Selection & Merge Sorts

Kuan-Yu Chen (陳冠宇)

Sorting

- Sorting means arranging the elements of an array so that they are placed in some relevant order which may be either **ascending** or **descending**

- A sorting algorithm is defined as an algorithm that puts the elements of a list in a certain order, which can be either **numerical** order, **lexicographical** order, or **any user-defined** order
  - **Bubble, Insertion**, Selection, **Tree**
  - Merge, Quick, Radix, Heap, Shell

Selection Sort.

- Selection sort is also a simple algorithm for sorting

- The procedure of the selection sort
  - Consider an array with N elements
    - First find the smallest value in the array and place it in the first position
    - Then, find the second smallest value in the array and place it in the second position
    - Repeat this procedure until the entire array is sorted

Example

- Please sort a given data array by using selection sort

39	9	81	45	90	27	72	18

PASS	ARR[0]	ARR[1]	ARR[2]	ARR[3]	ARR[4]	ARR[5]	ARR[6]	ARR[7]
1	9	39	81	45	90	27	72	18
2	9	18	81	45	90	27	72	39
3	9	18	27	45	90	81	72	39
4	9	18	27	39	90	81	72	45
5	9	18	27	39	45	81	72	90
6	9	18	27	39	45	72	81	90
7	9	18	27	39	45	72	81	90

Selection Sort..

```
SMALLEST (ARR, K, N, POS)

Step 1: [INITIALIZE] SET SMALL = ARR[K]
Step 2: [INITIALIZE] SET POS = K
Step 3: Repeat for J = K+1 to N-1
            IF SMALL > ARR[J]
                  SET SMALL = ARR[J]
                  SET POS = J
            [END OF IF]
        [END OF LOOP]
Step 4: RETURN POS
```

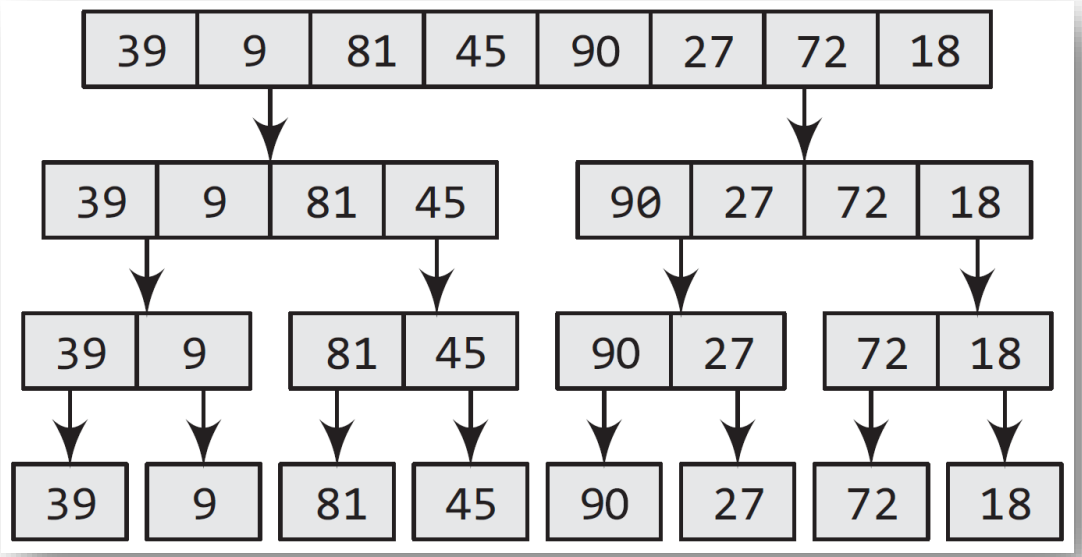
```
SELECTION SORT(ARR, N)

Step 1: Repeat Steps 2 and 3 for K = 0
        to N-1
Step 2:     CALL SMALLEST(ARR, K, N, POS)
Step 3:     SWAP A[K] with ARR[POS]
      [END OF LOOP]
Step 4: EXIT
```

39	9	81	45	90	27	72	18

PASS	ARR[0]	ARR[1]	ARR[2]	ARR[3]	ARR[4]	ARR[5]	ARR[6]	ARR[7]
1	9	39	81	45	90	27	72	18
2	9	18	81	45	90	27	72	39
3	9	18	27	45	90	81	72	39
4	9	18	27	39	90	81	72	45
5	9	18	27	39	45	81	72	90
6	9	18	27	39	45	72	81	90
7	9	18	27	39	45	72	81	90

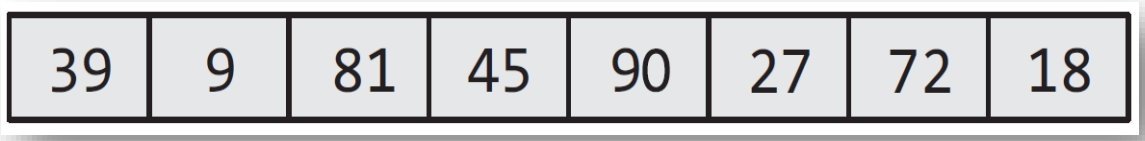
Merge Sort.

- Merge sort is a sorting algorithm that uses the **divide**, **conquer**, and **combine** algorithmic paradigm
  - ***Divide*** means partitioning the n-element array to be sorted into two sub-arrays
  - ***Conquer*** means sorting the two sub-arrays recursively
  - ***Combine*** means merging the two sorted sub-arrays

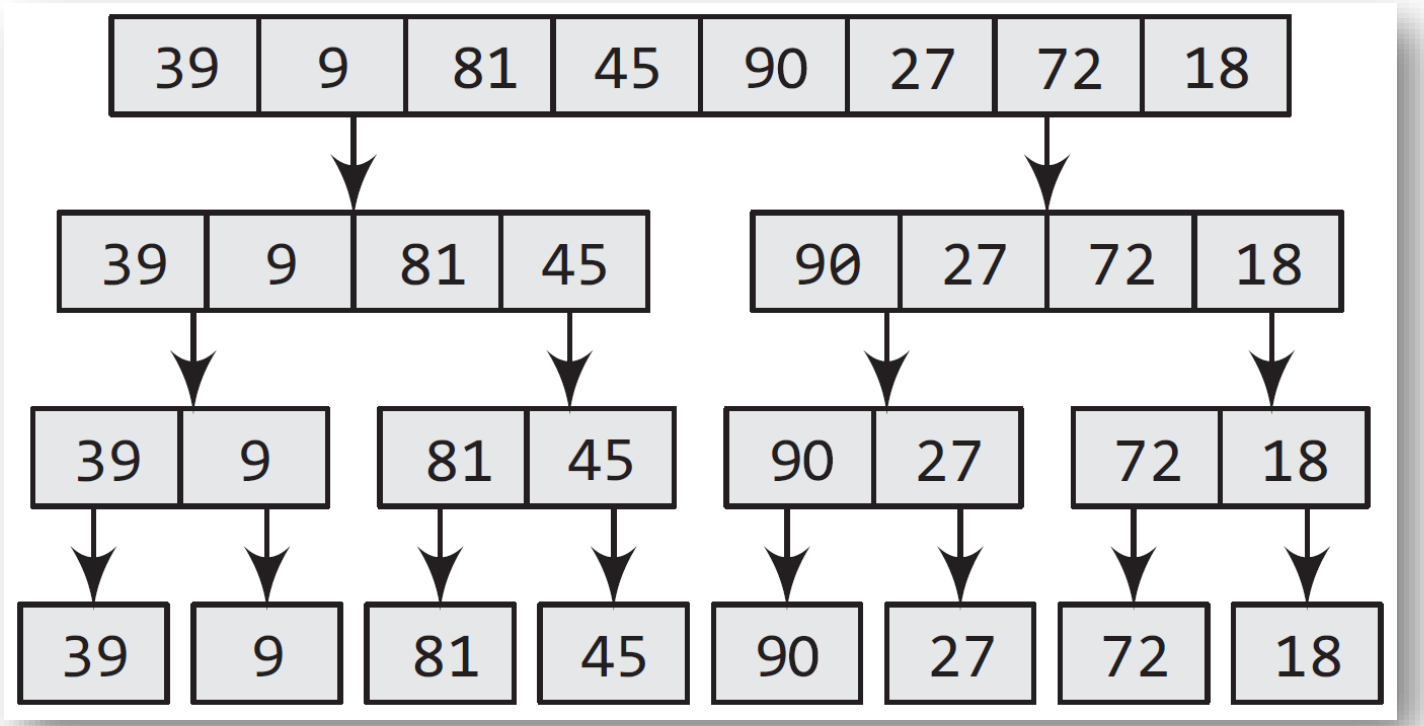


Example.

- Sort the given array using merge sort

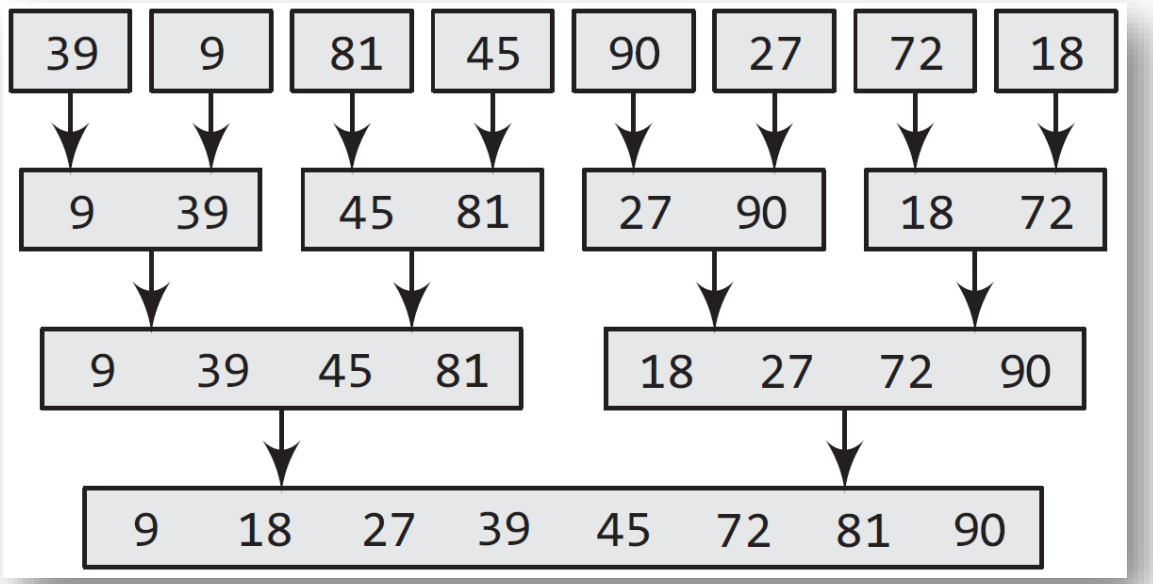


  - Divide and Conquer



Example..

- Conquer and Combine

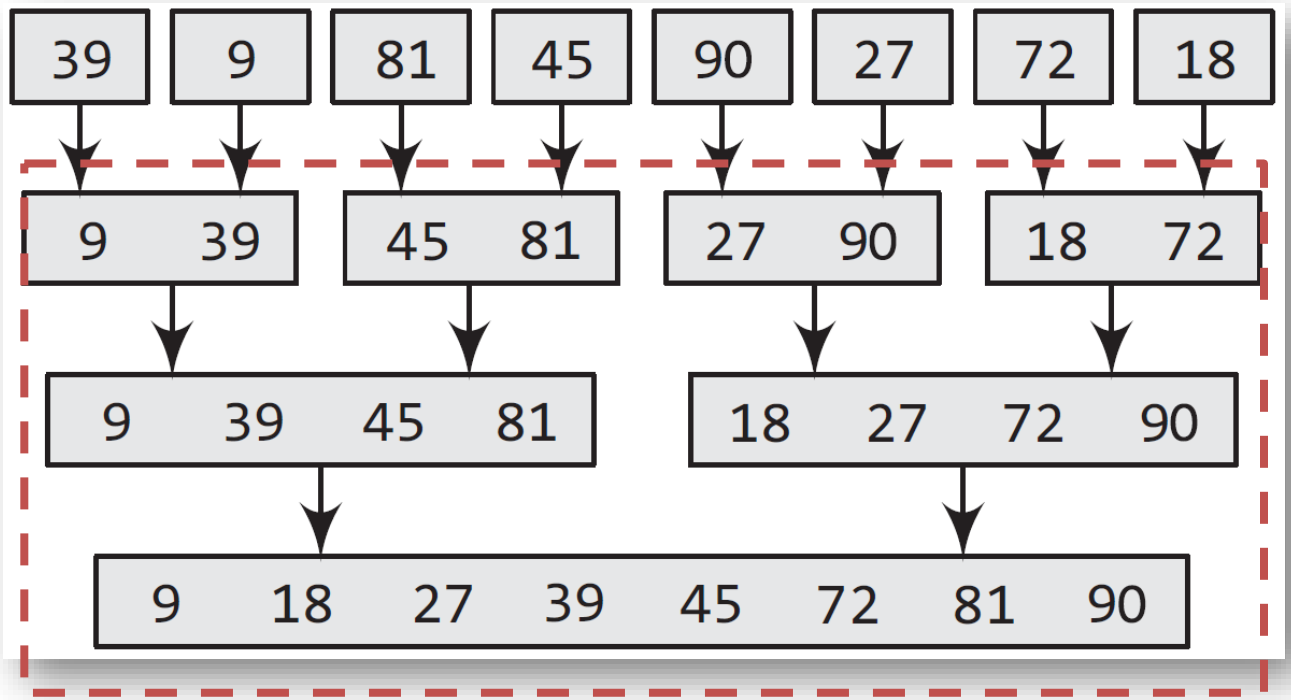


Merge Sort..

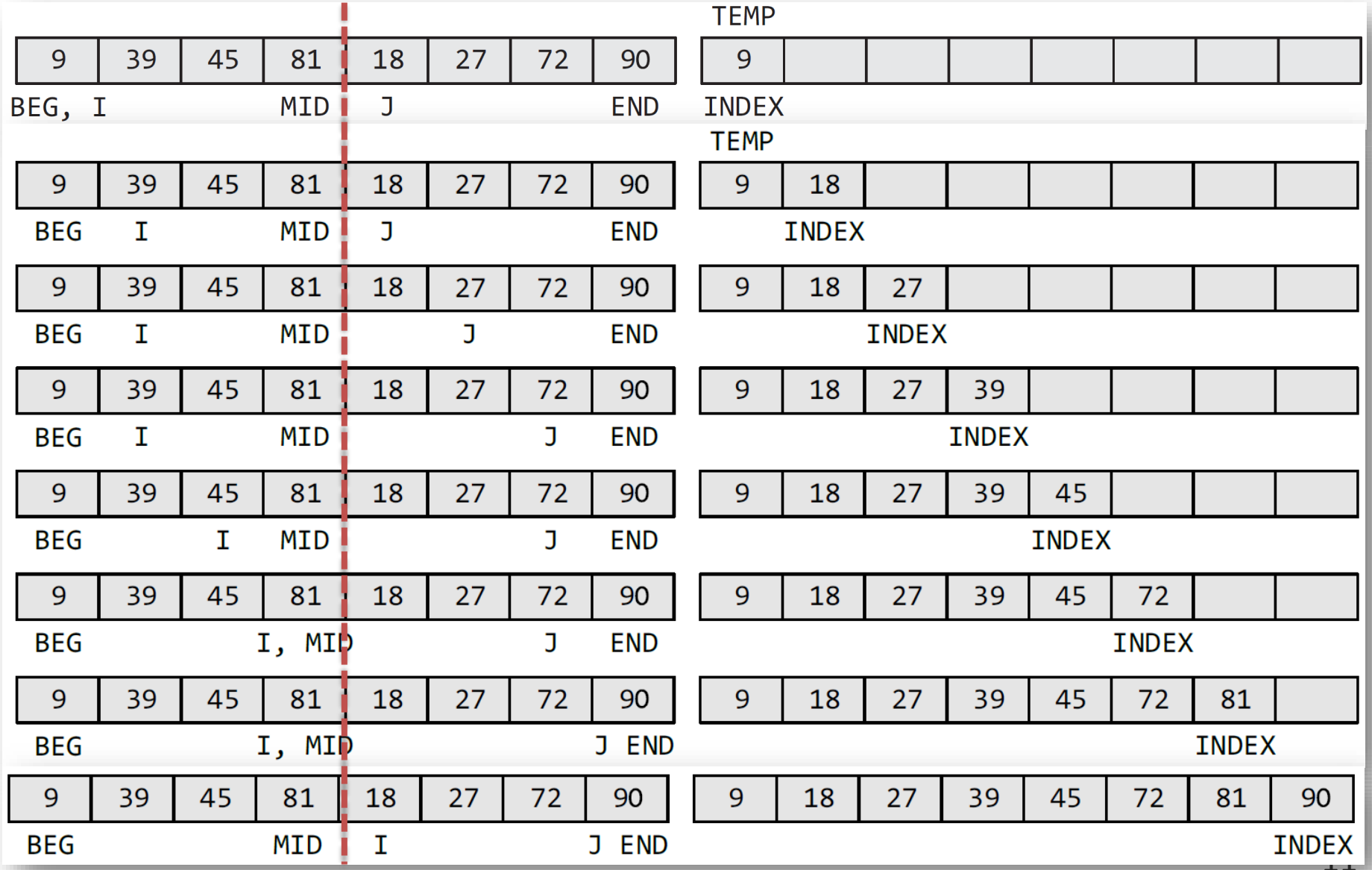
```
MERGE_SORT(ARR, BEG, END)
Step 1: IF BEG < END
            SET MID = (BEG + END)/2
            CALL MERGE_SORT (ARR, BEG, MID)
            CALL MERGE_SORT (ARR, MID + 1, END)
            MERGE (ARR, BEG, MID, END)
        [END OF IF]
Step 2: END
```

Merge Sort…

- The concept of the merge function is to compare two sub-arrays (ARR[I] and ARR[J]), the smaller of the two is placed in a temp array (TEMP) at the location specified by a index (INDEX) and subsequently the index value (I or J) is incremented
  - Example for the merge function



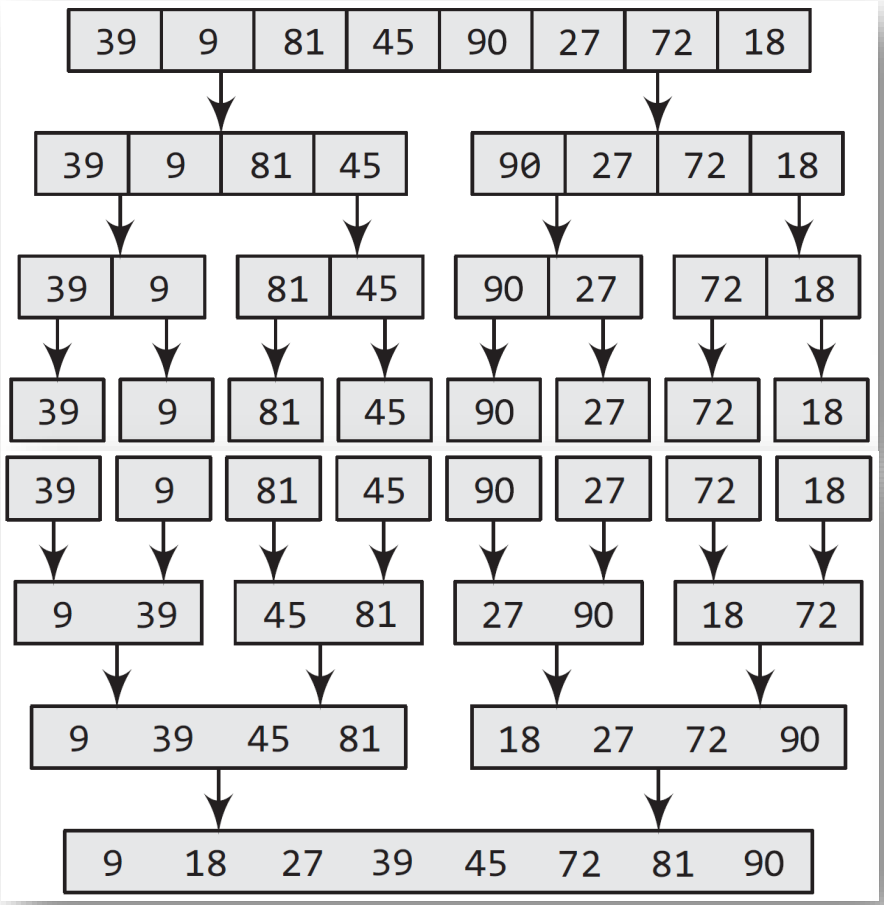
Merge Sort….


TEMP
9 39 45 81 18 27 72 90
9
BEG, I MID J END
INDEX
TEMP
9 39 45 81 18 27 72 90
9 18
BEG I MID J END
INDEX
9 39 45 81 18 27 72 90
9 18 27
BEG I MID J END
INDEX
9 39 45 81 18 27 72 90
9 18 27 39
BEG I MID J END
INDEX
9 39 45 81 18 27 72 90
9 18 27 39 45
BEG I MID J END
INDEX
9 39 45 81 18 27 72 90
9 18 27 39 45 72
BEG I, MID J END
INDEX
9 39 45 81 18 27 72 90
9 18 27 39 45 72 81
BEG I, MID J END
INDEX
9 39 45 81 18 27 72 90
9 18 27 39 45 72 81 90
BEG MID I J END
INDEX

Merge Sort…..

```
MERGE (ARR, BEG, MID, END)

Step 1: [INITIALIZE] SET I = BEG, J = MID + 1, INDEX = O
Step 2: Repeat while (I <= MID) AND (J<=END)
            IF ARR[I] < ARR[J]
                  SET TEMP[INDEX] = ARR[I]
                  SET I = I + 1
            ELSE
                  SET TEMP[INDEX] = ARR[J]
                  SET J = J + 1
            [END OF IF]
            SET INDEX = INDEX + 1
      [END OF LOOP]
```

Merge Sort……




```
MERGE_SORT(ARR, BEG, END)
Step 1: IF BEG < END
            SET MID = (BEG + END)/2
            CALL MERGE_SORT (ARR, BEG, MID)
            CALL MERGE_SORT (ARR, MID + 1, END)
            MERGE (ARR, BEG, MID, END)
        [END OF IF]
Step 2: END
```

```
MERGE (ARR, BEG, MID, END)

Step 1: [INITIALIZE] SET I = BEG, J = MID + 1, INDEX = 0
Step 2: Repeat while (I <= MID) AND (J<=END)
            IF ARR[I] < ARR[J]
                  SET TEMP[INDEX] = ARR[I]
                  SET I = I + 1
            ELSE
                  SET TEMP[INDEX] = ARR[J]
                  SET J = J + 1
            [END OF IF]
            SET INDEX = INDEX + 1
      [END OF LOOP]
```

Break for Rocling 2022

Questions?

kychen@mail.ntust.edu.tw