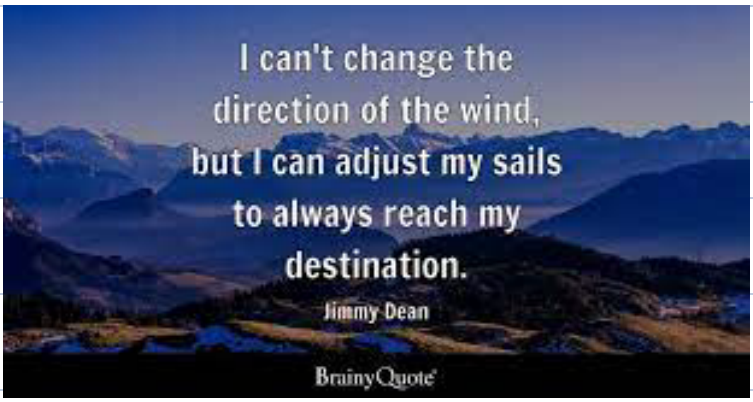
Sorting 1

Hello Everyone



Today's Agenda :

1) Create smallest number using an array
2) Count Sort on large value
3) Count sort on negative number
4) Merge two sorted array
5) Merge Sort
6) Calculate no. of pair A[i] > A[j]
7) Inversion count
8) Stable sort and inplace

Q) Find the smallest number that can be formed by rearranging the digits of given number in an array. Return the smallest number in the form an array.

Eg. A[] = {6, 3, 4, 2, 7, 2, 1}
Answer: {1, 2, 2, 3, 4, 6, 7}

$0 <= A[i] <= 9$

A[] → 4, 2, 7, 3, 9, 0
→ 0, 2, 3, 4, 7, 9

Idea 1: Sort the array. → O(nlogn)

Idea 2:

6, 3, 4, 2, 7, 2, 1, 3

freq → 0 1 2 2 1 0 1 1 0 0

0 1 2 3 4 5 6 7 8 9

ans → 1 2 2 3 3 4 6 7

```
int freq[10];

for i=0 to n-1
    int val = A[i];
    freq[val] = freq[val]+1;
```

O(n)

0 1 2 2 1 0 1 1 0 0

0 1 2 3 4 5 6 7 8 9

0 1 2 3 4 5 6 7

6, 3, 4, 2, 7, 2, 1, 3

```
int[A.length] ans;

int index = 0;
for d → 0 to 9
    for (int i = 1; i <= freq[d]; i++)
        ans[index] = d;
        index++;
}
```

0	1	2	3	4	5	6	7	8	9
0	~~0~~ 1	~~0~~ x2	~~0~~ x2	~~0~~ 1	0	~~0~~ 1	~~0~~ 1	0	0

d: 0 1 2 3 4

5 6 7 8

⇒ Count Sort

{1, 2, 2, 3, 3, 4, 6, 7}

TC: O(n)
SC: O(1)

arr[] → [10, 20, 30, 999]

0 999

$0 <= A[i] <= 10^9$

Will Count sort work if the rang of A[i] is more than 10^9 ?.

Integer → 4 bytes.

A[10^9] Space → $10^9 * 4$ bytes.

1 GB = 10^9 bytes.

$= \frac{4 * 10^9}{10^9}$ GB

≅ 4 GB

* Count sort on negative number.

A[] = {-2, 3, 8, 3, -2, 3}

int val = arr[i]
freq[val - smallest]

Smallest = -2
largest = 8

Range = 8 - (-2) + 1
= 11

freq = [2, _, _, _, _, 3, _, _, _, _, 1]

0	1	2	3	4	5	6	7	8	9	10
(-2)	(-1)	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)

Element ⇒ index + smallest.

O(N) → you have to find smallest, largest ele.

Code:

int[largest - smallest + 1] freq;

SC: O(Range)

int[] ans ← input length

```
for(int i=0; i<A.length ; i++)
{
    int val = A[i];
    int freqIndex = val - smallest;

    freq[freqIndex]++;
}
```

O(N)

```
int index=0
for(int i=0; i<freq.length; i++)
{
    int val = i + smallest;
    int freqCount = freq[i];

    for(int cnt=1 ; cnt<=freqCount; cnt++)
    {
        ans[index] = val;
        index++;
    }
}
```

TC: O(Range + N) / O(N)

SC: O(Range)

→ Largest - Smallest + 1

Q.) Merge two sorted array.

Given an integer array where all odd elements are sorted and even elements are sorted. We have to sort the entire array.

A[] = { 2, 5, 4, 8, 11, 13, 10, 15, 21 }

Idea 1: Sort the entire array → O(NlogN)

idea 2:

A[] = { 2, 5, 4, 8, 11, 13, 10, 15, 21 }

Odd → 5
Even → 4

Odd → | 5 | 11 | 13 | 15 | 21 |
Even → | 2 | 4 | 8 | 10 |

o

Odd → | 5 | 11 | 13 | 15 | 21 |

e

Even → | 2 | 4 | 8 | 10 |

final sorted Array

2 4 5 8 10 11 13 15 21

```
void merge ( int[] A )
{
    int N = A.length;
    // int oddCount : Count of odd elem
    // int evenCount : Count of even elem.
```

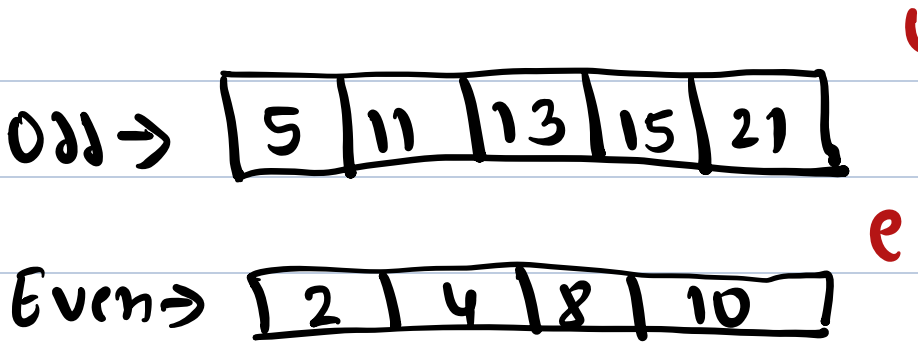
O(N){

```
int Even[evenCount];
int Odd[oddCount];

int odd_index = 0;
int even_index = 0;
int i = 0;

for (int i = 0; i < N; i++)
{
    if(A[i] % 2 == 0)
    {
        Even[even_index] = A[i];
        even_index++;
    }
    else
    {
        Odd[odd_index] = A[i];
        odd_index++;
    }
}
```

O(N)

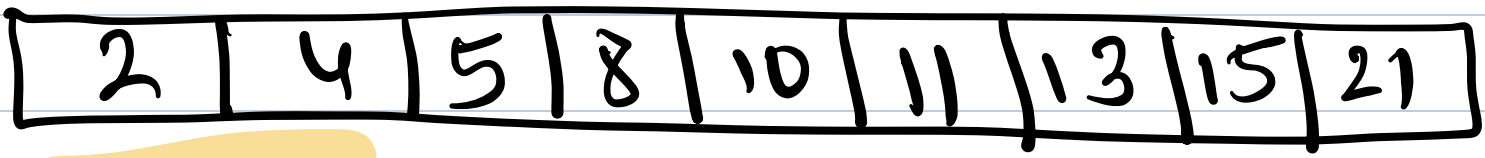
```
int index=0, int odd-index=0,
  int even-index=0;
```

A[] = { 2, 5, 4, 8, 11, 13, 10, 15, 21 }




```
int ans[n];
while (odd-index < odd.length &&
       even-index < even.length)
{
    if ( even[even-index] < odd[odd-index]
    {
        ans[index] = even[even-index];
        index++;
        even-index++;
    }
    else
```

```
{
        ans[index] = odd[odd-index];
        odd-index++;
        index++;
    }
}
```



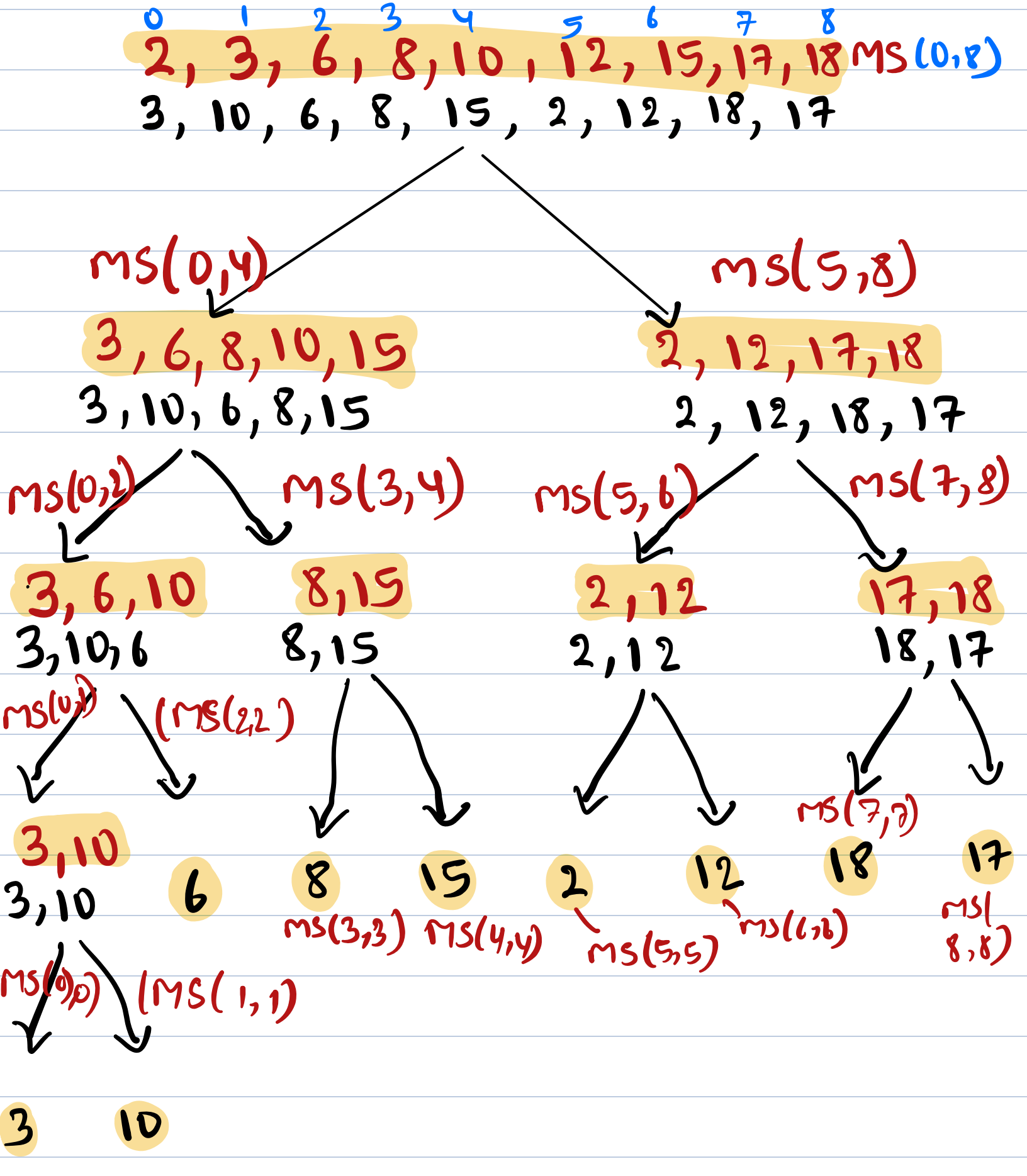

```
while (even-index < even.length)
{
    ans[index] = Even[even-index]
    even-index++;
    index++;
}

while (odd-index < odd.length)
{
    ans[index] = odd[odd-index];
    odd-index++;
    index++;
}
```

y

TC: O(n)
SC: O(n)

Break: 10.30 pm

Merge Sort:

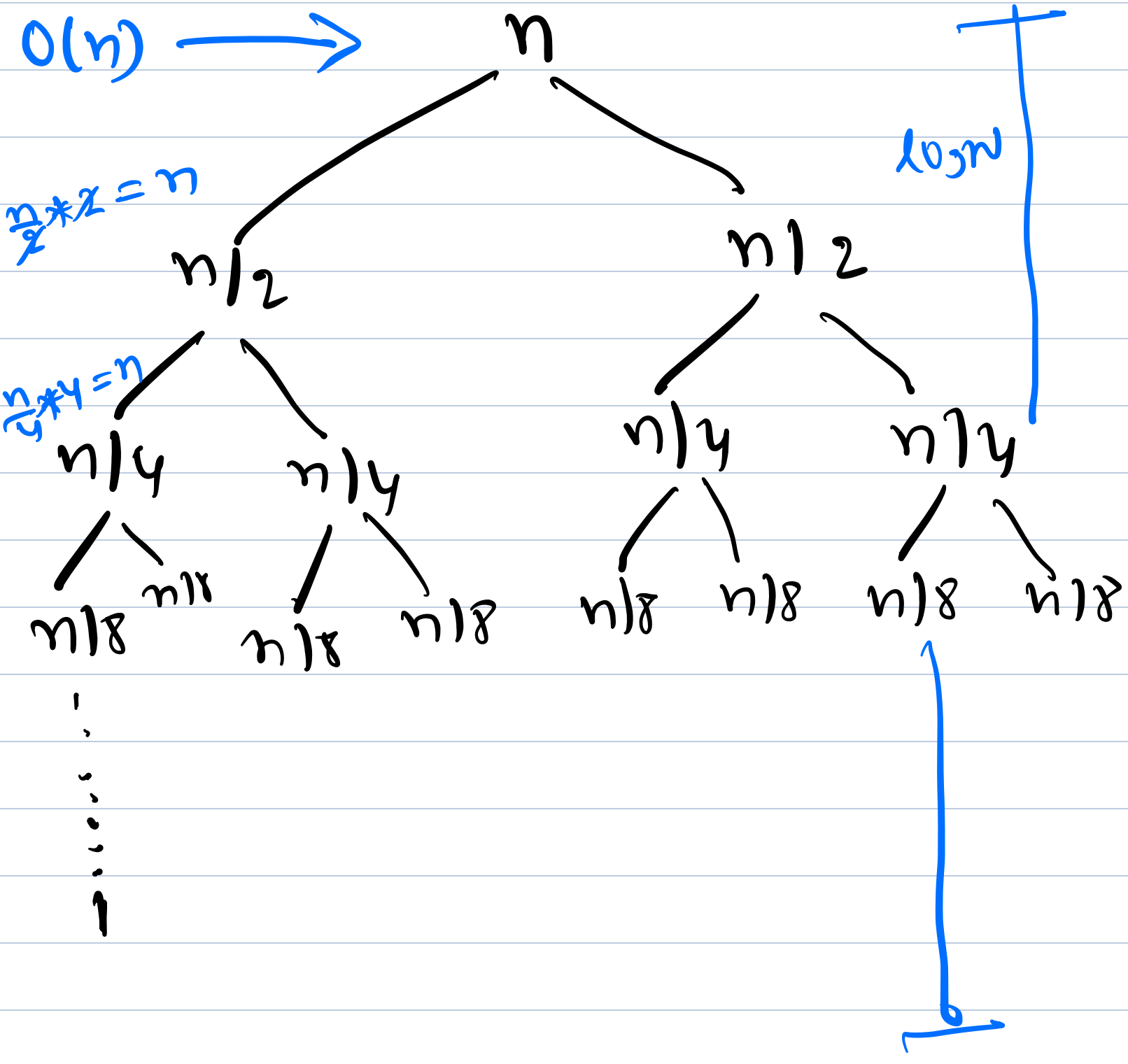

0 1 2 3 4 5 6 7 8
2, 3, 6, 8, 10, 12, 15, 17, 18
MS(0,8)
3, 10, 6, 8, 15, 2, 12, 18, 17
MS(0,4)
3, 6, 8, 10, 15
3, 10, 6, 8, 15
MS(5,8)
2, 12, 17, 18
2, 12, 18, 17
MS(0,2)
MS(3,4)
MS(5,6)
MS(7,8)
3, 6, 10
8, 15
2, 12
17, 18
3, 10, 6
8, 15
2, 12
18, 17
MS(0,1)
MS(2,2)
MS(7,7)
3, 10
6
8
15
2
12
18
17
MS(3,3)
MS(4,4)
MS(5,5)
MS(6,6)
MS(8,8)
MS(0,0)
MS(1,1)
3
10

```
void mergeSort(int[] arr, int l,
               int r)
{
    if(l==r){ return; }

    int mid = (l+r)/2;
    mergeSort(arr, l, mid);
    mergeSort(arr, mid+1, r);
    merge(arr, l, mid, r);
}

void merge(int[] arr, int l, int
           mid, int r)
{
    int n= r-l+1;
    int leftLength= mid-l+1;
    int rightLength= r- mid;
```

```
int left[leftlength];
int right[rightlength];
    int index=0
for(int i=left; i<=mid; i++)
{
    left['index'] = arr[i]
     index++;
}

  index=0;

for(int i=mid+1; i<=r; i++)
{
    right[index] = arr[i];
     index++;
}
```

```
index = l;
int p1 = 0;    // left array
int p2 = 0;    // right array

while (p1 < leftLength && p2 < rightLength)
{
    if (left[p1] <= right[p2])
    {
        arr[index] = left[p1];
        index++;
        p1++;
    }
    else
    {
        arr[index] = right[p2];
        index++;
        p2++;
    }
}
```

```
while (p1 < leftlength)
{
    arr[index] = left[p1];
    index++;
    p1++;
}

while (p2 < rightlength)
{
    arr[index] = right[p2];
    index++;
    p2++;
}
}
```



TC: $O(\log n) * n = O(n \log n)$

SC: $O(\log n)$ + n

↑ recursive

↓ merge

/stack space operation.

Q) Given two array, A[n] and B[m] (n: size of A, m: size of B)
calculate the no. of pair i, j
such that A[i] > B[j].

A[3] = { 7, 0, 3 }
B[3] = { 2, 0, 6 }

(7,2) (7,0) (7,6) (3,2) (3,0)

BF: use two loop.
TC: O(n*m)
SC: O(1)

Approach 2:

1) sort array A : O(nlogn)
2) sort array B : O(mlogm)

Tc: $O(n \log n + m \log m + n + m)$

*) Given an arr[N], calculate no. of pair [i, j] such that i < j and arr[i] > arr[j]. i and j are index of an array.

A = { 4, 5, 1, 2, 6, 3 }
0 1 2 3 4 5

Ans = 7

(0, 2) (0, 3) (0, 5)
(1, 2) (1, 3) (1, 5)
(4, 5)

Quiz: { 5, 2, 6, 1 }
0 1 2 3

Ans = 4

(0, 1) (0, 3) (1, 3) (2, 3)

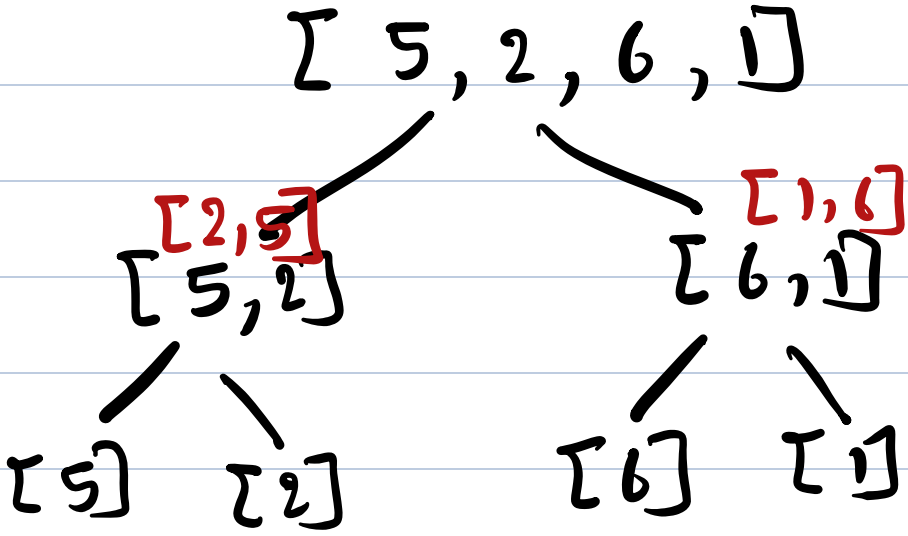
Quiz: {5, 3, 1, 4, 2}
0 1 2 3 4

Ans = 7.

(0,1) (0,2) (0,3) (0,4)
(1,2) (1,4)
(3,4)

BF Approach : O(n^2) using nested loop

{{TODO}}



Stable sort:

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

Way 1

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

↘ stable

Way 2

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

↘ not stable.

Airport

Economy Business privilleged.

Inplace: you are not using extra space.

Doubt Session

{1, 3, 5} ⇒ 135

ans = ans * 10 + digit.

{10, 22, 101, 74}

Max = 101

Min = 10

A[3] = {3, 5, 7}

p1

B[3] = {0, 2, 6}

p2

ans = ∅ → 3 → 37

```
for (int i=1 ; i<=10; i++)
{
    if (i == 5)
    { for (i=0; i<=N; i++)
```