## Agenda

1) Bubble sort
2) Insertion sort
} Algorithms

3) Merge 2 sorted arrays (Ques)
4) Merge sort } Algorithm

What is sorting: Arranging the data

2 9 15 20 31 → increasing order

15 12 10 5 -8 → decreasing order

7 6 8 12 → inc. order based on count of factors

# of 2 4 4 6

How to sort ?

```
int [] A = { 7, 9, 5, 1, 3 };
```

```
Arrays.sort(A);
```

↳ Array variable name

i) A => 1 3 5 7 9

note : Array.sort(A) => O(nlogn) — TC

# Bubble Sort

A = 3 8 6 2 4

0 1 2 3 4

itr 1 : 3 6 2 4 8

itr 2 : 3 2 4 6 8

itr 3 : 3 2 4 6 8

itr 4 : 2 3 4 6 8

A = 3 8 6 2 4

0 1 2 3 4

itr 1 : 3 ~~8~~ ~~6~~ ~~2~~ ~~4~~ 8

itr 2 : 3 ~~6~~ ~~2~~ ~~4~~ 6 8

itr 3 : ~~3~~ ~~2~~ 4 6 8

itr 4 : 2 3 4 6 8

2 3 4 6 8

if (A[j] > A[j+1]){

swap

3

A = 4 3 1

itr 1 : 3 4 4 → 1 4 4 → 1 4 4

itr 2 : 1 4 4 → 1 3 4

1 3 4

A = 3 8 6 2 4

0 1 2 3 4

n-1 itr

itr 1 : 3 8 6 2 4 → 3 6 8 2 4 → 3 6 2 8 4 → 3 6 2 4 8

itr 2 : 3 6 2 4 8 → 3 2 6 4 8

itr 3 : 3 2 4 6 8 → 2 3 4 6 8

itr 4 : 2 3 4 6 8

2 3 4 6 8

0 to<n-i-1

| i | j |
|---|---|
| 0 | 0 to 3 |
| 1 | 0 to 2 |
| 2 | 0 to 1 |
| 3 | 0 to 0 |

```
void bubble_sort (int [] A) {
    int n = A.length;
    for (int i = 0; i < n-1; i++) {
        for (int j=0; j < n-i-1 ; j++) {
            if (A[j] > A[j+1] ) {
                int temp = A[j];
                A[j] = A[j+1];
                A[j+1] = temp;
            }
        }
    }
}
```

TC: $O(n^2)$

SC: $O(1)$

Dry run

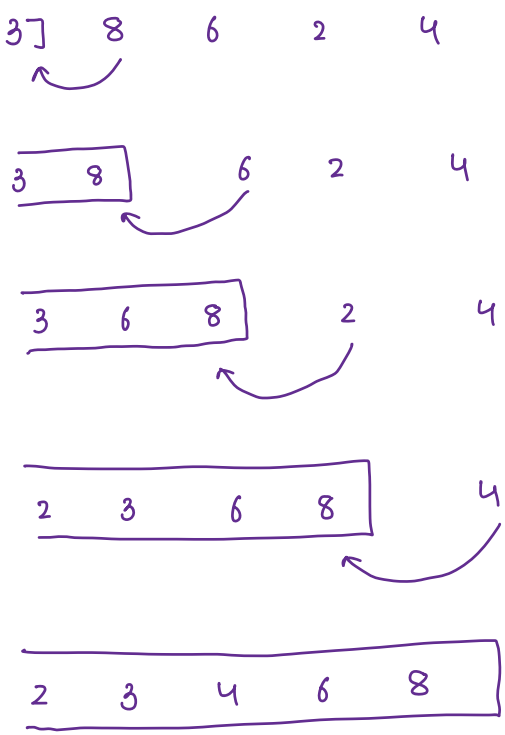
```
void bubble_sort (int [] A) {
    int n = A.length;
    for (int i = 0; i < n-1; i++) {
        for (int j=0; j < n-i-1 ; j++) {
            if (A[j] > A[j+1] ) {
                int temp = A[j];
                A[j] = A[j+1];
                A[j+1] = temp;
            }
        }
    }
}
```

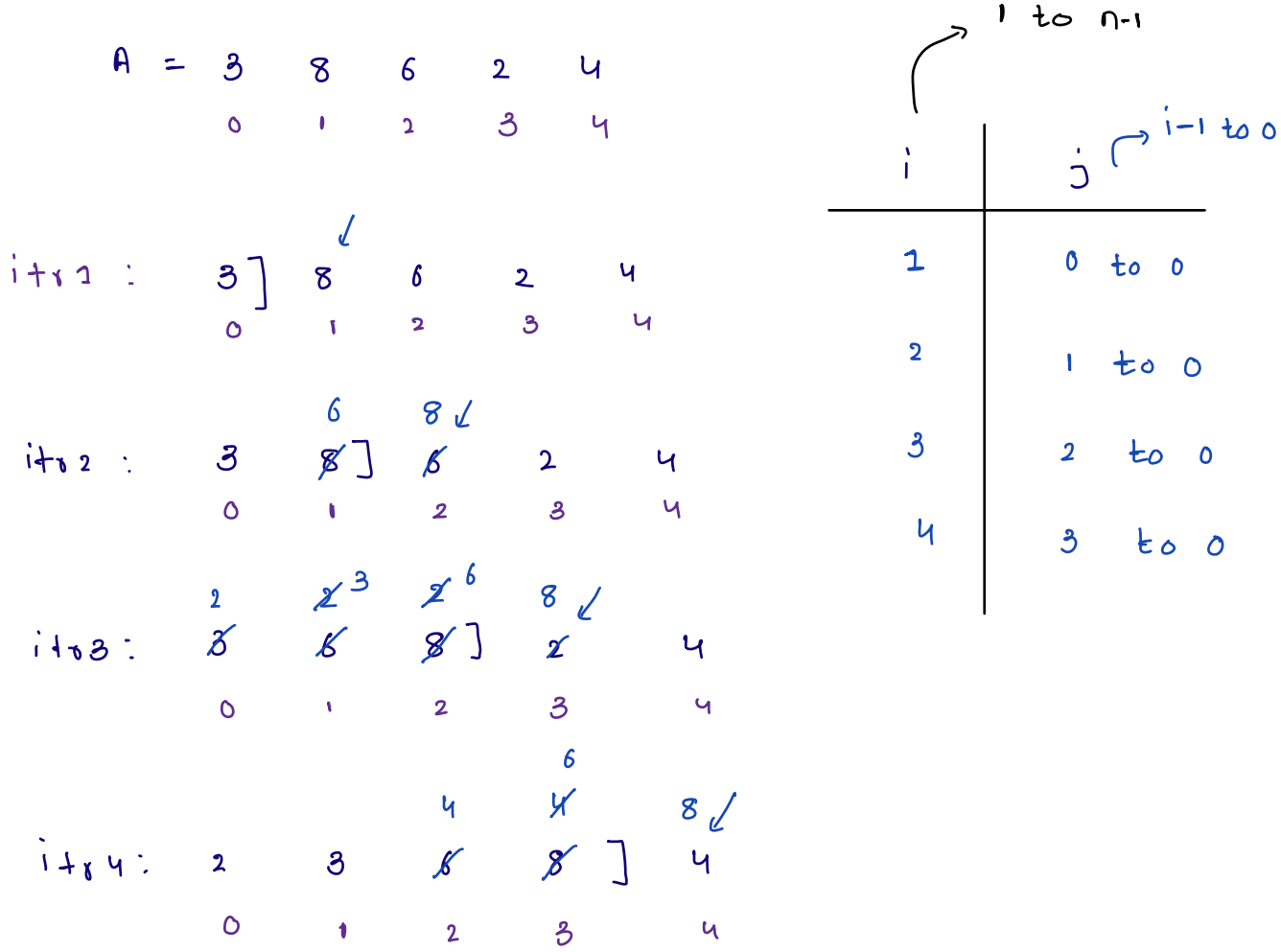
A = 8 5 4 1

| i | j | steps |
|---|---|---|
| 0 | 0, 1, 2 | 5 ~~8~~ 4 ~~8~~ 1 [8]<br>~~8~~ ~~5~~ ~~4~~ 1 |
| 1 | 0, 1 | 4 ~~5~~ 1 [5]<br>~~5~~ ~~4~~ ~~1~~ 8 |
| 2 | 0 | 1 [4]<br>~~4~~ ~~1~~ 5 8 |
| | | => 1 4 5 8 |

Insertion sort

A = 3 8 6 2 4




A = 3 8 6 2 4
0 1 2 3 4
itr 1 : 3 8 6 2 4
itr 2 : 3 8 6 2 4
itr 3 : 2 3 6 8 4
itr 4 : 2 3 6 8 4
2 3 4 6 8
1 to n-1
i-1 to 0
i
j
1 0 to 0
2 1 to 0
3 2 to 0
4 3 to 0

```
void insertion_sort (int[] A) {
    int n= A.length;

    for (int i=1; i<n; i++) {
        for (int j=i-1; j>=0; j--) {
            if( A[j] > A[j+1]) {
                int temp = A[j];
                A[j] = A[j+1];
                A[j+1] = temp;
            }
            else {
                break;
            }
        }
    }
}
```

TC: $O(n^2)$

SC: $O(1)$

```
void insertion_sort (int[] A) {
    int n = A.length;

    for (int i = 1; i < n; i++) {
        for (int j = i-1; j >= 0; j--) {
            if( A[j] > A[j+1]) {
                int temp = A[j];
                A[j] = A[j+1];
                A[j+1] = temp;
            }
            else {
                break;
            }
        }
    }
}
```

A = 8 9 4 6

| i | j | |
|---|---|---|
| 1 | 0 to 0 | 8 9 4 6 |
| 2 | 1 to 0 | 8 4 9 6 |
| 3 | 2 to 0 | 4 6 8 9 |
| | | ⇒ 4 6 8 9 |

**

Q.1 Merge two sorted arrays.

TC: O(n+m)

A => 2 5 9 12 15

B => 3 6 8 10 16 18

ans =

| 2 | 3 | 5 | 6 | 8 | 9 | 10 | 12 | 15 | 16 | 18 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

A => 2 5 9 12 15

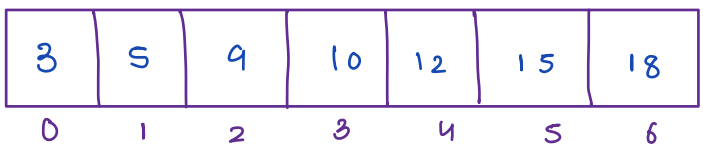
B => 3 6 8 10 16 18

ans =

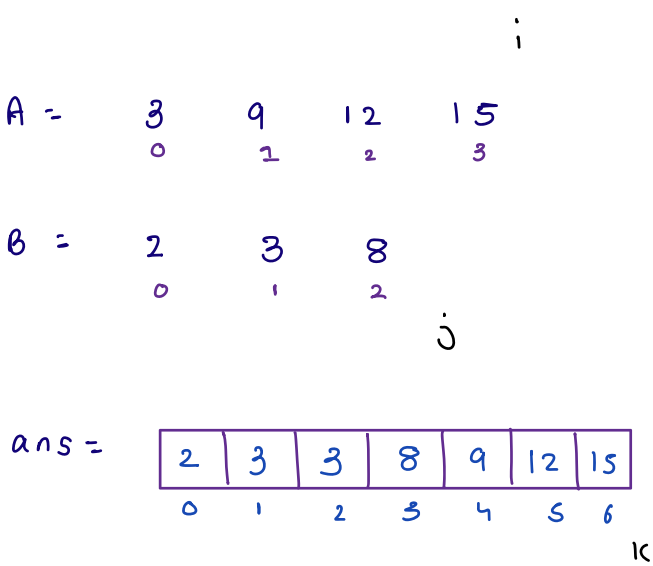
| 2 | 3 | 5 | 6 | 8 | 9 | 10 | 12 | 15 | 16 | 18 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

A = 3 5 9 12

B = 10 15 18

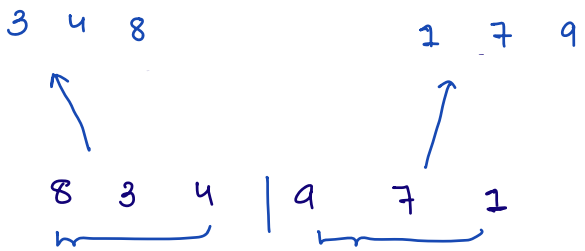



```
while(i < n && j < m) {
    if(A[i] < B[j]) {
        ans[k] = A[i];
        i++;
        k++;
    }
    else {
        ans[k] = B[j];
        j++;
        k++;
    }
}

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


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




TC: O(n+m)

Merge sort  TC: O(nlogn)



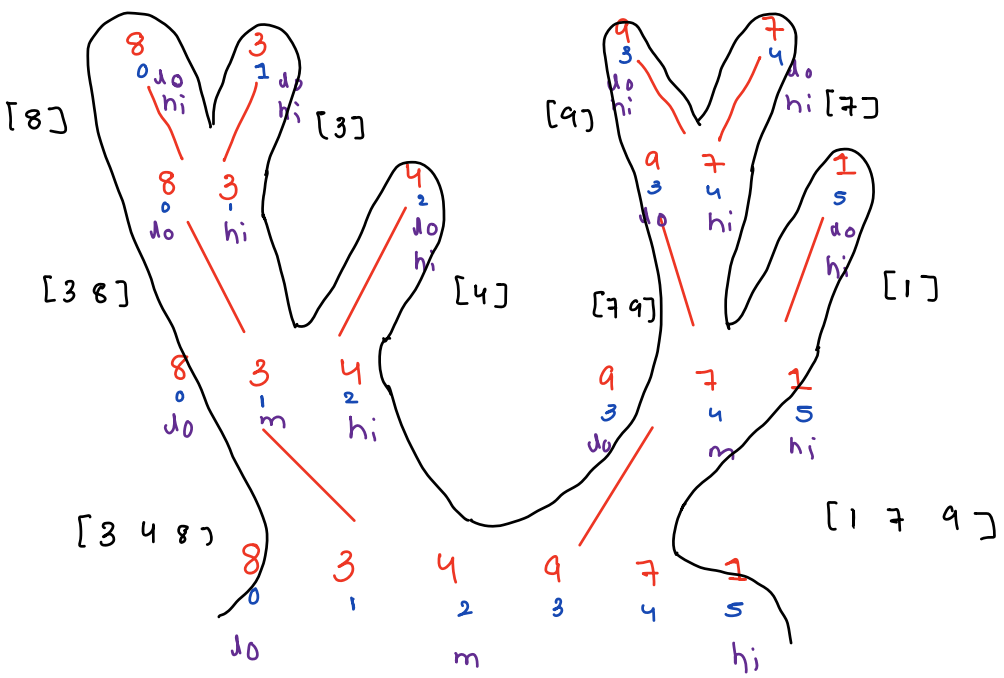

```
int[] mergesort (int[]arr, int lo, int hi) {

    int mid = (lo+hi)/2;

    int[]A = mergeSort(arr, lo, mid);

    int[]B = mergeSort(arr, mid+1, hi);

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

    return ans;

}
```

8 3 4 9 7 1

lo  mid  hi

```
int [] mergesort (int [] arr, int lo, int hi) {
    int mid = (lo+hi)/2;
    int [] A = mergeSort (arr, lo, mid);
    int [] B = mergeSort (arr, mid+1, hi);
    int [] ans = merge (A, B);
    return ans;
}
```

```
if (lo == hi) {
    int [] sa = new int [1];
    sa[0] = arr [lo];
    return sa;
}
```




[ 1 3 4 7 8 9 ]

todo :

merge sort ⇒

TC : O (nlogn)

SC : O (n)

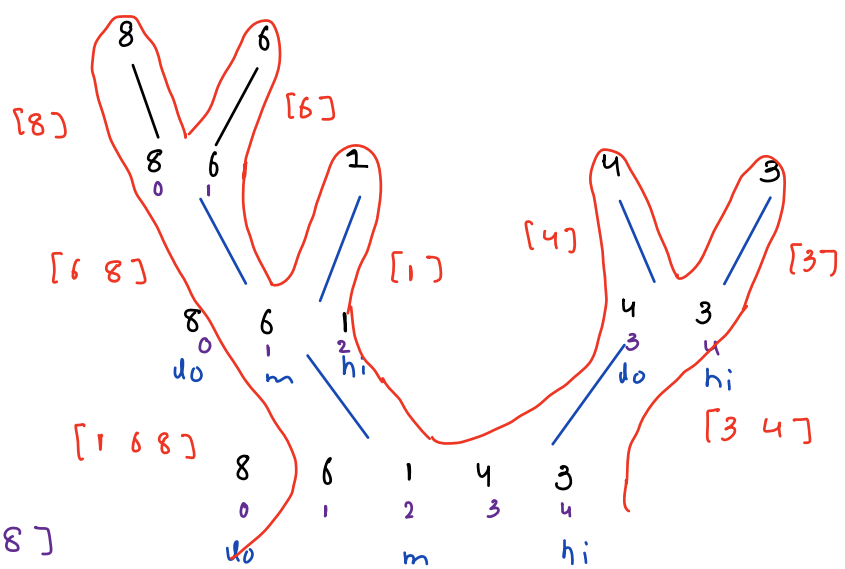
---

Doubts

=

```
int [] mergeSort (int [] arr, int lo, int hi) {
    int mid = (lo+hi)/2;
    int [] A = mergeSort (arr, lo, mid);
    int [] B = mergeSort (arr, mid+1, hi);
    int [] ans = merge (A, B);
    return ans;
}
```

```
if (lo == hi) {
    int [] sa = new int [1];
    sa [0] = arr [lo];
    return sa;
}
```




=> [1 3 4 6 8]

## Subarray with sum k

k = 9

| | -1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| | ↑ | 2 | 4 | 7 | 1 | -6 | 4 | 1 | 3 | 6 |
| Sum : | 0 | 2 | 6 | 13 | 14 | 8 | 12 | 13 | 16 | 22 |

| | |
|---|---|
| 0 → -1 | 8 → 4 |
| 2 → 0 | 12 → 5 |
| 6 → 1 | 16 → 7 |
| 13 → 2 | |
| 14 → 3 | |

Sum - k

22 - 9 = 13

map.get(13) = 2

map.containsKey (sum - k)

{ s = ~~-1~~ map.get(sum - k) + 1
{ e = ~~-1~~ i

Sum vs first index