Merge, Quick & Radix Sorts

Kuan-Yu Chen (陳冠宇)

2018/11/12 @ TR-212, NTUST

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

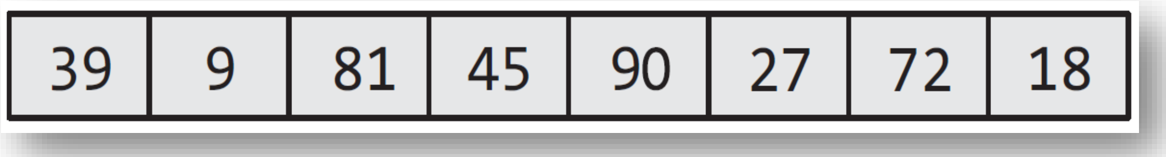
- A sorting algorithm is defined as an algorithm that puts the elements of a list in a certain order, which can be either numerical order, lexicographical order, or any user-defined order
  - **Bubble, Insertion, Selection, Tree**
  - Merge, Quick, Radix, Heap, Shell

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

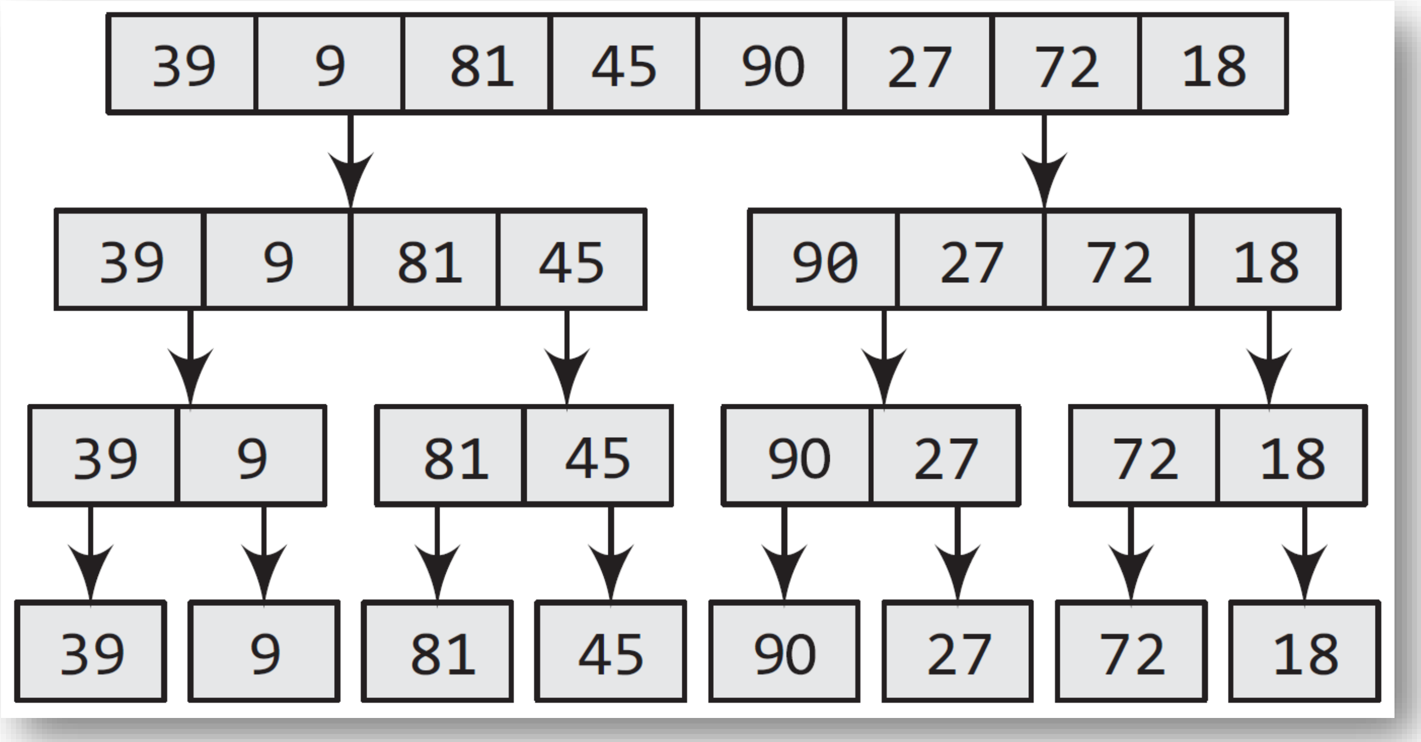
Example.

- Sort the given array using merge sort



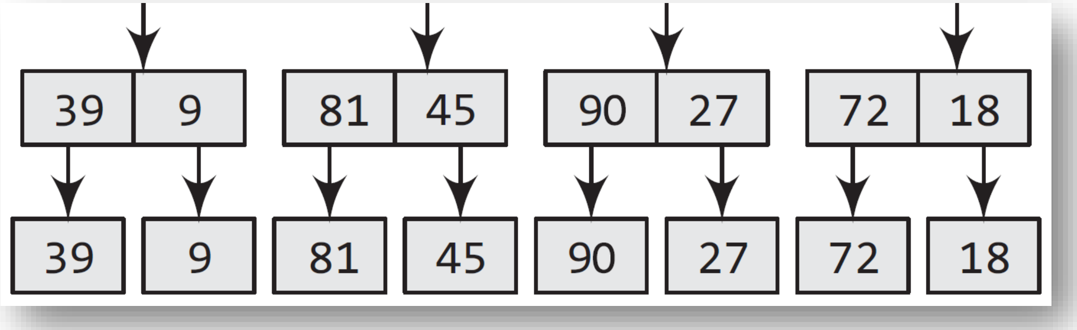
39	9	81	45	90	27	72	18

  - Divide and Conquer



Example..

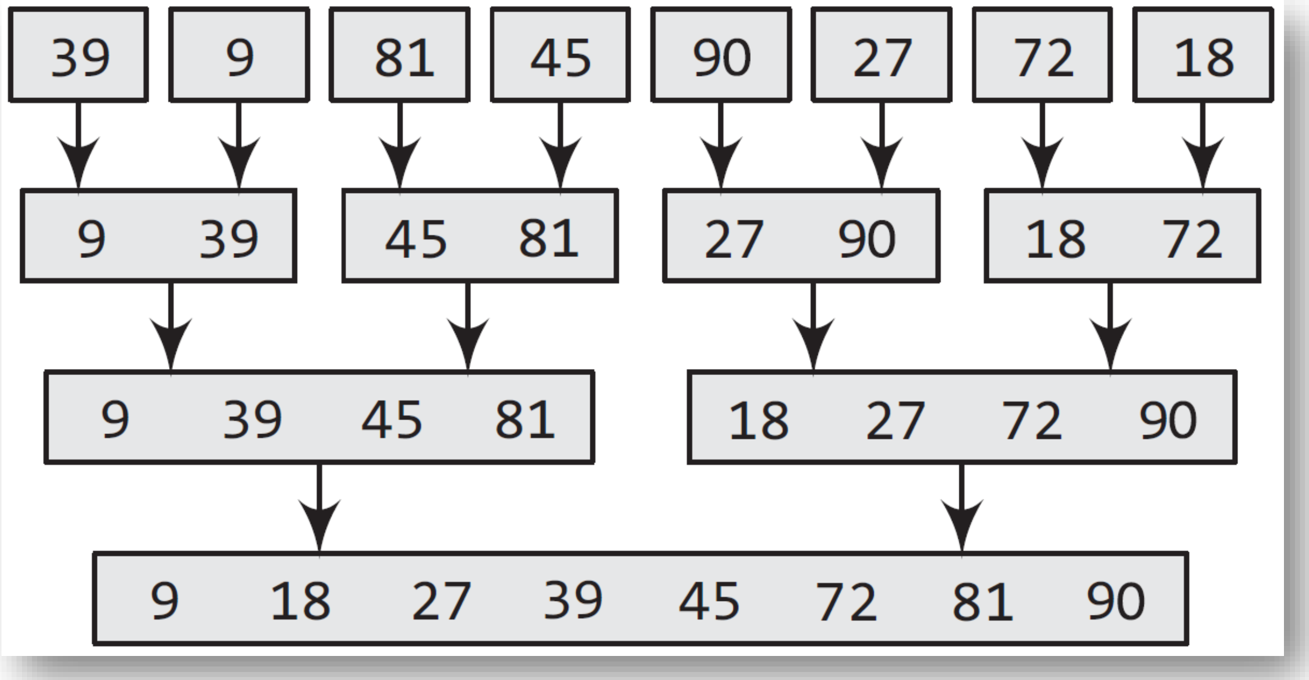
- Divide and Conquer



39 9 | 81 45 | 90 27 | 72 18

39 | 9 | 81 | 45 | 90 | 27 | 72 | 18

- Combine



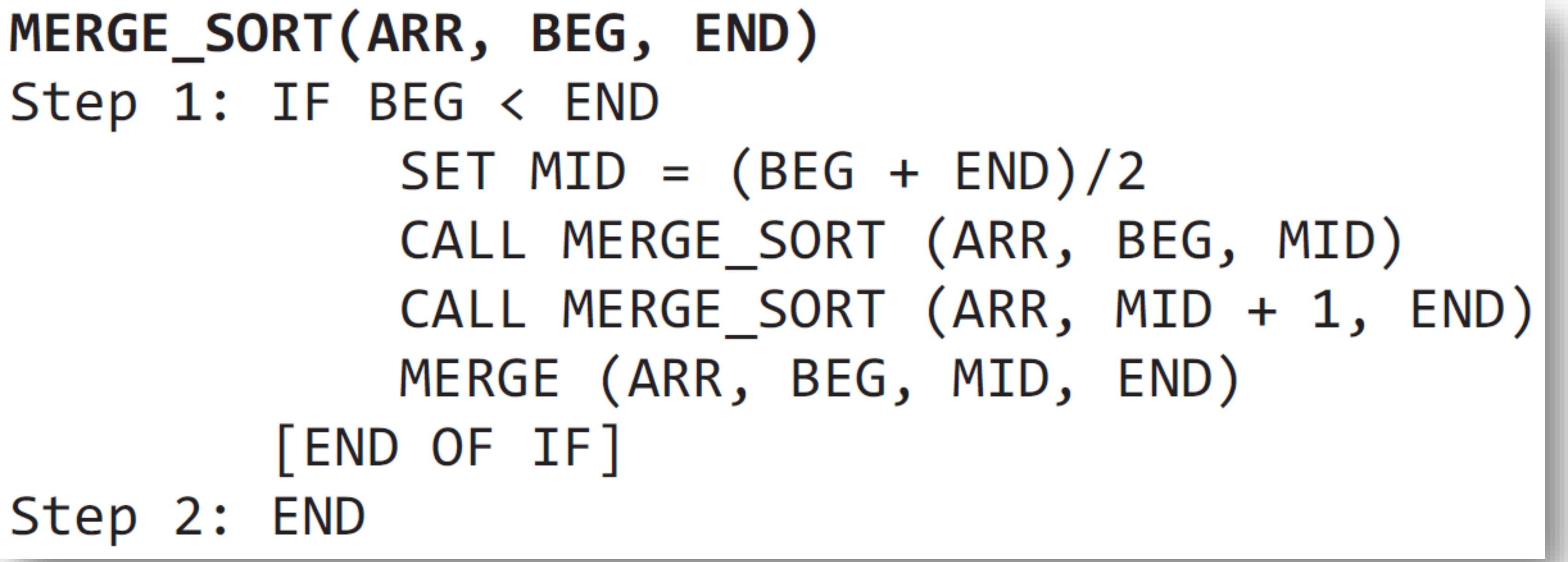
39 | 9 | 81 | 45 | 90 | 27 | 72 | 18

9 39 | 45 81 | 27 90 | 18 72

9 39 45 81 | 18 27 72 90

9 18 27 39 45 72 81 90

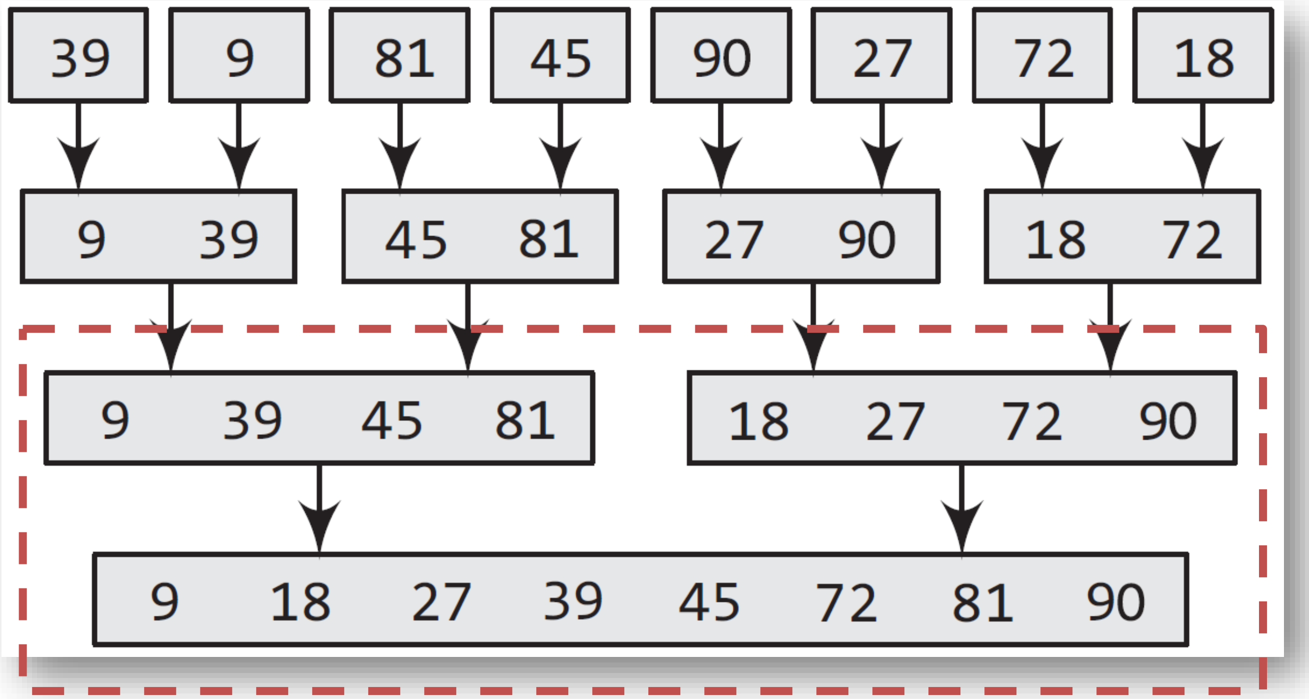
Merge Sort..



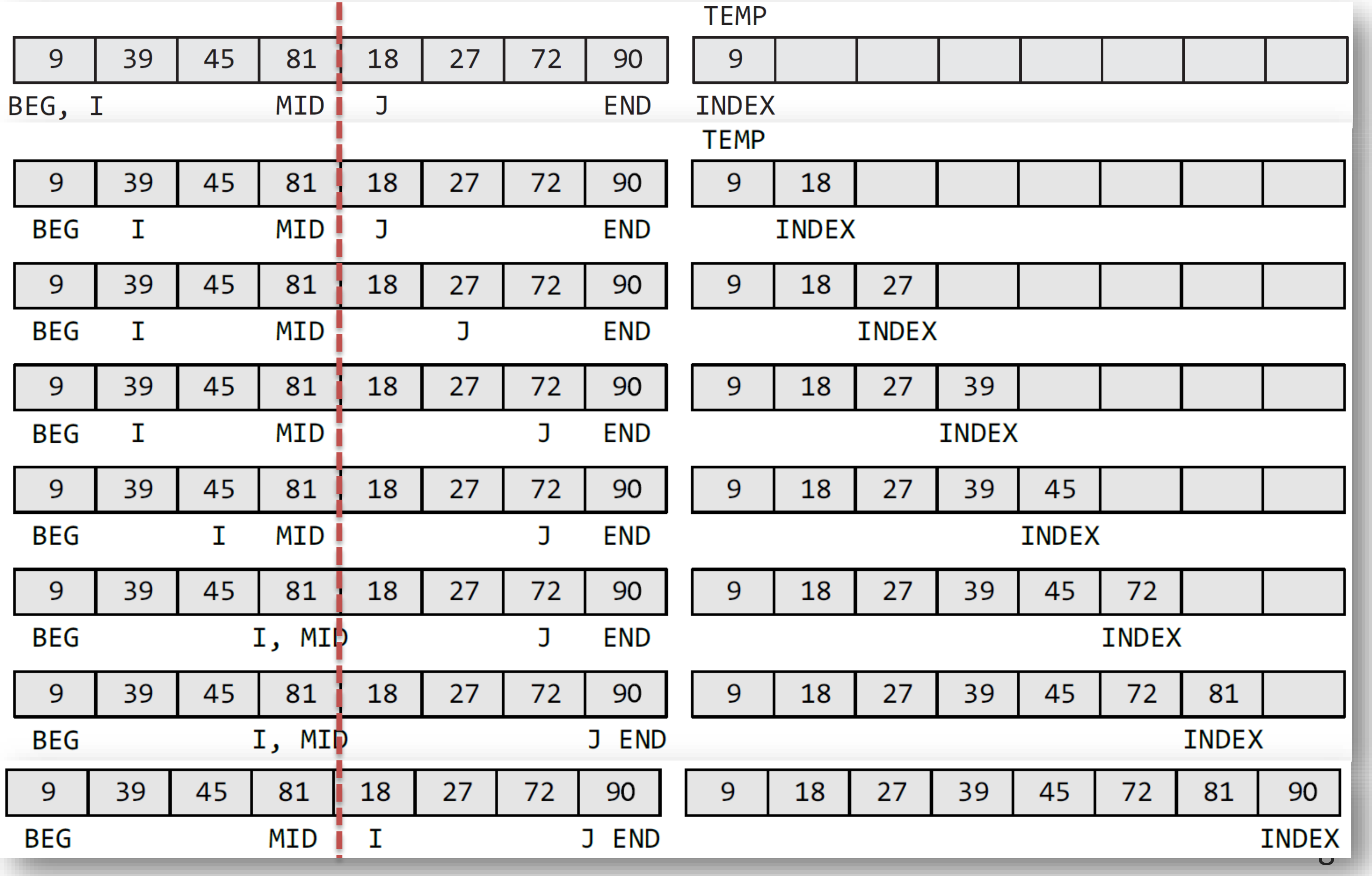
```
MERGE_SORT(ARR, BEG, END)
Step 1: IF BEG < END
            SET MID = (BEG + END)/2
            CALL MERGE_SORT (ARR, BEG, MID)
            CALL MERGE_SORT (ARR, MID + 1, END)
            MERGE (ARR, BEG, MID, END)
        [END OF IF]
Step 2: END
```

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….



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

TEMP

9							
INDEX							

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

TEMP

9	18						
	INDEX						

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

9	18	27					
		INDEX					

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

9	18	27	39				
			INDEX				

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

9	18	27	39	45			
				INDEX			

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

9	18	27	39	45	72		
					INDEX		

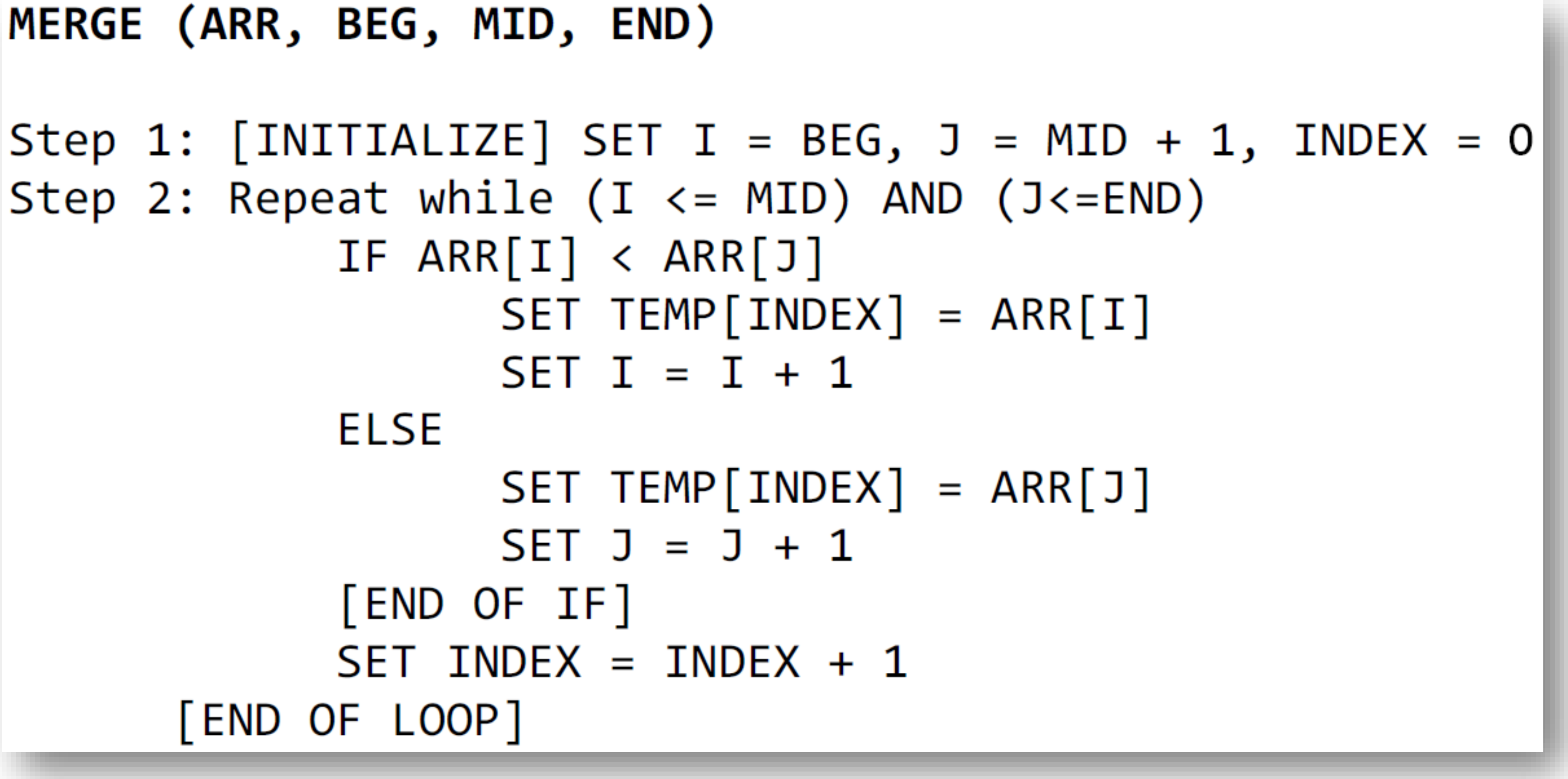
9	39	45	81	18	27	72	90
BEG			I, MID				J END

9	18	27	39	45	72	81	
						INDEX	

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

9	18	27	39	45	72	81	90
							INDEX

Merge Sort.....



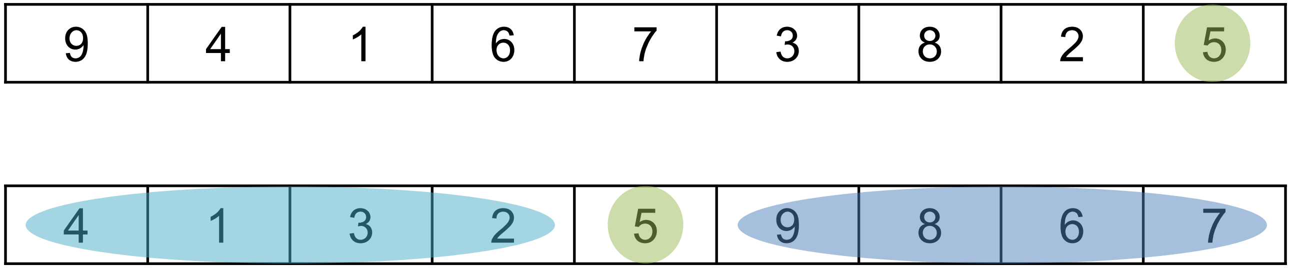
```
MERGE (ARR, BEG, MID, END)

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]
```

Quick Sort.

- Quick sort is a widely used sorting algorithm developed by C. A. R. Hoare
  - Quick sort is also known as partition exchange sort

- The quick sort algorithm works as follows:
  1. Select an element **pivot** from the array elements
  2. Rearrange the elements in the array in such a way that all elements that are less than the pivot appear before the pivot and all elements greater than the pivot element come after it
  3. Recursively sort the two sub-arrays thus obtained

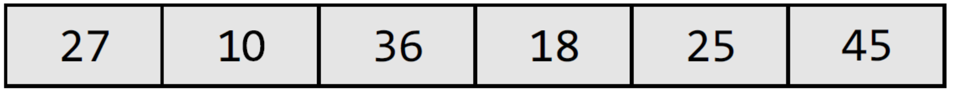


9	4	1	6	7	3	8	2	5

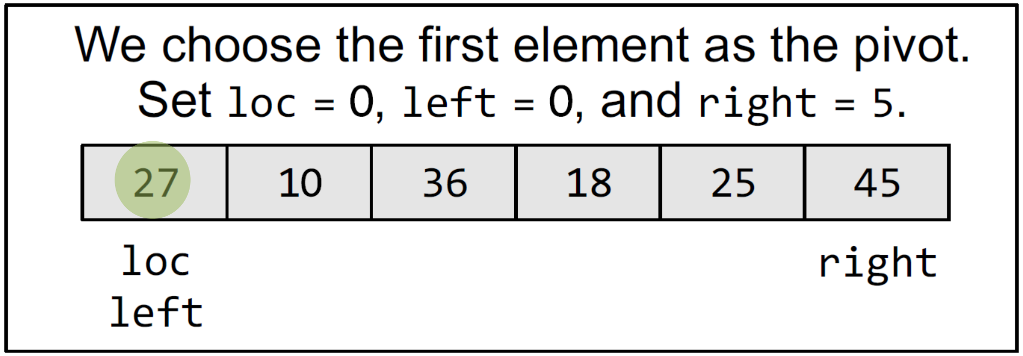
4	1	3	2	5	9	8	6	7

Example.

- Sort the given array using quick sort algorithm

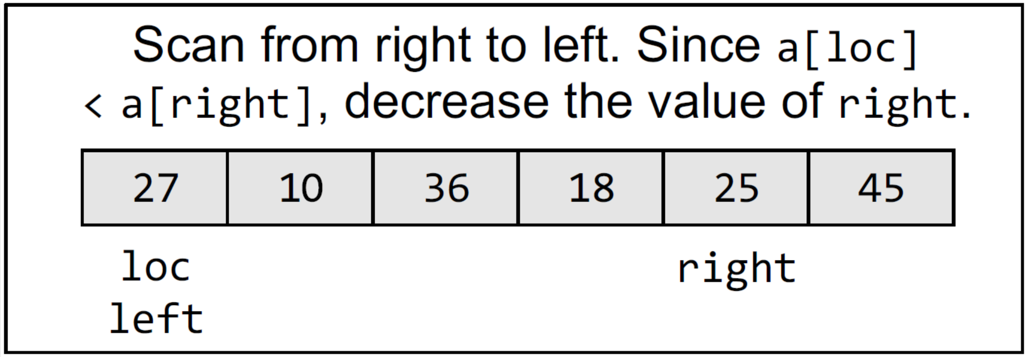


27	10	36	18	25	45



We choose the first element as the pivot. Set loc = 0, left = 0, and right = 5.

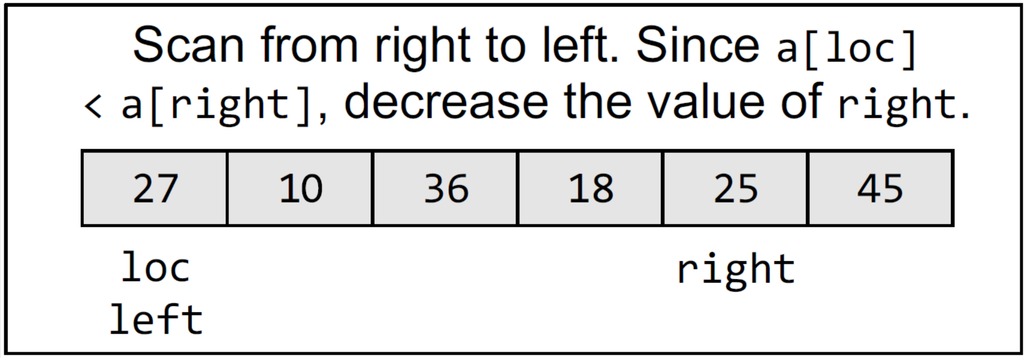
27	10	36	18	25	45



Scan from right to left. Since a[loc] < a[right], decrease the value of right.

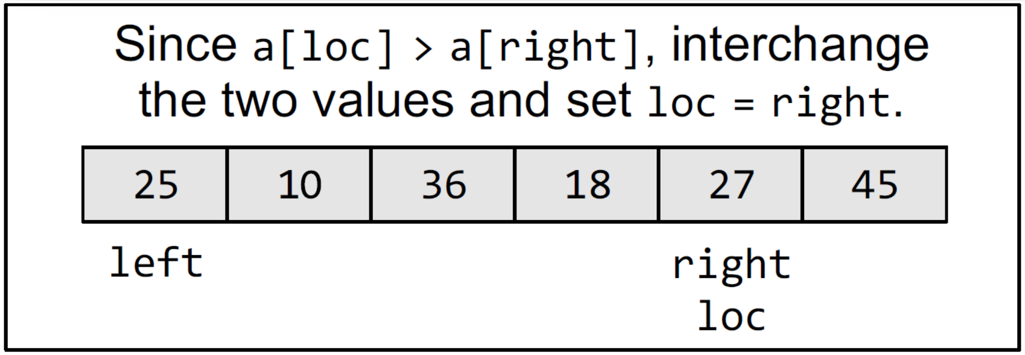
27	10	36	18	25	45

Example..



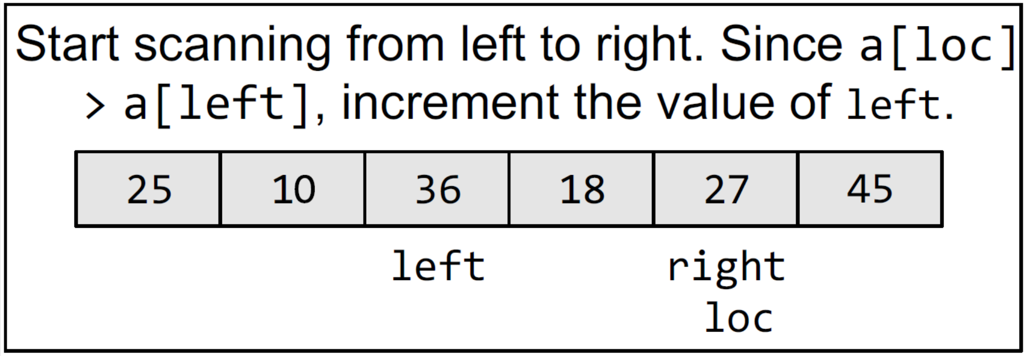
Scan from right to left. Since a[loc] < a[right], decrease the value of right.

27	10	36	18	25	45
loc left				right	



Since a[loc] > a[right], interchange the two values and set loc = right.

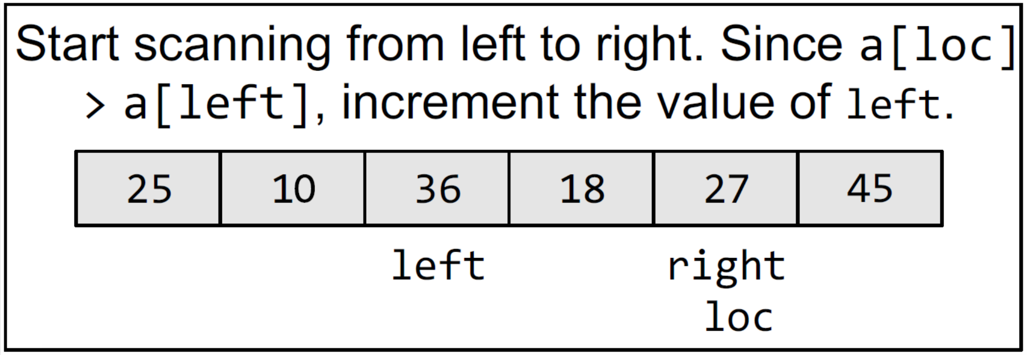
25	10	36	18	27	45
left				right loc	



Start scanning from left to right. Since a[loc] > a[left], increment the value of left.

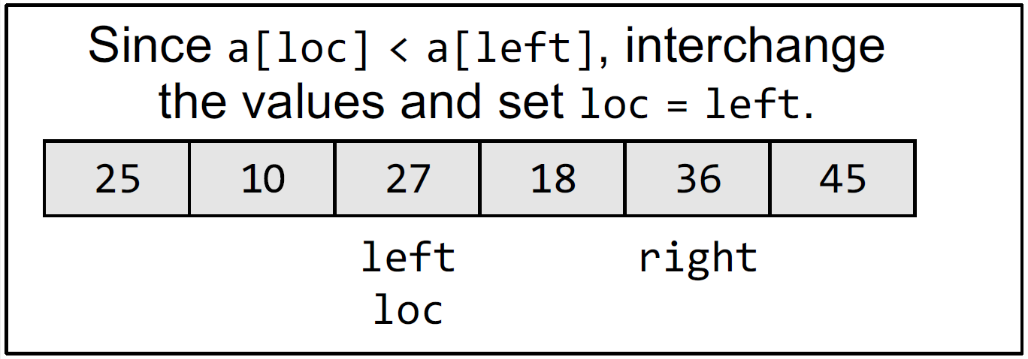
25	10	36	18	27	45
		left		right loc	

Example…



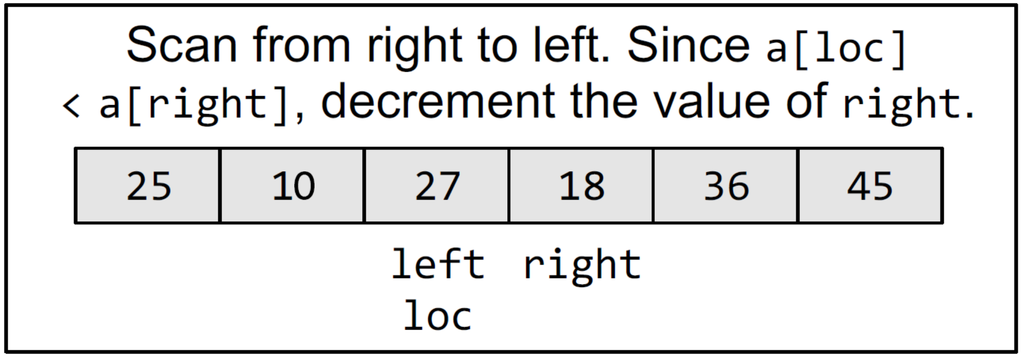
Start scanning from left to right. Since a[loc] > a[left], increment the value of left.

25	10	36	18	27	45
		left		right loc	



Since a[loc] < a[left], interchange the values and set loc = left.

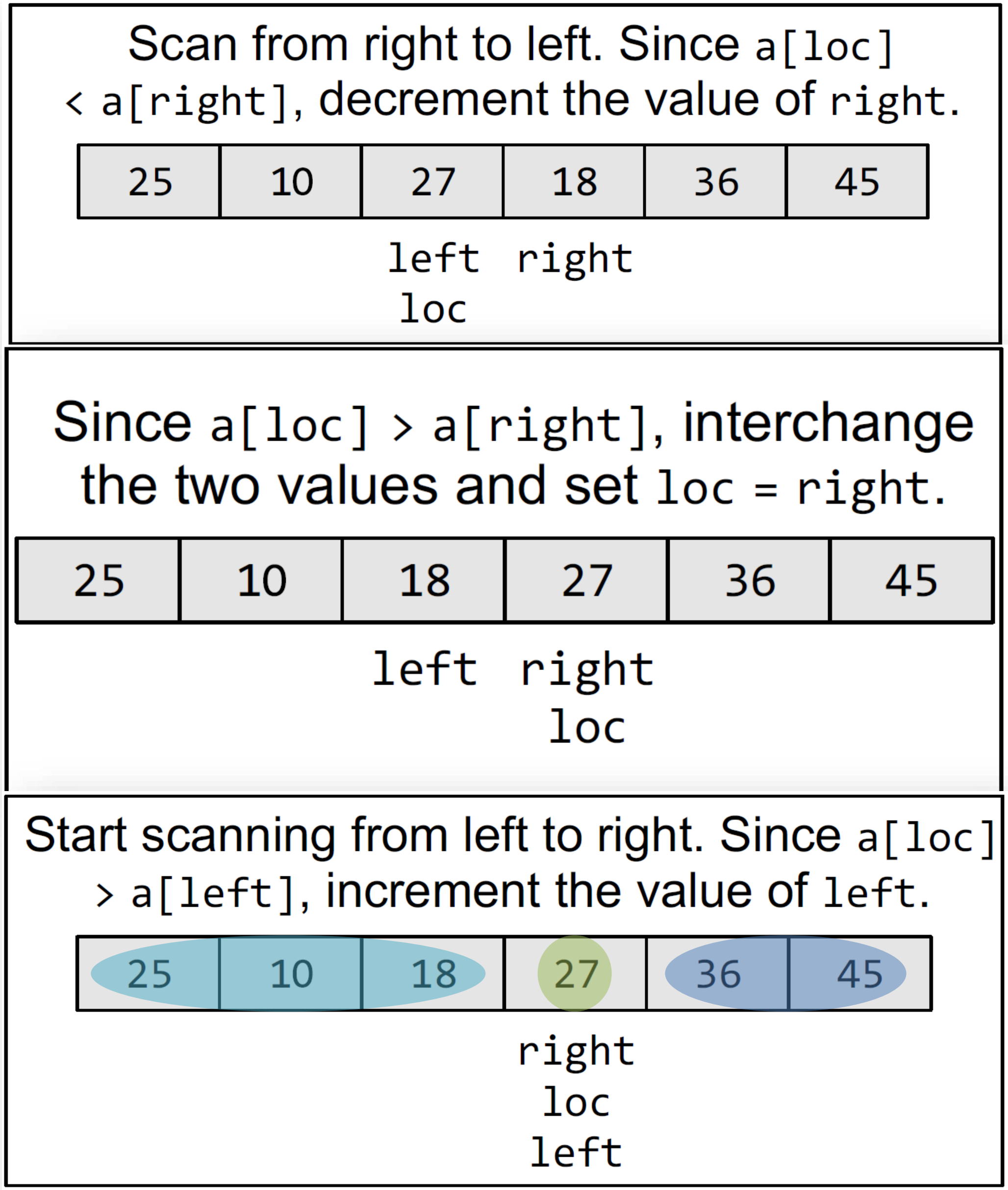
25	10	27	18	36	45
		left loc		right	



Scan from right to left. Since a[loc] < a[right], decrement the value of right.

25	10	27	18	36	45
		left loc	right		

Example….



Scan from right to left. Since a[loc] < a[right], decrement the value of right.

25	10	27	18	36	45
		left loc	right		

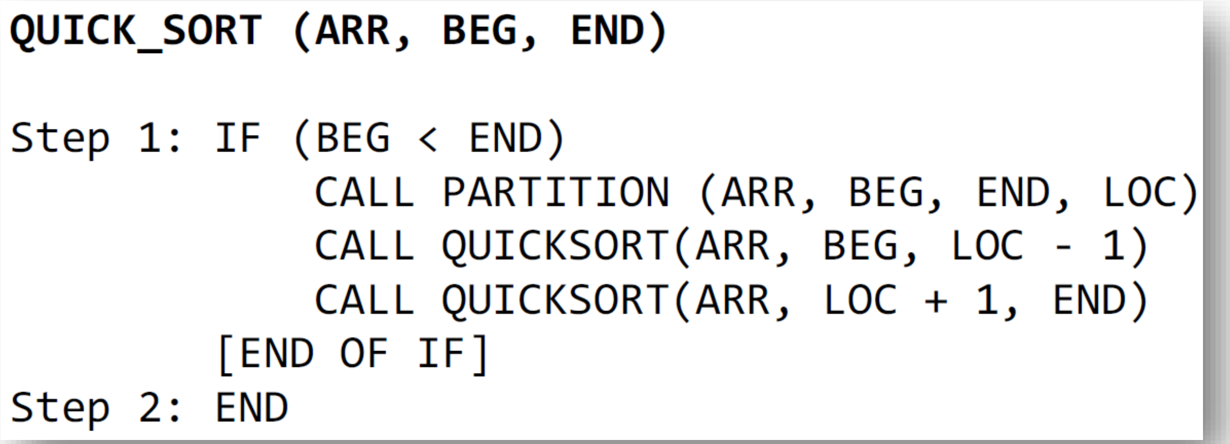
Since a[loc] > a[right], interchange the two values and set loc = right.

25	10	18	27	36	45
		left	right loc		

Start scanning from left to right. Since a[loc] > a[left], increment the value of left.

25	10	18	27	36	45
			right loc left		

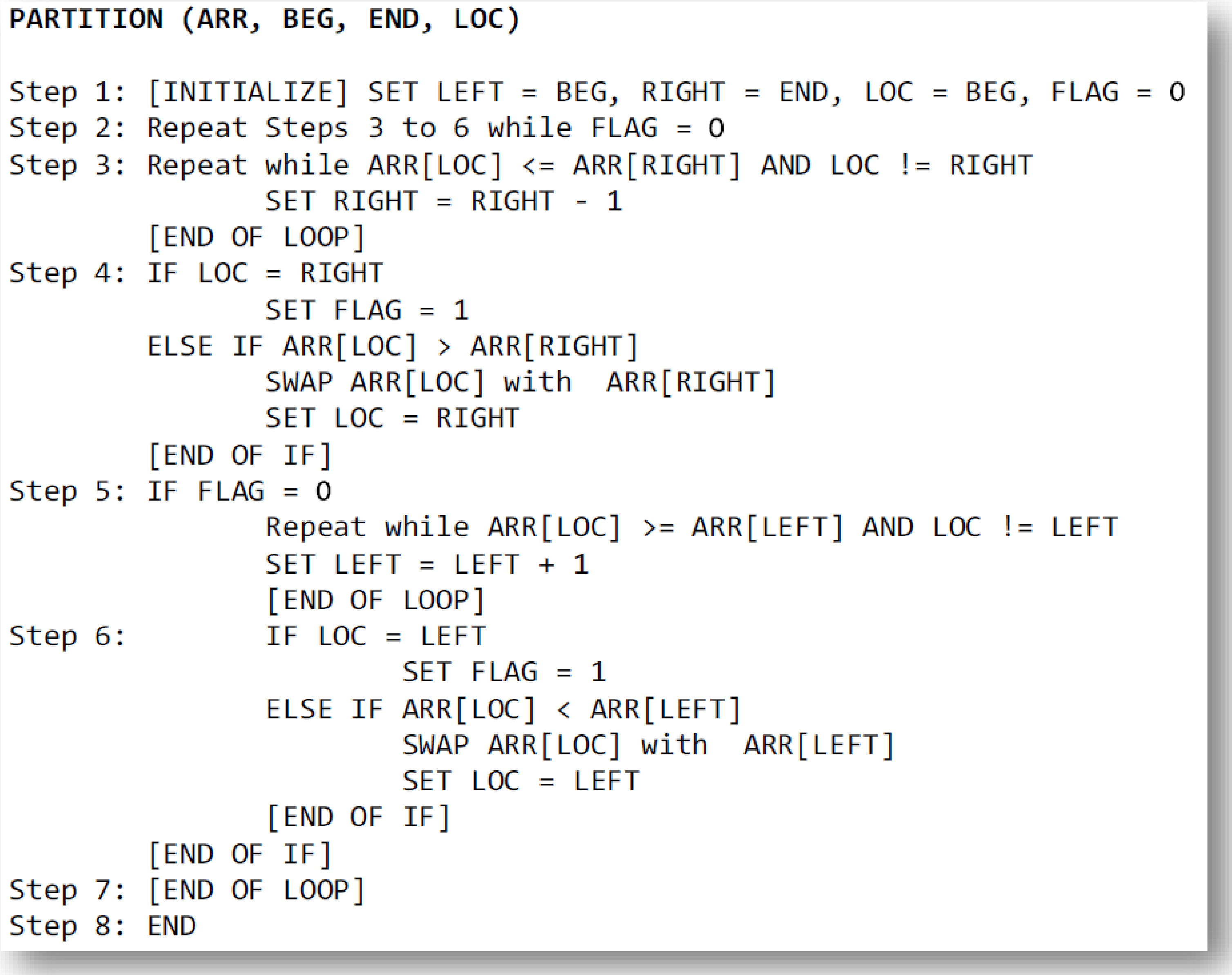
Quick Sort..



```
QUICK_SORT (ARR, BEG, END)

Step 1: IF (BEG < END)
            CALL PARTITION (ARR, BEG, END, LOC)
            CALL QUICKSORT(ARR, BEG, LOC - 1)
            CALL QUICKSORT(ARR, LOC + 1, END)
        [END OF IF]
Step 2: END
```

Quick Sort...

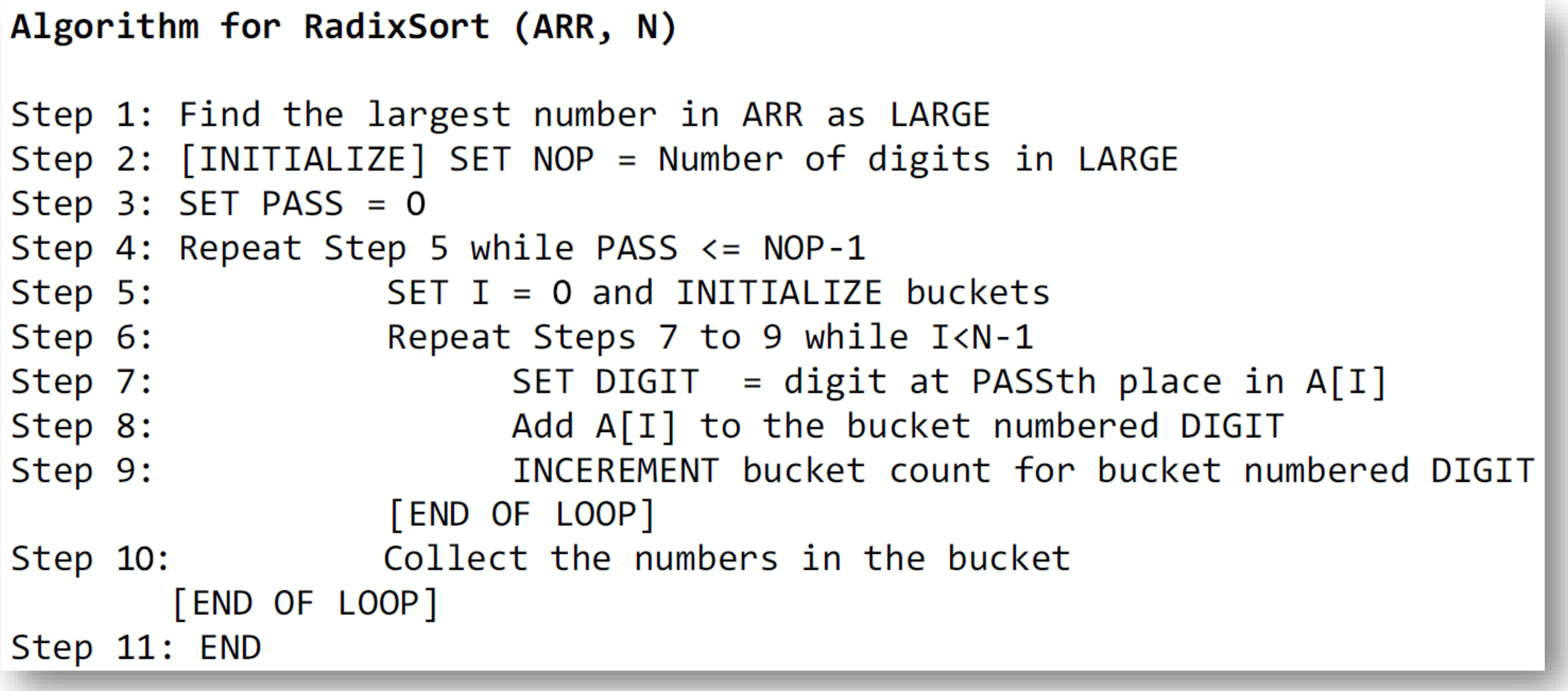


```
PARTITION (ARR, BEG, END, LOC)

Step 1: [INITIALIZE] SET LEFT = BEG, RIGHT = END, LOC = BEG, FLAG = 0
Step 2: Repeat Steps 3 to 6 while FLAG = 0
Step 3: Repeat while ARR[LOC] <= ARR[RIGHT] AND LOC != RIGHT
               SET RIGHT = RIGHT - 1
        [END OF LOOP]
Step 4: IF LOC = RIGHT
               SET FLAG = 1
        ELSE IF ARR[LOC] > ARR[RIGHT]
               SWAP ARR[LOC] with  ARR[RIGHT]
               SET LOC = RIGHT
        [END OF IF]
Step 5: IF FLAG = 0
               Repeat while ARR[LOC] >= ARR[LEFT] AND LOC != LEFT
               SET LEFT = LEFT + 1
               [END OF LOOP]
Step 6:        IF LOC = LEFT
                       SET FLAG = 1
               ELSE IF ARR[LOC] < ARR[LEFT]
                       SWAP ARR[LOC] with  ARR[LEFT]
                       SET LOC = LEFT
               [END OF IF]
        [END OF IF]
Step 7: [END OF LOOP]
Step 8: END
```

Radix Sort.

- Radix sort is a linear sorting algorithm for **integers** and uses the concept of sorting names in alphabetical order
  - Radix sort is also known as bucket sort

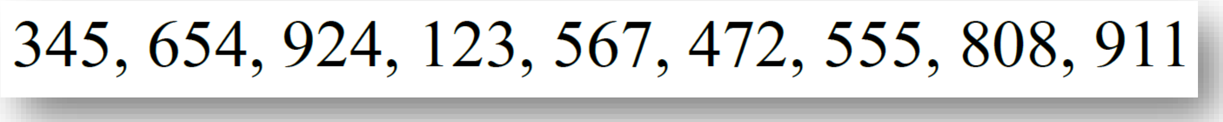


```
Algorithm for RadixSort (ARR, N)

Step 1: Find the largest number in ARR as LARGE
Step 2: [INITIALIZE] SET NOP = Number of digits in LARGE
Step 3: SET PASS = 0
Step 4: Repeat Step 5 while PASS <= NOP-1
Step 5:           SET I = 0 and INITIALIZE buckets
Step 6:           Repeat Steps 7 to 9 while I<N-1
Step 7:                 SET DIGIT  = digit at PASSth place in A[I]
Step 8:                 Add A[I] to the bucket numbered DIGIT
Step 9:                 INCEREMENT bucket count for bucket numbered DIGIT
                  [END OF LOOP]
Step 10:          Collect the numbers in the bucket
        [END OF LOOP]
Step 11: END
```

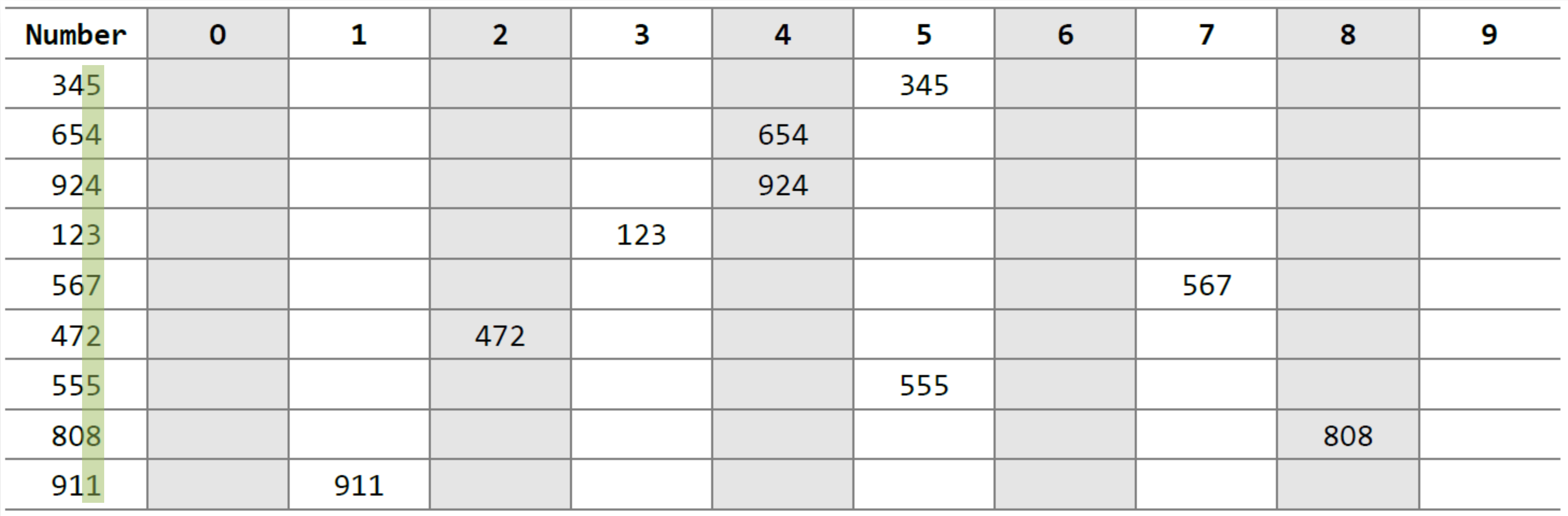
Example.

- Sort the given numbers using radix sort



345, 654, 924, 123, 567, 472, 555, 808, 911

  - The first step: The numbers are sorted according to the digit at ones place
    - The new order is 911, 472, 123, 654, 924, 345, 555, 567, 808

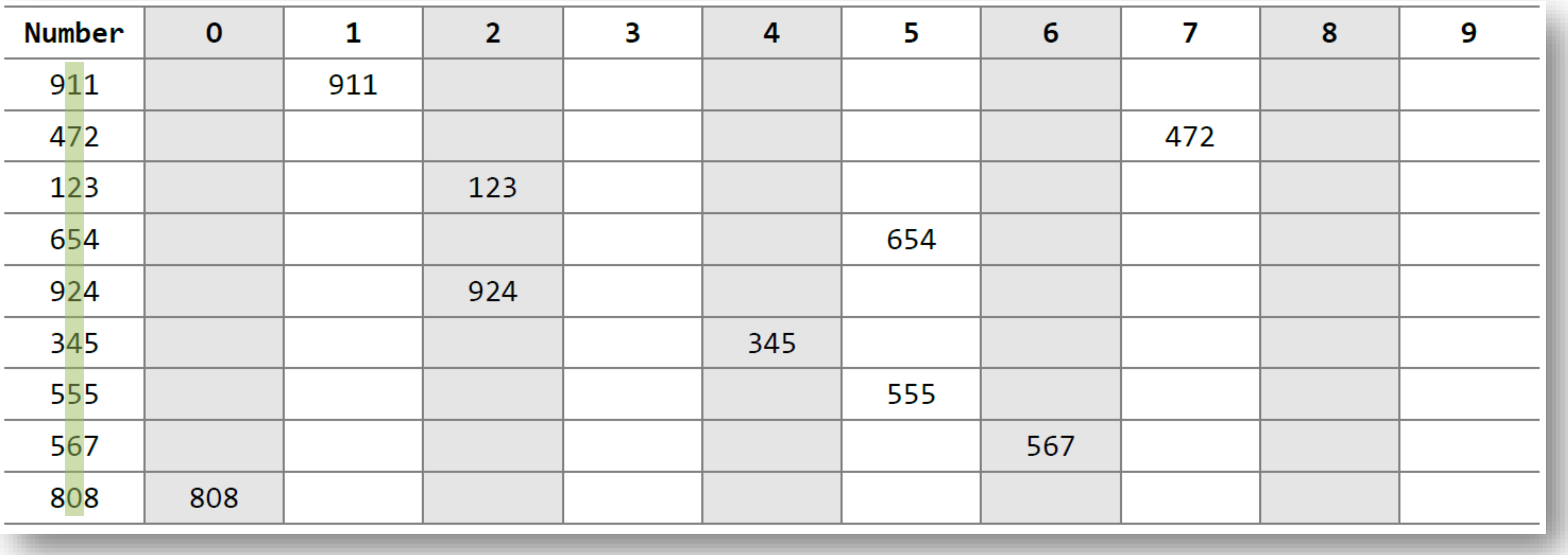


Number	0	1	2	3	4	5	6	7	8	9
345						345				
654					654					
924					924					
123				123						
567								567		
472			472							
555						555				
808									808	
911		911								

Example..

- based on the new order: 911, 472, 123, 654, 924, 345, 555, 567, 808

- The second step: The numbers are sorted according to the digit at the tens place
  - Consequently, the new order is: 808, 911, 123, 924, 345, 654, 555, 567, 472

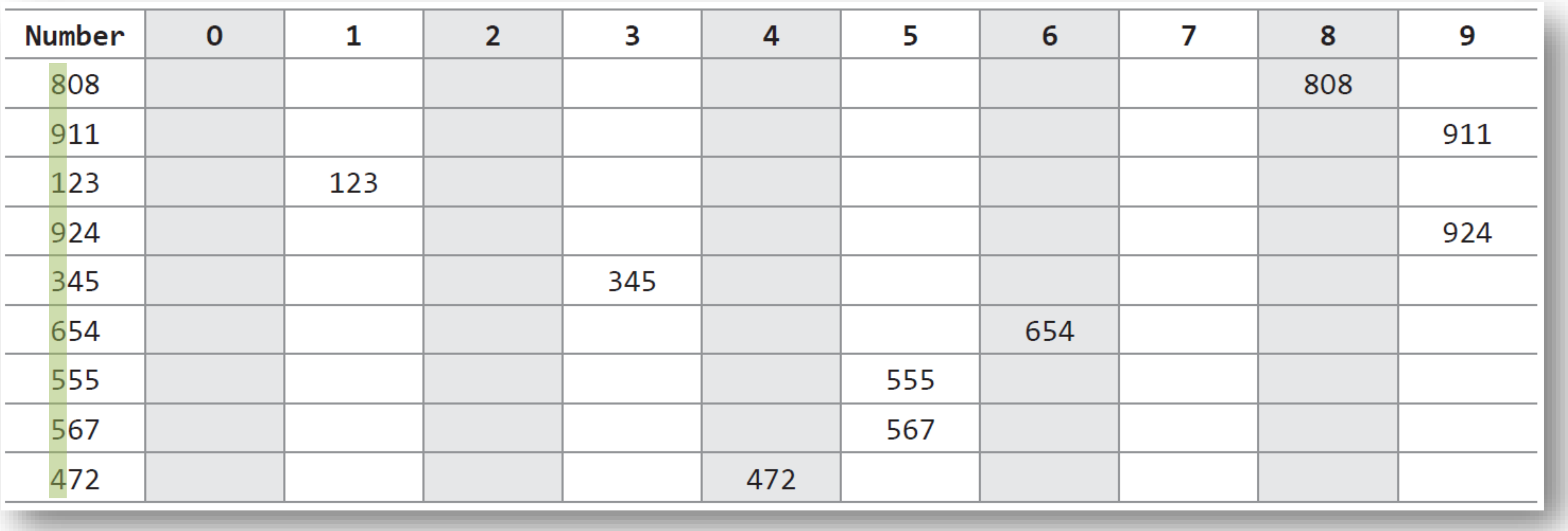


Number	0	1	2	3	4	5	6	7	8	9
911		911								
472								472		
123			123							
654						654				
924			924							
345					345					
555						555				
567							567			
808	808									

Example…

- Based on the new order: 808, 911, 123, 924, 345, 654, 555, 567, 472

- The third step is: The numbers are sorted according to the digit at the hundreds place
  - Finally, the ordered sequence is: 123, 345, 555, 567, 654, 808, 911, 924



Number	0	1	2	3	4	5	6	7	8	9
808									808	
911										911
123		123								
924										924
345				345						
654							654			
555						555				
567						567				
472					472					

Questions?

kychen@mail.ntust.edu.tw