ThS.Trần Lê Như Quỳnh

ALGORITHMS

DATA STUCTURE

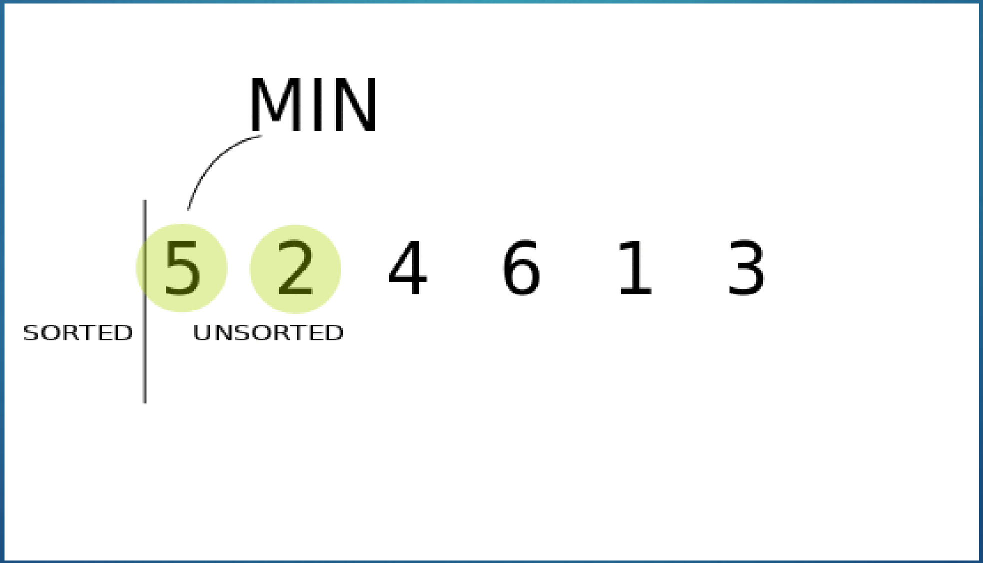
SORT ALGORITHM

ALGORITHM

SELECT SORT ALGORITHM

SORT ALGORITHM

SELECTION SORT ALGORITHM

- Finding the smallest (or largest depending on the sorting order) element in the unsorted sub list exchanging it with the leftmost unsorted element (putting in sorted order) and moving the sub list boundaries one element to the right.
- 

Exercise

34	15	17	35	78	20	11

Step 1: index = 0, a[index] = 34, min = 34, minIndex = 0
34> 15=> min = 15, minIndex = 1
34>17 > min => ko cap nhat min
34 <35 => ko cap nhat min
34 <78 => ko cap nhat min
34>20 >min => ko cap nhat min
34> 11<min => min = 11, minIndex = 6
Đổi chỗ cho 34 và 11

11	15	17	35	78	20	34

Exercise

11	15	17	35	78	20	34

Step 2: index = 1, a[index] = 15, min = 15, minIndex = 0
15<17=> ko cap nhat min
15< 35 => ko cap nhat min
15 <78 => ko cap nhat min
15 <20 => ko cap nhat min
15 <34 => ko cap nhat min

11	15	17	35	78	20	34

ThS.Trần Lê Như Quỳnh

Exercise

11	15	17	35	78	20	34

Step 3: index = 2, a[index] =17, min = 17, minIndex = 2
17<35=> ko cap nhat min
17 <78 => ko cap nhat min
17 <20 => ko cap nhat min
17 <34 => ko cap nhat min

11	15	17	35	78	20	34

ThS.Trần Lê Như Quỳnh

Exercise

11	15	17	35	78	20	34

Step 4: index = 3, a[index] =35, min = 35, minIndex = 3
35 <78 => ko cap nhat min
35 > 20 => min = 20, minIndex =5
35 > 34 > min => ko cap nhat min

11	15	17	20	78	35	34

ThS.Trần Lê Như Quỳnh

Exercise

11	15	17	20	78	35	34

Step 5: index = 4, a[index] =78, min = 78, minIndex = 4
78 > 35 => min = 35, minIndex =5
78 > 34 <35 > min => min = 34, minIndex =6

11	15	17	20	34	35	78

ThS.Trần Lê Như Quỳnh

Exercise

11	15	17	20	34	35	78

Step 5: index = 5, a[index] =35, min = 35, minIndex = 5
35 < 78=> ko cập nhật min

11	15	17	20	34	35	78

ThS.Trần Lê Như Quỳnh

RUNNING TIME

- Best: O(n^2)
- Worst: O(n^2)
- AVG: O(n^2)

ThS.Trần Lê Như Quỳnh

RULE

- Step 1: i = 1
- Step 2: Finding X[min] or X[max] in X[i] ... X[n]
- Step 3: Swap X[i] to X[min], if min or max equal i , quit this step.
- Step 4:
  * If i <= n-1 so that i = i +1, run step 2 again.
  * Else, stop, finish sort array.

RECURSIVE IMPLEMENT SELECTION SORT ALGORITHM

```
 public class SelectionSort {
  private static void swap(int[] a, int i, int j) {
  // switch value at index i to value at index  j
}
  public static int[] selectionSort_Min(int[] array,int stepNum) {
  if(stepNum > array.length -2){
          return array;
  } else{
  for (int j = stepNum; j < array.length; j++) {
  // Find the index of the minimum value
  // swap
  }
    }
          return selectionSort_Min(array,stepNum + 1) ; }
```

ThS.Trần Lê Như Quỳnh

NON RECURSIVE IMPLEMENT SELECTION SORT ALGORITHM

ThS.Trần Lê Như Quỳnh

```
public class SelectionSort {
  private static void swap(int[] a, int i, int j) {
  // switch value at index i to value at index  j
}
  public static int[] selectionSort_Min(int[] array) {
  for (int i = 0; i < array.length - 1; i++) {
   for (int j = i + 1; j < array.length; j++) {
  // Find the index of the minimum value
  // swap
  }}
  return array; }
```

Bubble sort

SORT ALGORITHM

Bubble sort (sinking sort)

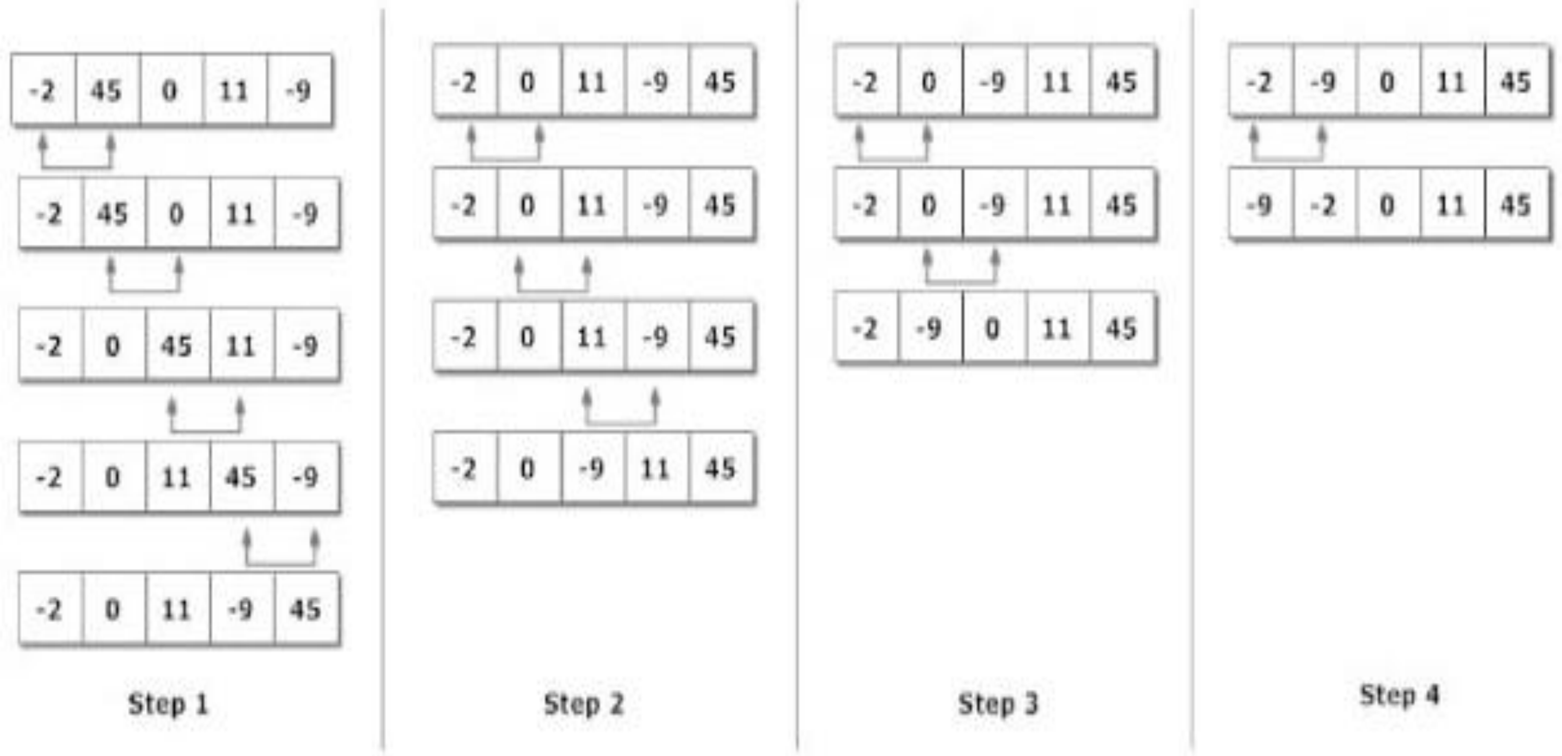
- Sorting is to place elements in increasing or decreasing order.
- Comparing the adjacent pair ,
- if they are in not right order , then they swapped each other position.
- When there are no elements swapped in one full iteration of element list , then it indicates that bubble sort is completed.

RUNNING TIME

- Best: $O(n)$
- Worst: $O(n^2)$
- AVG: $O(n^2)$

Video

Example



Step 1

-2 45 0 11 -9

-2 45 0 11 -9

-2 0 45 11 -9

-2 0 11 45 -9

-2 0 11 -9 45

Step 2

-2 0 11 -9 45

-2 0 11 -9 45

-2 0 11 -9 45

-2 0 -9 11 45

Step 3

-2 0 -9 11 45

-2 0 -9 11 45

-2 -9 0 11 45

Step 4

-2 -9 0 11 45

-9 -2 0 11 45

Figure: Working of Bubble sort algorithm

RULE

- Step 1: i=1
- Step 2: compare max or min and swap (if necessary) from X[i] to X[n] or X[n] to X[i]
- Step 3: i=i+1
- Step 4:
  * If i < n, run step 2 again.
  * Else, stop, finish sorted array.

IMPLEMENT BUBBLE SORT ALGORITHM (RECURSIVE)

```
public static int[] min_bubbleSortRecursive(int[] arr, int n)
    {
        // Base case
        if (n == 1)
            // TO DO
        for (int i=0; i<n-1; i++)
            if (arr[i] > arr[i+1])
            {
             // SWAP
            }
        //RECURSIVE N-1;
    }
```

ThS.Trần Lê Như Quỳnh

IMPLEMENT BUBBLE SORT ALGORITHM(NON-RECURSIVE)

```
public static int[] min_bubbleSort(int[] arr)
    {
        int n = arr.length;
        for (int i = 0; i < n-1; i++)
            for (int j = 0; j < n-i; j++)
                if (arr[j] > arr[j+1])
                {
         // SWAP
                }
        return arr;
    }
```

ThS.Trần Lê Như Quỳnh

Insertion sort

SORT ALGORITHM

Definition

1. Divide to sub list with 2 elements (begin or end)
2. Compare each element with other in sub list, sorted it by ASC or DESC
3. Adding 1 element to sub list and do step 2 again, until has sorted list

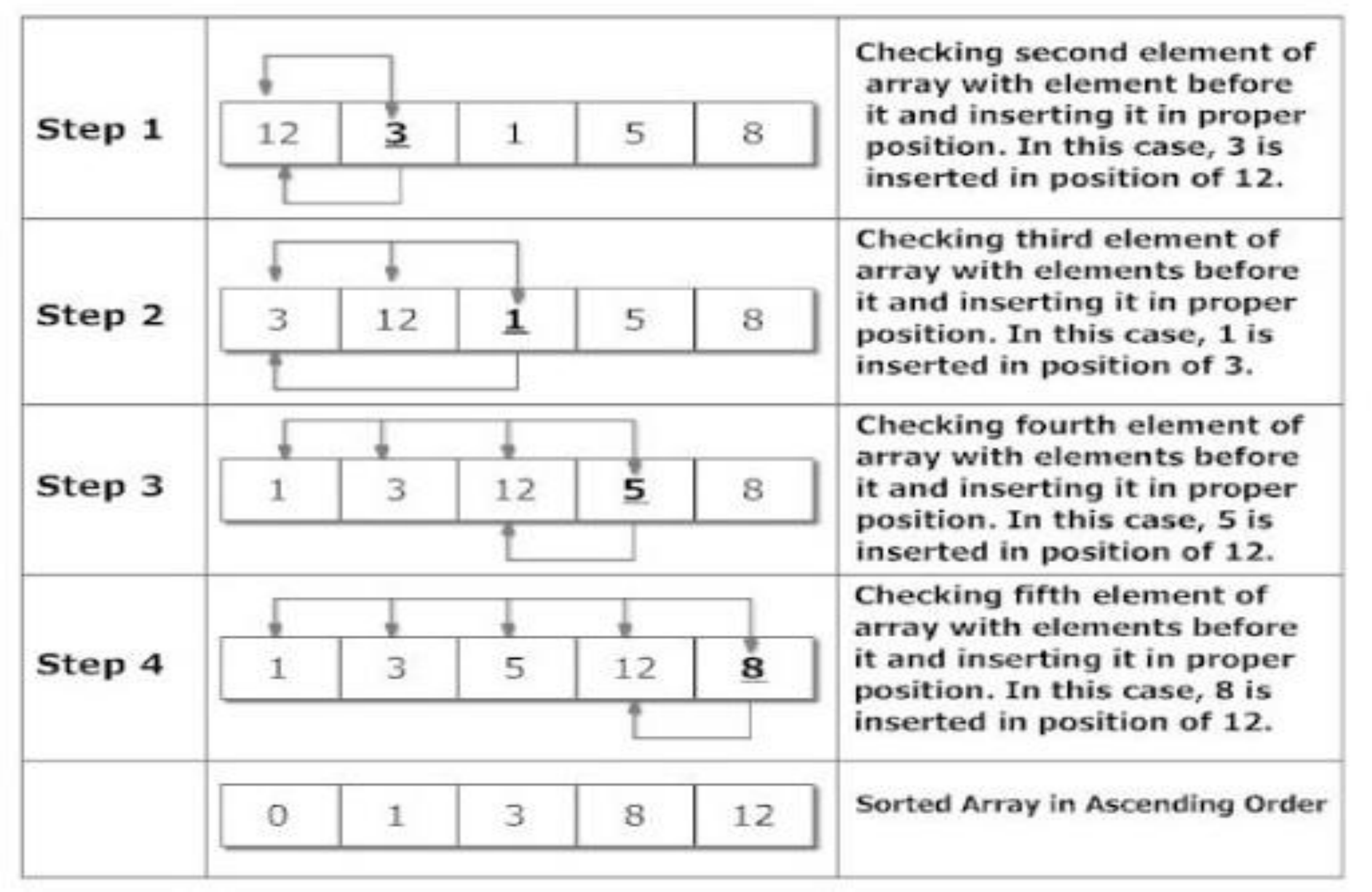
RUNNING TIME

- Best: O(n)
- Worst: O(n^2)
- AVG: O(n^2)

Video

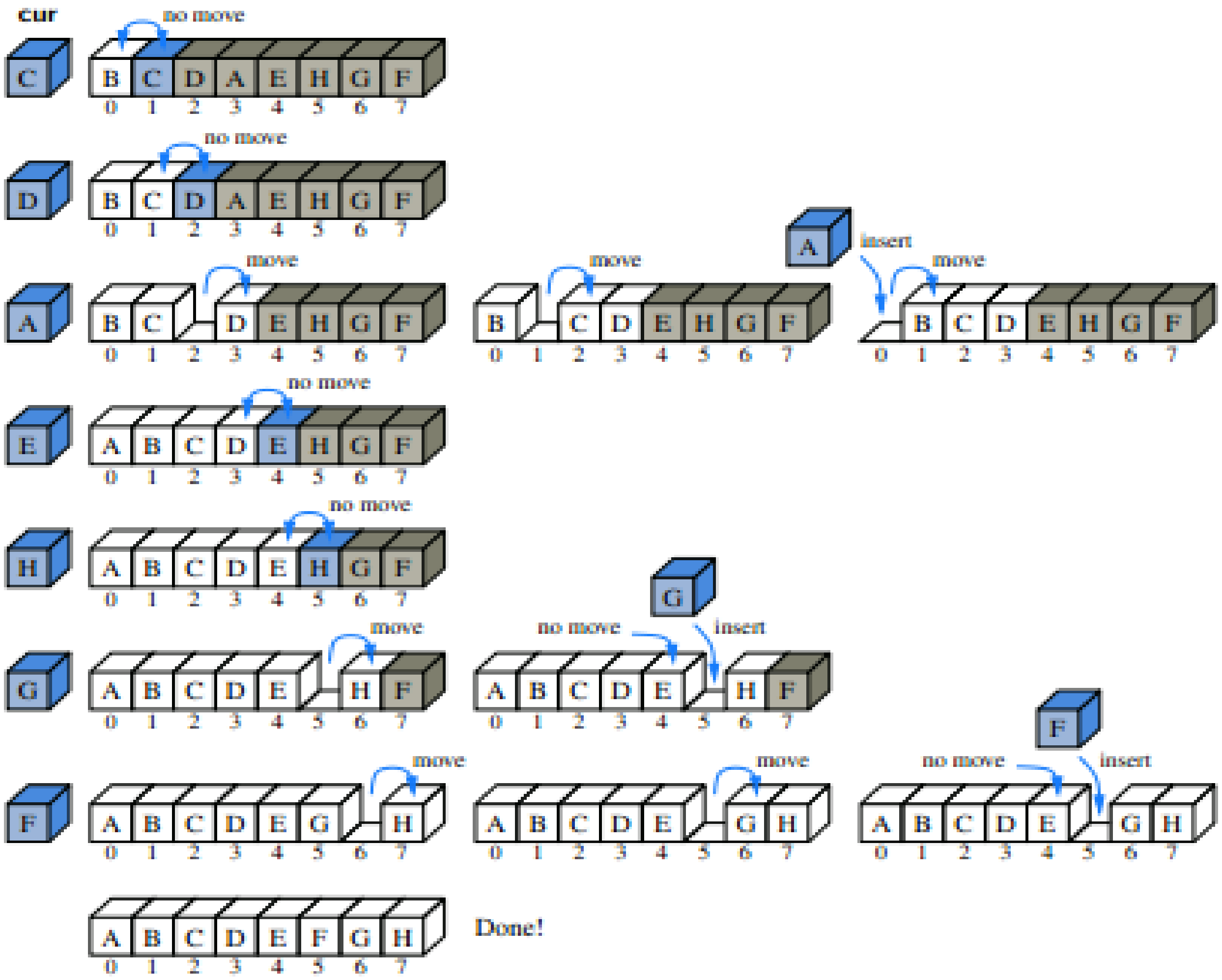
ThS.Trần Lê Như Quỳnh

EXAMPLE



Step 1	12	3	1	5	8	Checking second element of array with element before it and inserting it in proper position. In this case, 3 is inserted in position of 12.
Step 2	3	12	1	5	8	Checking third element of array with elements before it and inserting it in proper position. In this case, 1 is inserted in position of 3.
Step 3	1	3	12	5	8	Checking fourth element of array with elements before it and inserting it in proper position. In this case, 5 is inserted in position of 12.
Step 4	1	3	5	12	8	Checking fifth element of array with elements before it and inserting it in proper position. In this case, 8 is inserted in position of 12.
	0	1	3	8	12	Sorted Array in Ascending Order

EXAMPLE


cur
no move
C
B C D A E H G F
0 1 2 3 4 5 6 7
no move
D
B C D A E H G F
move
A
B C D E H G F
move
B C D E H G F
A
insert
move
B C D E H G F
no move
E
A B C D E H G F
no move
H
A B C D E H G F
move
G
A B C D E H F
no move
G
insert
A B C D E H F
move
F
A B C D E G H
move
A B C D E G H
no move
F
insert
A B C D E G H
A B C D E F G H
Done!

Recursive idea

- Base Case:
- If array size is 1 or smaller, return.
- Recursively sort first n-1 elements. Insert last element at its correct position in sorted array
- // Sort an arr[] of size n insertionSort(arr, n)
- Loop from i = 1 to n-1. a) Pick element arr[i] and insert it into sorted sequence arr[0..i-1]

IMPLEMENT INSERT SORT _RECURSIVE

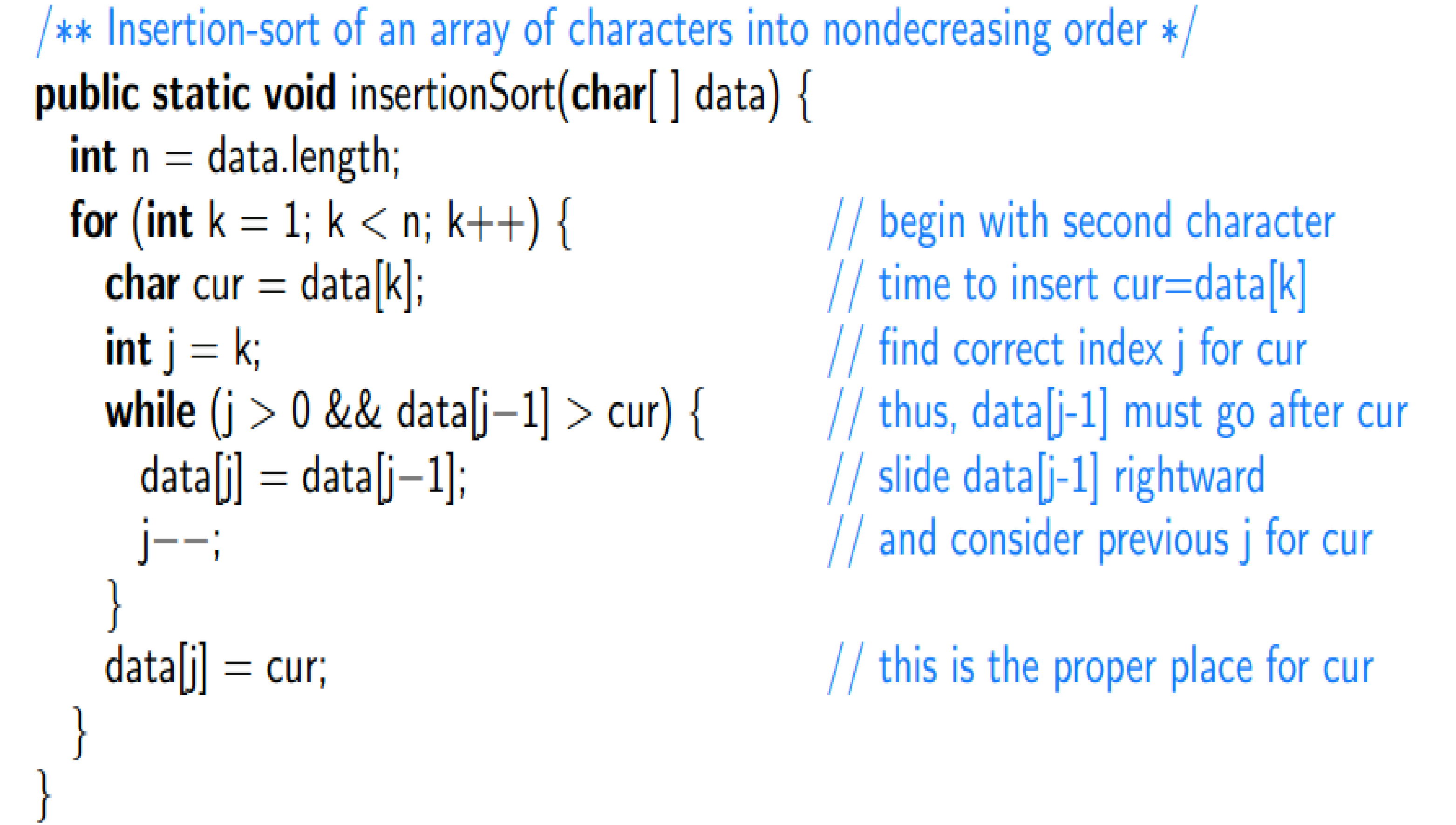
```
int[ ] insertionSortRecursive(int[] arr, int n)
{
    // Base case
    if (n <= 1)
        return arr;
    // Sort first n-1 elements
    insertionSortRecursive( arr, n-1 );
    // Insert last element at its correct position
    // in sorted array.
    int last = arr[n-1];
    int j = n-2;
    /* Move elements of arr[0..i-1], that are
      greater than key, to one position ahead
      of their current position */
    while (j >= 0 && arr[j] > last)
    {
        arr[j+1] = arr[j];
        j--;
    }
    arr[j+1] = last;
}
```

IMPLEMENT INSERT SORT _NON RECURSIVE

```
void insertionSort(int arr[], int n)
{
   int i, key, j;
   for (i = 1; i < n; i++)
   {
       key = arr[i];
       j = i-1;

       /* Move elements of arr[0..i-1], that are
          greater than key, to one position ahead
          of their current position */
       while (j >= 0 && arr[j] > key)
       {
           arr[j+1] = arr[j];
           j = j-1;
       }
       arr[j+1] = key;
   }
}
```

IMPLEMENT INSERT-SORT ALGORITHM



```
/** Insertion-sort of an array of characters into nondecreasing order */
public static void insertionSort(char[ ] data) {
  int n = data.length;
  for (int k = 1; k < n; k++) {                 // begin with second character
    char cur = data[k];                         // time to insert cur=data[k]
    int j = k;                                  // find correct index j for cur
    while (j > 0 && data[j−1] > cur) {          // thus, data[j-1] must go after cur
      data[j] = data[j−1];                      // slide data[j-1] rightward
      j−−;                                      // and consider previous j for cur
    }
    data[j] = cur;                              // this is the proper place for cur
  }
}
```

ThS.Trần Lê Như Quỳnh

BÀI TẬP LÝ THUYẾT

- Sinh viên làm ra giấy chụp hình và nộp lại hoặc làm file word nộp lại.
- Chạy bằng tay 3 giải thuật trên để sắp xếp lại các mảng bên dưới.

A

23	14	25	0	-14	31	22

B

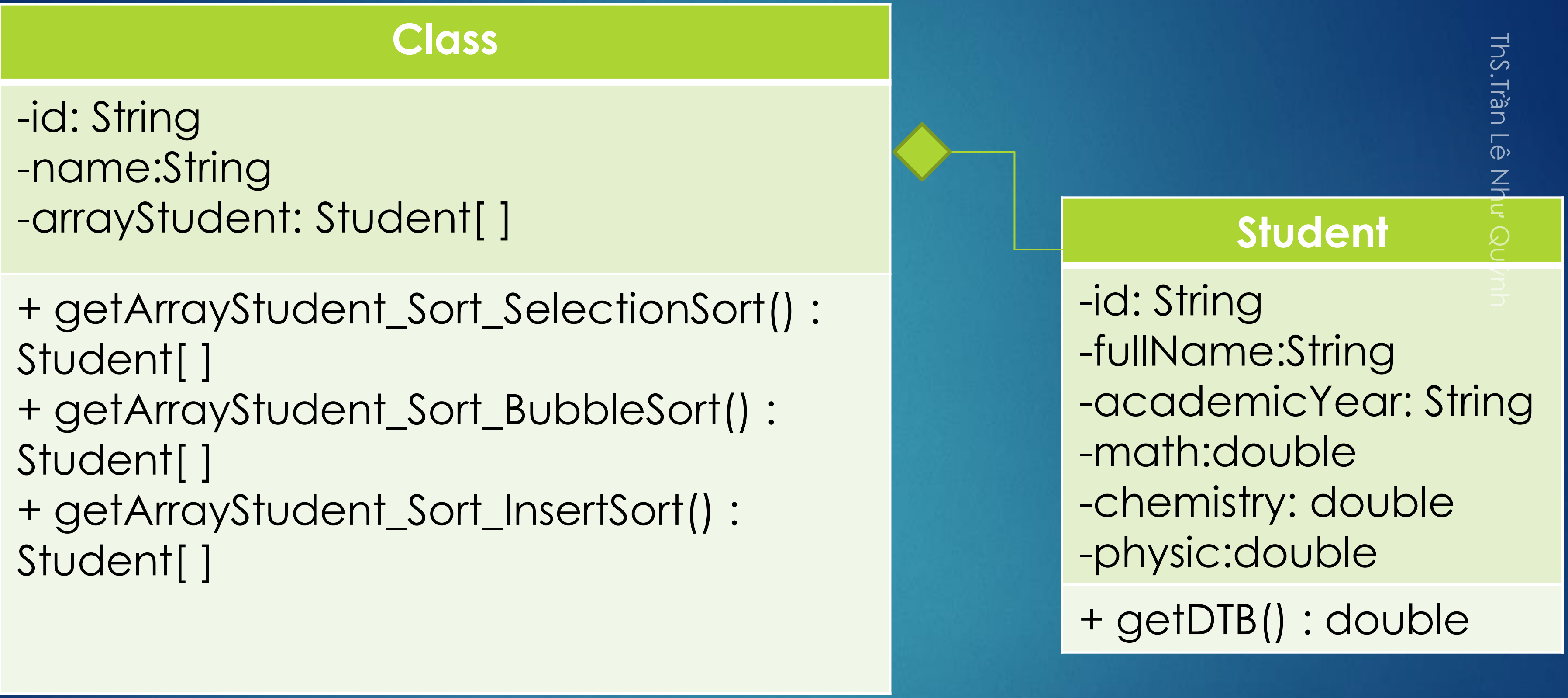
G	K	L	A	J	I	Q	Z	C

- Tiến hành so sánh tốc độ thực thi giữa các giải thuật khi chạy các mảng trên

ThS.Trần Lê Như Quỳnh

EXCERCISE

Manage class’s scorce



Class

-id: String
-name:String
-arrayStudent: Student[ ]

+ getArrayStudent_Sort_SelectionSort() : Student[ ]
+ getArrayStudent_Sort_BubbleSort() : Student[ ]
+ getArrayStudent_Sort_InsertSort() : Student[ ]

Student

-id: String
-fullName:String
-academicYear: String
-math:double
-chemistry: double
-physic:double

+ getDTB() : double