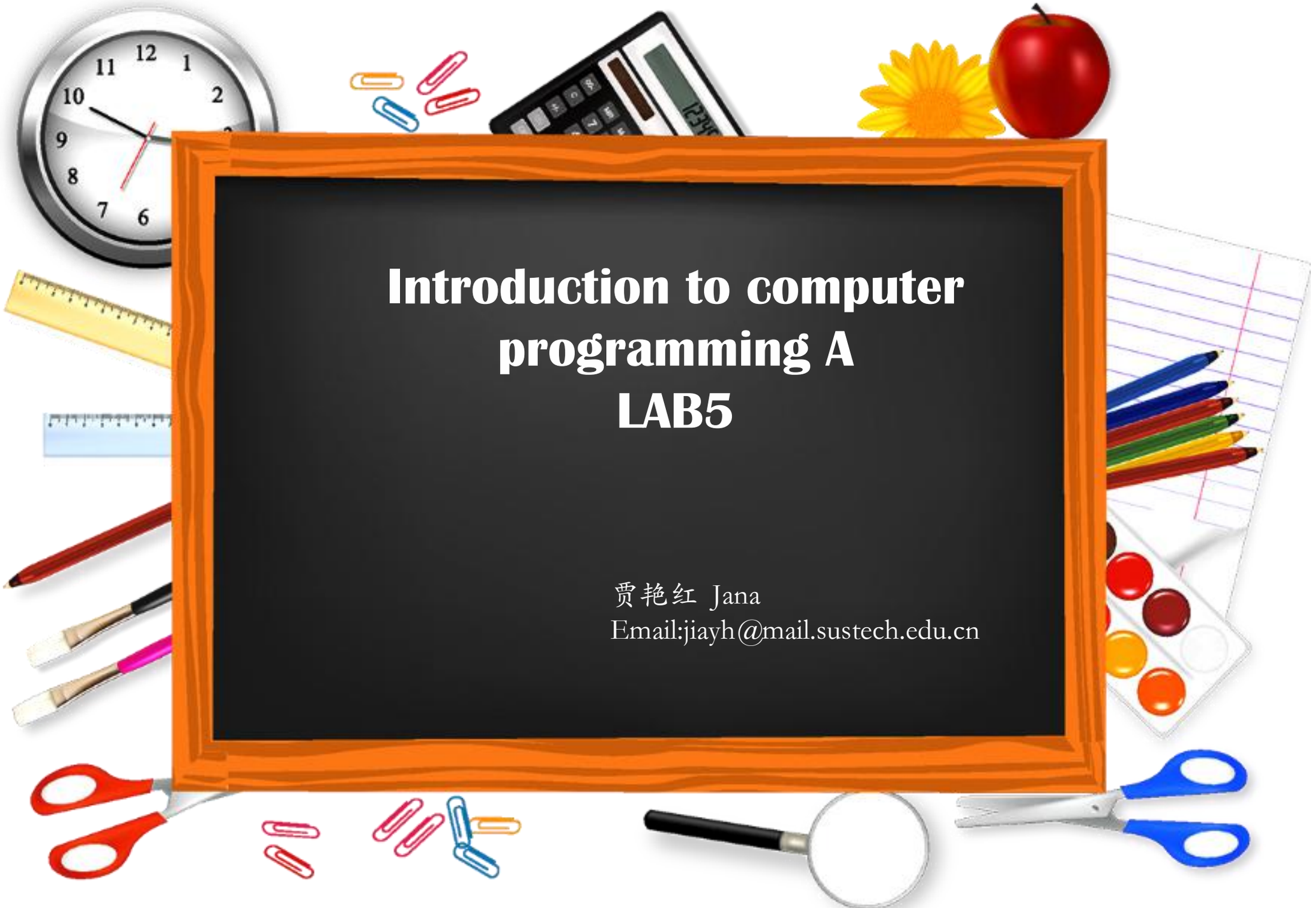


Introduction to computer programming A
LAB5

贾艳红 Jana
Email:jiayh@mail.sustech.edu.cn

LAB OBJECTIVES

1. Learn array initializer (Declare, create, and initialize).
2. Learn how to copy and print array by for loop
3. Learn how to using array to realize some simple algorithms.

knowledge points

Array Declaration, Definition, Initialization

```java
public class SimpleArrayDemo {

    public static void main(String[] args) {
        // decalring an array
        double[] newArray0;
①       // Defining an array
        newArray0 = new double[5];
        // initializing the array
        newArray0[0] = 5.6;
        newArray0[1] = 4.5;
        newArray0[2] = 3.3;
        newArray0[3] = 13.2;
        newArray0[4] = 4.0;

        // decalring and Defining an array
②       int[] newArray1 = new int[5];
        // Initializing elements of array seperately
        for (int n = 0; n < newArray1.length; n++) {
            newArray1[n] = n;
        }

        // declaring an array variable
        double[] newArray2;
③       // defining and Initializing
        newArray2 = new double[] { 1.9, 2.9, 3.4, 3.5 };

        // traverse the array elements sequentially
        for (double element : newArray2) {
            System.out.print(element + " ");
        }

        System.out.println();
        //decalring、Defining and Initializing
4       double[] newArray3 = { 3.3, 2.1, 4.4, 4.5 };
        //Accessing Array Elements through the index

        for (int i = 0; i < newArray3.length; i++) {
            System.out.print(newArray3[i] + " ");
        }

    }

}
```

```java
double[] myList;
myList = {1.9, 2.9, 3.4, 3.5};
```

Wrong

Traverse the array elements

```java
public class SimpleArrayDemo {

    public static void main(String[] args) {
        // decalring an array
        double[] newArray0;
        // Defining an array
        newArray0 = new double[5];
        // initializing the array
        newArray0[0] = 5.6;
        newArray0[1] = 4.5;
        newArray0[2] = 3.3;
        newArray0[3] = 13.2;
        newArray0[4] = 4.0;

        // decalring and Defining an array
        int[] newArray1 = new int[5];
        // Initializing elements of array seperately
        for (int n = 0; n < newArray1.length; n++) {
            newArray1[n] = n;
        }

        // declaring an array variable
        double[] newArray2;
        // defining and Initializing
        newArray2 = new double[] { 1.9, 2.9, 3.4, 3.5 };

        // traverse the array elements sequentially
        for (double element : newArray2) {
            System.out.print(element + " ");
        }

        System.out.println();
        //decalring、Defining and Initializing
        double[] newArray3 = { 3.3, 2.1, 4.4, 4.5 };
        //Accessing Array Elements through the index

        for (int i = 0; i < newArray3.length; i++) {
            System.out.print(newArray3[i] + " ");
        }

    }

}
```

①

for-each loop

②

Traverse the array elements

The syntax of Java for-each loop:

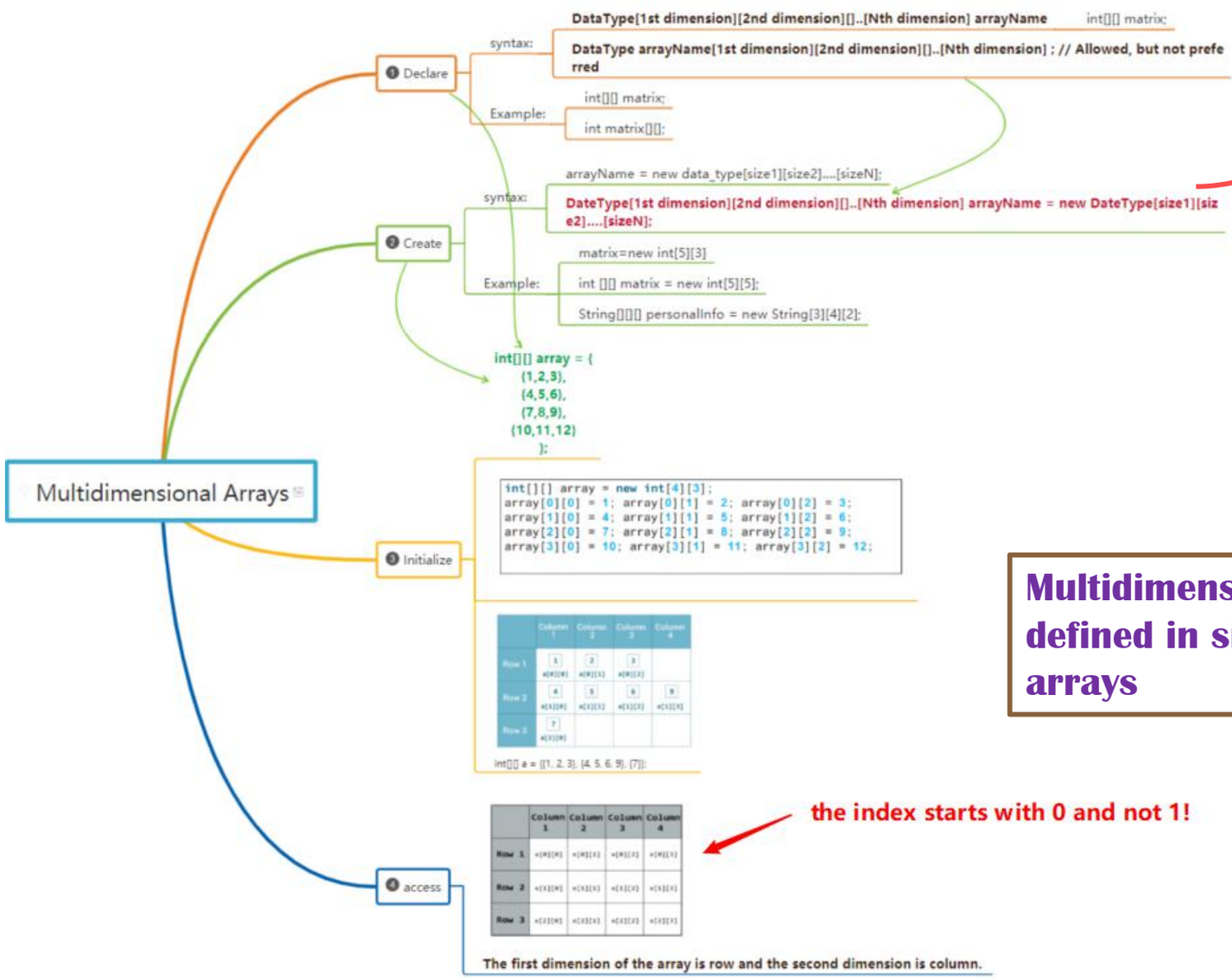
```
for(data_type variable : array | collection){
//body of for-each loop
}
```

For-each loop Example: Traversing the array elements

```
//An example of Java for-each loop
class ForEachExample{
  public static void main(String args[]){
   //declaring an array
   int arr[]={12,13,14,44};
   //traversing the array with for-each loop
   for(int i:arr){
     System.out.println(i);
   }
 }
}
```

Multidimensional Arrays

- **DateType**: Type of data to be stored in the array. For example: int, char, etc.
- **dimension**: The dimension of the array created. For example: 1D, 2D, etc.
- **arrayName**: Name of the array
- **size1, size2, …, sizeN:** Sizes of the dimensions respectively.



Multidimensional Arrays can be defined in simple words as array of arrays

Multidimensional Arrays

Example: Print all elements of 2d array Using Loop

```
class MultidimensionalArray {
   public static void main(String[] args) {
      int[][] a = {
            {1, -2, 3},
            {-4, -5, 6, 9},
            {7},
      };

      for (int i = 0; i < a.length; ++i) {
         for(int j = 0; j < a[i].length; ++j) {
            System.out.println(a[i][j]);
         }
      }
   }
}
```

Equivalent

using for..each loop:

```
for (int[] innerArray: a) {
    for(int data: innerArray) {
        System.out.println(data);
    }
}
```

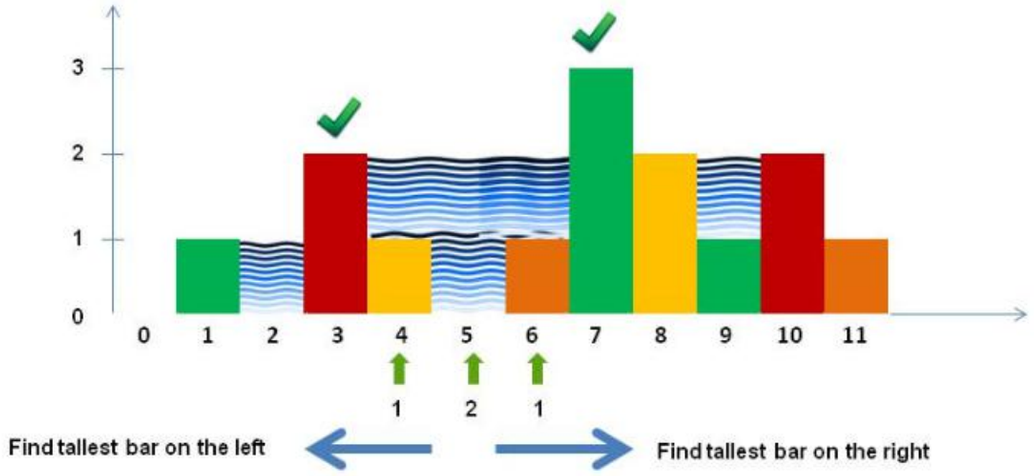
```
1
-2
3
-4
-5
6
9
7
```

Multidimensional Arrays

```java
import java.util.Scanner;

public class MultiDimArrayDemo {

    public static void main(String[] args) {
        //declaring、defining a matrix
        int[][] matrix = new int[5][5];
        // Accessing Array elements through row index and column index

        matrix[2][1]=7;
        printMatrix(matrix);

        // declaring、defining and initializing a Multidimensional Array
        String[][] names = { { "Mr. ", "Mrs. ", "Ms. " }, { "Smith", "Jones" } };
        // Accessing Array elements through row index and column index
        System.out.println(names[0][0] + names[1][0] + " " );
        System.out.println(names[0][2] + names[1][1]);


    }
    // print each elemnt
    public static void printMatrix(int[][] matrix) {
        for (int row = 0; row < matrix.length; row++) {
            for (int column = 0; column < matrix[row].length; column++) {
                System.out.print(matrix[row][column] + " ");
            }
            System.out.println();
        }
    }
}
```

```
Console
<terminated> MultiDim
0 0 0 0 0
0 7 0 0 0
0 0 0 0 0
0 0 0 0 0
Mr. Smith
Ms. Jones
```

Exercise3

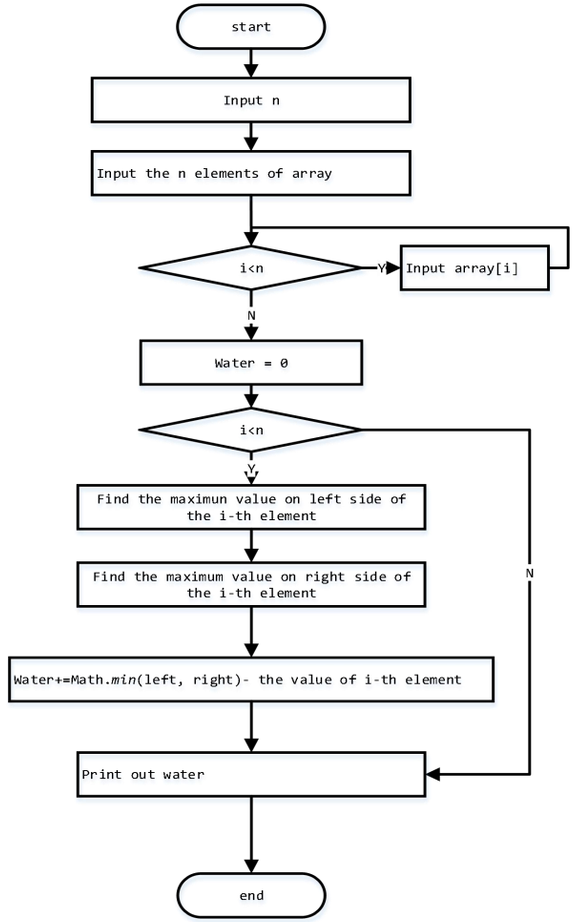


So for example,

water trapped at index 5 = min(2,3) - 1 = 1

index 6 = min(2,3) - 0 = 2

index 7 = min(2,3) - 1 = 1



Approach 1 (Naive Approach)

1. Traverse every array element
2. For each element,
   - Find the highest bars on left and right sides.
   - Take the smaller of two heights.
   - The difference between smaller height and height of current element is the amount of water that can be stored in this array element.
3. Time complexity of this solution is $O(n^2)$ as
   - we are traversing each element i.e. n times +
   - for each element we are searching left and right bar with max hight i.e. again n times.

This is not very efficient solution for the given problem.

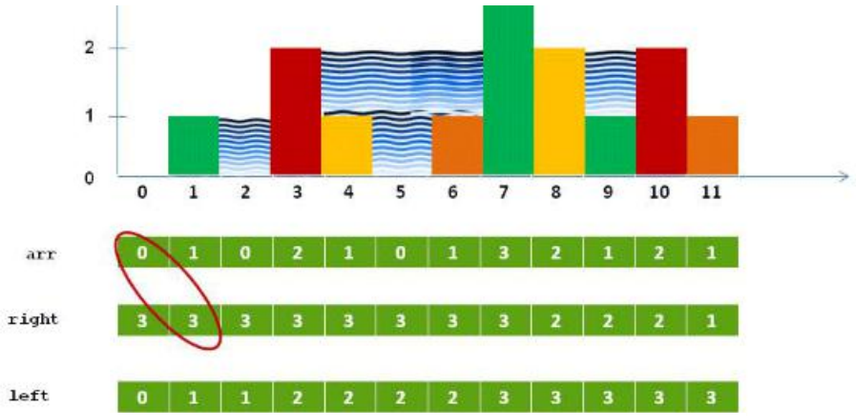
Exercise3

Approach 2

1. First compute highest bar on the left of every bar in O(n)
   - by getting maximum between height of current bar and height of the previous tallest bar. Store this maximum value at the current index of `left` array.

2. Similarly compute highest bar on the right of every bar in O(n) and store it in `right` array.

Here is the graphical representation of above two steps.



3. Now at any point of the bar the maximum water that can be trapped will be minimum of left height and right height minus the height of the bar.

> *Amount of water stored on the top of the bar i = (maximum of left(i) and right(i)) - arr(i)*

4. Keep adding the value for each bar.

Yeah... We solved it.

4 Exercises

Complete the exercises in the **2021S-Java-A-Lab-5.pdf** and submit to the blackboard as required.

贾艳红 Jana
Email:jiayh@mail.sustech.edu.cn