Agenda

1) Bubble sort
2) Insertion sort

} Algorithms

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

What is sorting? Arranging the data

3 9 12 16 24 ⇒ inc. order

29 24 15 10 0 -2 ⇒ dec. order

7 6 8 12 ⇒ inc. order but based on no. of factors.

	7	6	8	12
# cf :	2	4	4	6

How to sort in Java

```
int[] A = {3, 4, 1, 6};

Arrays.sort(A);        { inc. order }
```

$\hookrightarrow$ array variable name

// A ⇒ { 1 3 4 6 }

Arrays.sort(A) ⇒ TC : O(n log n)

Bubble sort

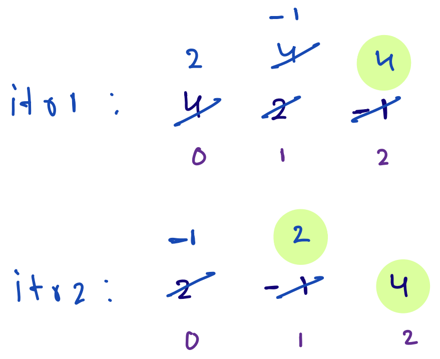
	3	8	6	2	4
	0	1	2	3	4
itr 1 :	3	6	2	4	8
itr 2 :	3	2	4	6	8
itr 3 :	3	2	4	6	8
itr 4 :	2	3	4	6	8

itr : n-1

	3	8	6	2	4
	0	1	2	3	4
itr 1 :	3	$\overset{6}{\cancel{8}}$	$\overset{\overset{2}{\cancel{8}}}{\cancel{6}}$	$\overset{\overset{4}{\cancel{8}}}{\cancel{2}}$	$\overset{8}{\cancel{4}}$
	0	1	2	3	4
itr 2 :	3	$\overset{2}{\cancel{6}}$	$\overset{\overset{4}{\cancel{6}}}{\cancel{2}}$	$\overset{6}{\cancel{4}}$	8
	0	1	2	3	4
itr 3 :	$\overset{2}{\cancel{3}}$	$\overset{3}{\cancel{2}}$	4	6	8
	0	1	2	3	4
itr 4 :	2	3	4	6	8
	0	1	2	3	4

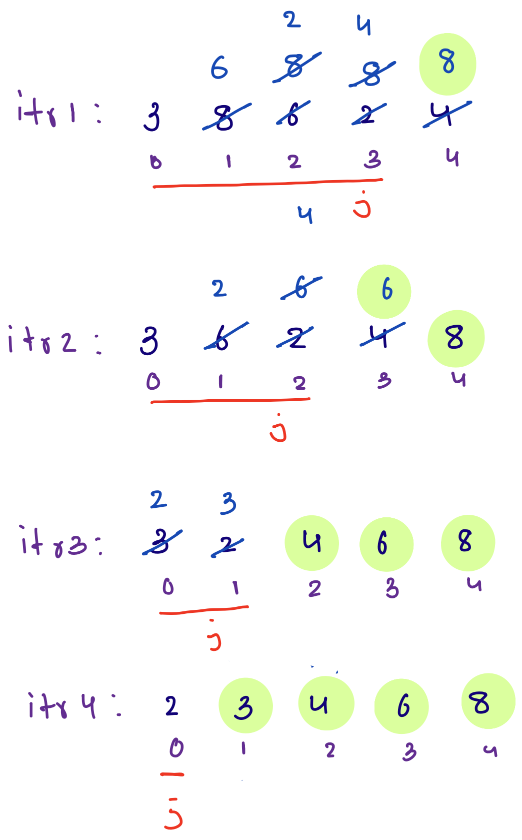
```
if (A[j] > A[j+1]) {
        swap;
3
```

$$A = \begin{array}{ccc} 4 & 2 & -1 \\ 0 & 1 & 2 \end{array}$$



code

$$\begin{array}{ccccc} 3 & 8 & 6 & 2 & 4 \\ 0 & 1 & 2 & 3 & 4 \end{array}$$



iterations = n-1

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: $O(1)$

A = 8 6 1 3
    0 1 2 3

```
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;
            }
        }
    }
}
```

i	j				
0	0, 1, 2	~~8~~ 6	~~6~~ ~~8~~ 1	~~1~~ ~~8~~ 3	~~3~~ 8
		0	1	2	3
1	0, 1	~~6~~ 1	~~1~~ ~~6~~ 3	~~3~~ 6	8
		0	1	2	3
2	0	1	3	6	8
		0	1	2	3

Insertion sort

$$A = \begin{array}{ccccc} 3 & 8 & 6 & 2 & 4 \\ 0 & 1 & 2 & 3 & 4 \end{array}$$

itr1:
$$\begin{tikzcd} 3\,] & 8 \arrow[l, bend left] & 6 & 2 & 4 \end{tikzcd}$$

itr2:
$$\begin{tikzcd} 3 & 8\,] & 6 \arrow[l, bend left] & 2 & 4 \end{tikzcd}$$

itr3:
$$\begin{tikzcd} 3 & 6 & 8\,] & 2 \arrow[l, bend left] & 4 \end{tikzcd}$$

itr4:
$$\begin{tikzcd} 2 & 3 & 6 & 8\,] & 4 \arrow[l, bend left] \end{tikzcd}$$

$$2 \quad 3 \quad 4 \quad 6 \quad 8\,]$$

A = 3 8 6 2 4
0 1 2 3 4

✓
itr1 : 3] 8 6 2 4
0 1 2 3 4

6 8 ✓
itr2 : 3 ~~8~~] ~~6~~ 2 4
0 1 2 3 4

3 6
2 ~~2~~ ~~2~~ 8 ✓
itr3 : ~~3~~ ~~6~~ ~~8~~] ~~2~~ 4
0 1 2 3 4

4 ~~4~~ 6 8 ✓
itr4 : 2 3 ~~6~~ ~~8~~] ~~4~~
0 1 2 3 4

2 3 4 6 8

if (A[j] > A[j+1]) {
swap
3

$\curvearrowright$ i-1 to r

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 : $O(1)$

Dry run

```
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 ;
    |     |     }
    |     }
    }
}
```

$A = \underset{0}{7} \quad \underset{1}{4} \quad \underset{2}{2} \quad \underset{3}{5}$

i	j	
1	0	$\underset{0}{\overset{4}{\cancel{7}}} \mid \underset{1}{\overset{7\checkmark}{\cancel{4}}} \quad \underset{2}{2} \quad \underset{3}{5}$
2	1, 0	$\underset{0}{\overset{2}{\cancel{4}}} \quad \underset{1}{\overset{\cancel{2}4}{\cancel{7}}} \mid \underset{2}{\overset{7\checkmark}{\cancel{2}}} \quad \underset{3}{5}$
3	2, 1, $\cancel{0}$	$\underset{0}{2} \quad \underset{1}{4} \quad \underset{2}{\overset{5}{\cancel{7}}} \mid \underset{3}{\overset{7\checkmark}{\cancel{5}}}$

**

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	
	0	1	2	3	4	
B ⇒	3	6	8	10	16	18
	0	1	2	3	4	5

ans:

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

i

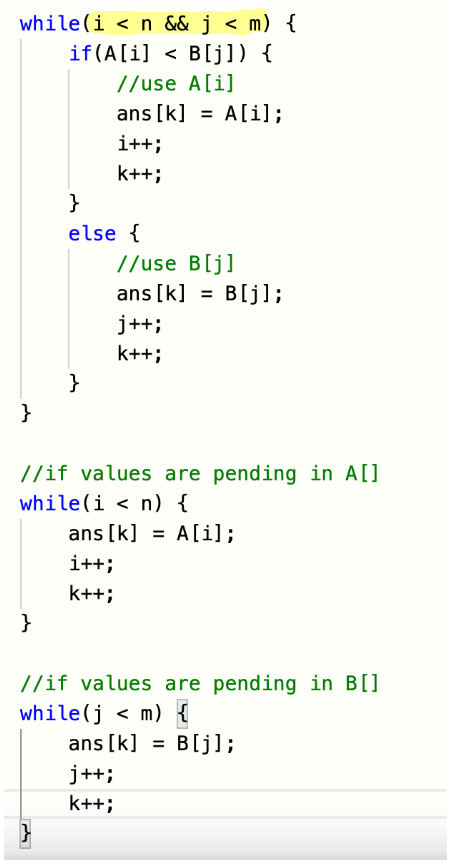
A =	3	9	15	20	25
	0	1	2	3	4

B =	8	12	15
	0	1	2

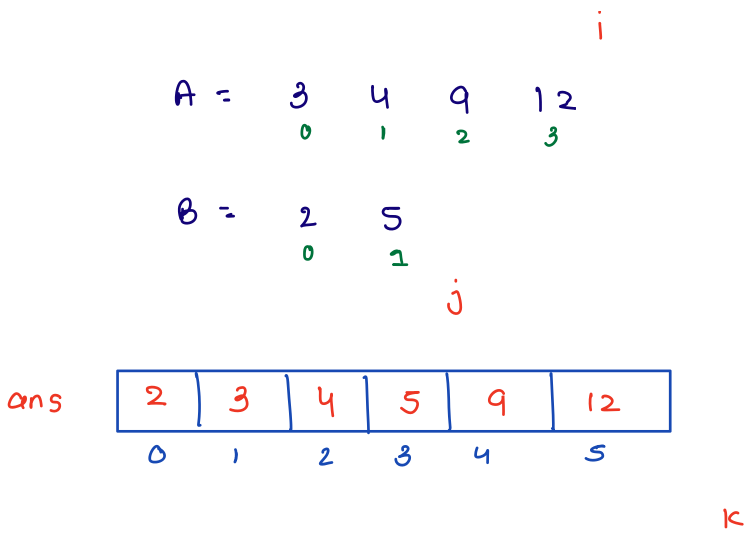
j

ans	3	8	9	12	15	15	20	25
	0	1	2	3	4	5	6	7

k

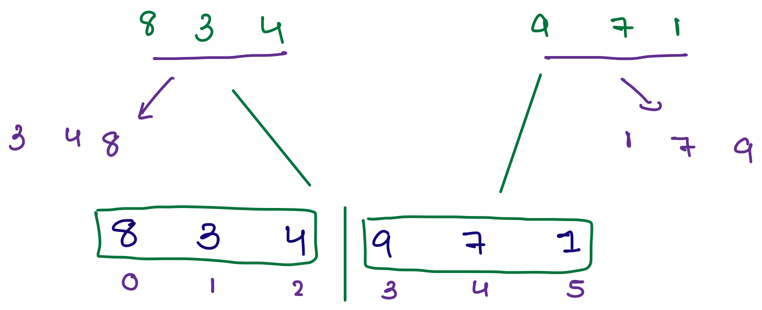


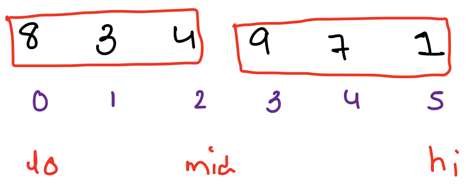
```
while(i < n && j < m) {
    if(A[i] < B[j]) {
        //use A[i]
        ans[k] = A[i];
        i++;
        k++;
    }
    else {
        //use B[j]
        ans[k] = B[j];
        j++;
        k++;
    }
}

//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++;
}
```



Merge sort   TC: O(nlogn)

↳ divide and conquer



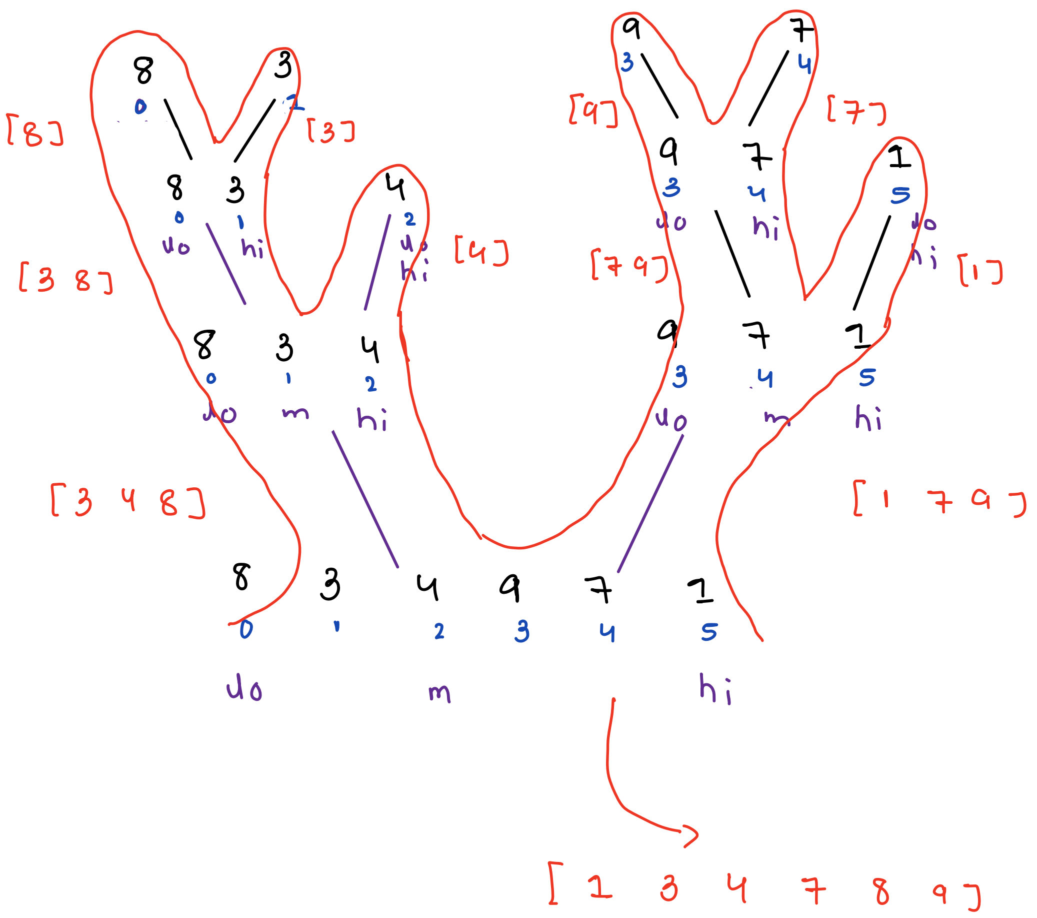


$$mid = \frac{(lo + hi)}{2}$$

(lo, mid)

(mid+1, hi)

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

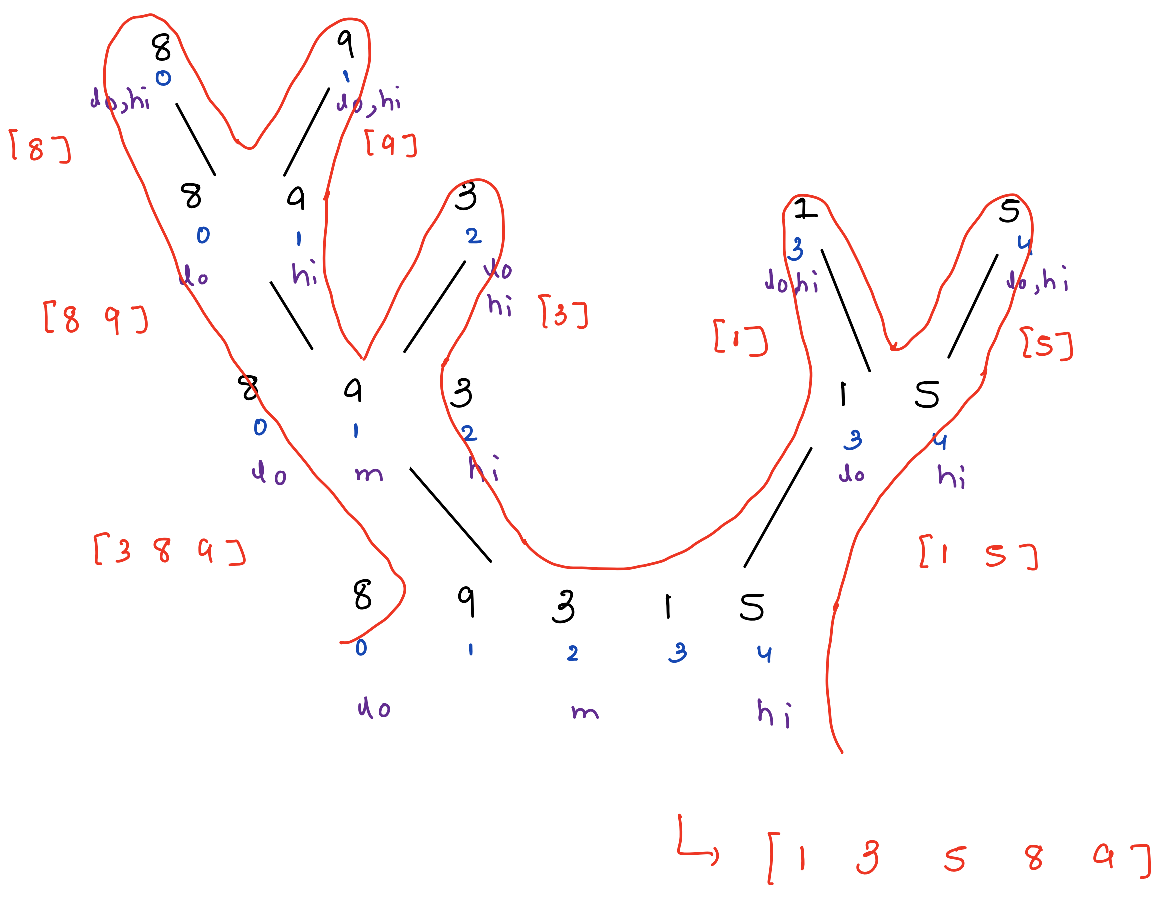


todo: understand complexities

TC: O(nlogn)

SC: O(n)

```
int[] mergeSort (int[] arr, int lo, int hi )
                                          if (lo == hi) {
    int mid = (lo+hi) / 2 ;                   int [] sa = new int [1] ;
    int[] A = mergeSort (arr, lo, mid);       sa[0] = arr [lo] ;
    int[] B = mergeSort (arr, mid+1, hi);     return sa;
    int [] ans = merge (A, B);             }
    return ans;
}
```



8 0 lo,hi [8]
9 1 lo,hi [9]
8 0 lo 9 1 hi [8 9]
3 2 lo hi [3]
8 0 lo 9 1 m 3 2 hi [3 8 9]
1 3 lo,hi [1]
5 4 lo,hi [5]
1 3 lo 5 4 hi [1 5]
8 0 lo 9 1 3 2 m 1 3 5 4 hi
[1 3 5 8 9]