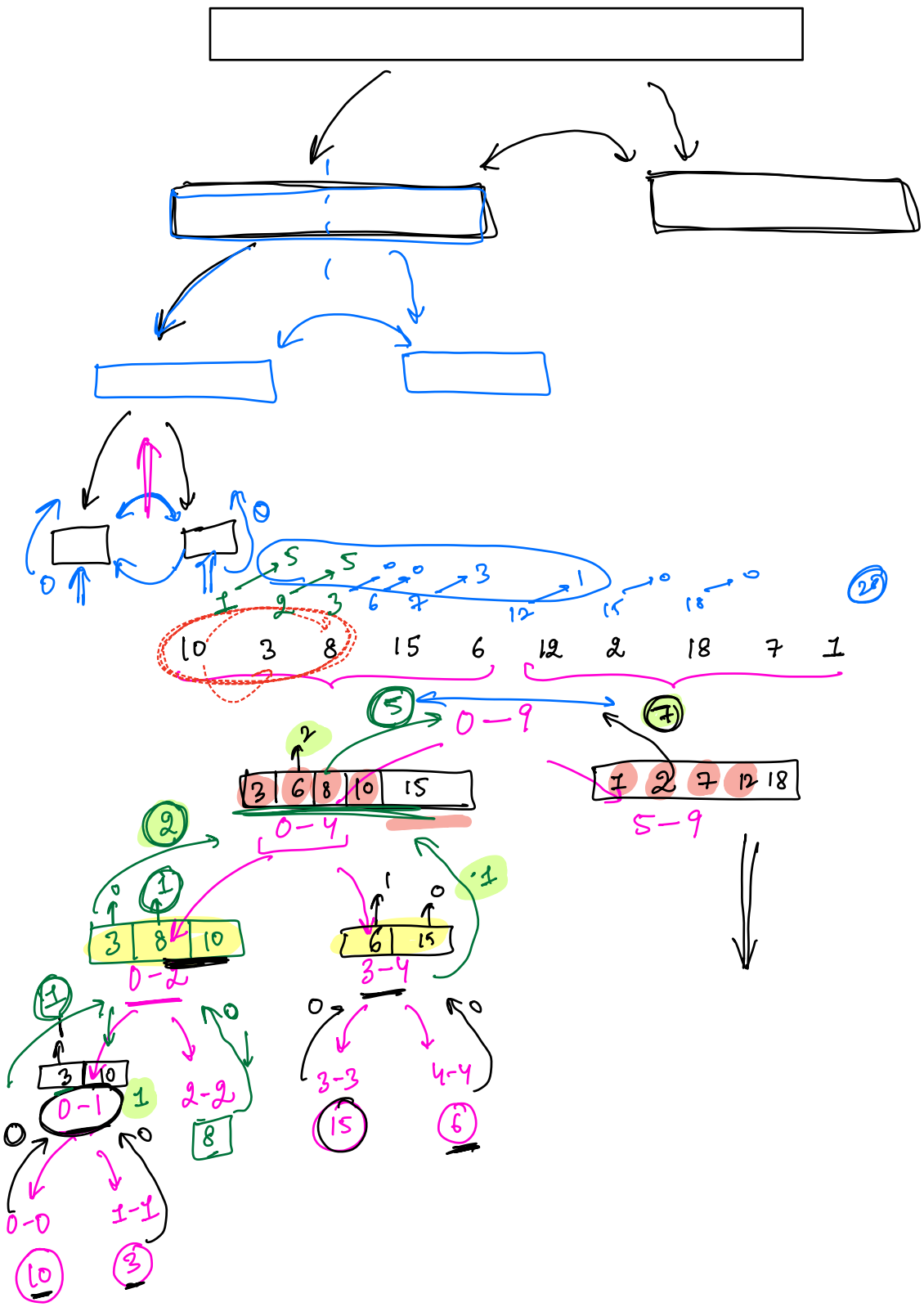
2 18

10	3	8	15	6	12	2	18	7	1
0	1	1	0	3	1	6	0	5	9

} 26

$a[i] > a[i+1]$ ⇐ $a[i]>a[j]$, $i<j$


z
x+y+2
10 3 8 15 6
12 2 18 7 1
x
y
larger
smaller
count = 5
count = 7
y-l
y-p1
count1 = 5
count1 = 7
larger
smaller

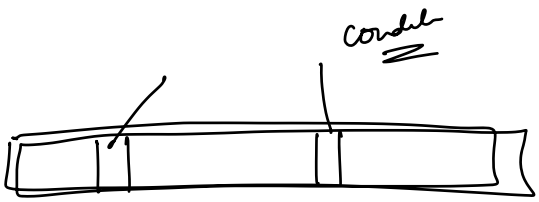

10 3 8 15 6 12 2 18 7 1
0-9
5-9
0-4
0-2
3-4
0-1
0-0
1-1
2-2
3-3
4-4

```
int inversion(int arr[], int l, int r)
{
    if (l==r) return 0;
    mid = (l+r)/2
    int x = inversion(arr, l, mid)
    int y = inversion(arr, mid+1, r)

    int z = merge(arr, l, mid+1, r);

    return x+y+z;
}
```

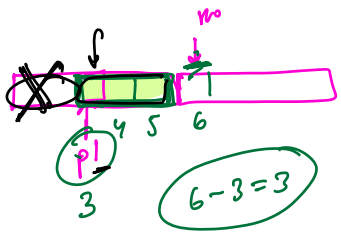
T.C: nlogn

S.C: n+logn

divide & conquer



```
int merge( int arr[], int l, int m, int r)
{
    int p1 = l;
        p2 = m;
        p3 = 0;   int count=0;
    int temp[ r-l+1];
        while ( p1 < m && p2 <= r )
        {
                if( arr[p1] <= arr[p2])
            {   temp[p3++] = arr[p1++];
            }
            else
            {   temp[p3++]= arr[p2++]; }
                count += m-p1;
            }
        }
        while (     ) {   }
        while (     ) {   }

//      for( i = 0;  i< r-l+1; i++)
        {
            arr[ l+i] = temp[i];
        }
    return count;
}
```



sort ( ) ⟹ asc order — most of the cases of magnitude

desc → sort in asc + reverse

sort the array in ascorder of no of factors

sort (all, ,) → comparator f^n

range

if (a[i] > a[i+1]) swap

if (a[i] < a[i+1]) swap ( )

a[i] a[i+1] >

count(x)

# of factors x

if (count(a[i]) > count(a[i+1])) swap

4 2 8 2 2 8

bool comp (int a, int b)

{

// if you want 'a' to come befor 'b'

return true; // asc

if (a <= b) return true; // no swap

return false;

desc if (a > b) ret true;

}

sort acc to inc order of factors count

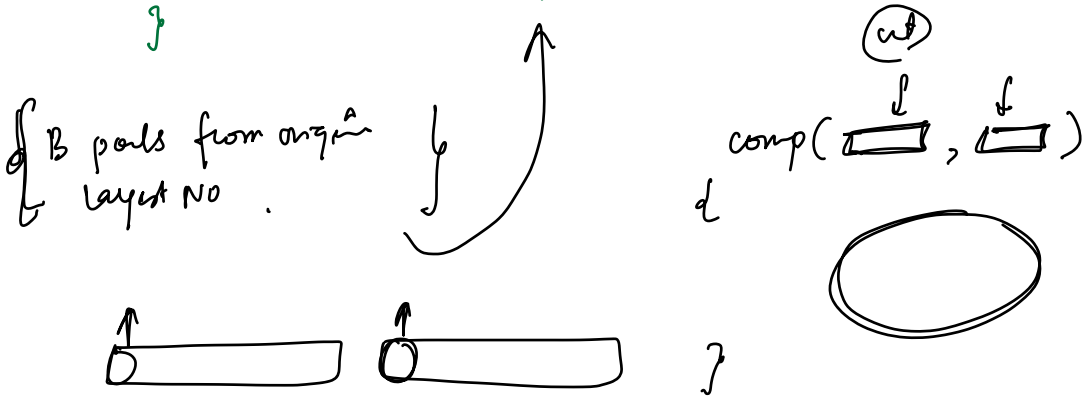
```
bool comp( int a, int b)
{
    if ( count(a) <= count(b))
      int true;
      return false;
}
```

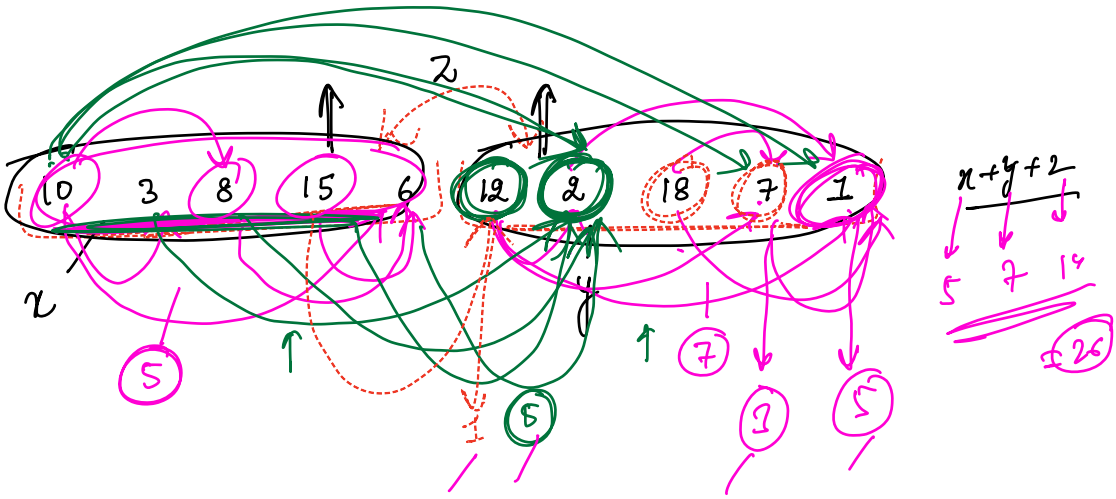
array → sort acc to asc order of count of factors,
if two numbers have equal no of factors,
sort acc to magnitude of elem

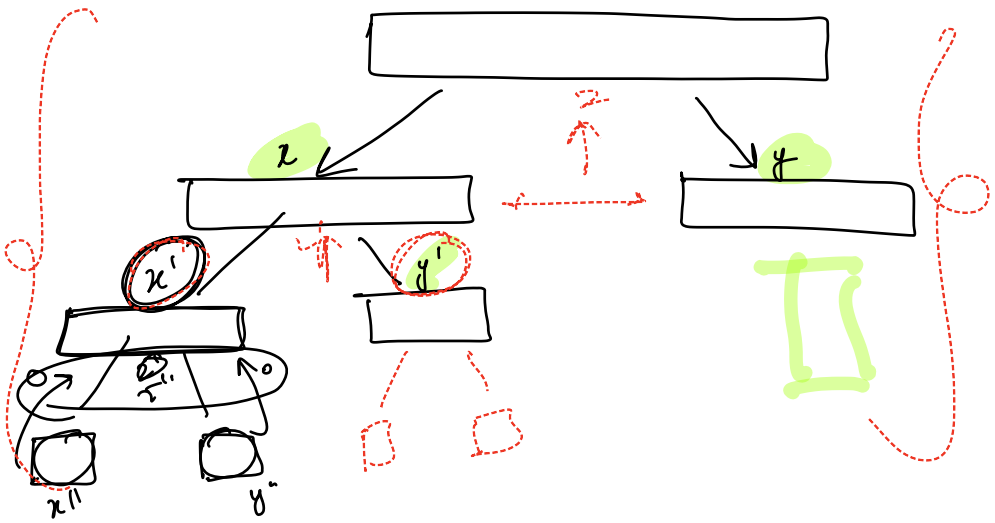
```
comp( a, b)
{
    int x = count(a);
    int y = count(b);
    if (x < y)
        return true;
    else ( x == y && a < b)
        return true;
    else
        return false;
}
```

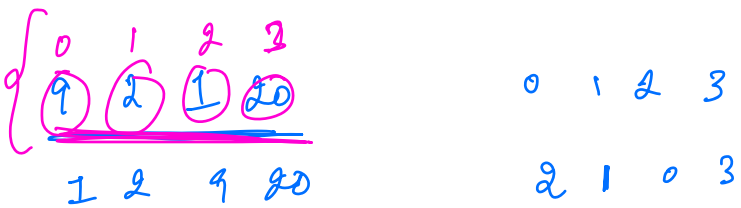
a = 7 b = 5

x <= y
return true
return false;

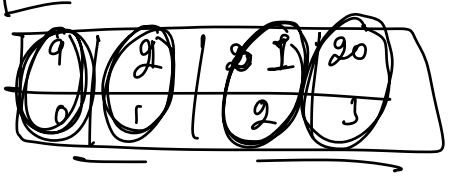








vector < pair



comp(pair< a >, pair< b >)

{

a.first < b.first


Selet
Bubble
Ins
Merge
cout
quick sat
Heaps
Topolog
Radix ✓
Buck ✗