Assignment 1

Summative Assignment 1

LC Data Structures and Algorithms

due date 5 March 2025, 12pm

You are given an $n \times m$ matrix with integer entries that has the following properties:

(1) Each row has a unique maximum value,

(2) If the maximum value in row `i` of the matrix is located at column `j`, then the maximum value in row `i+1` of the matrix is located at a column `k`, where `k` $\geq$ `j`.

The goal of this assignment is to find the maximum value in such a matrix.

Question 1. Write a function `maxIndex` that finds the index of the maximum entry of of a row between columns with indices `start` and `end` inclusively. The row is given as an array `row`. What is the time complexity of your solution? Explain your answer.

Question 2. A rectangular *block* of a matrix is given by a row and column of the upper-left corner in `startRow` and `startCol`, and row and column of the lower-right corner `endRow` and `endCol`, such that `startRow` $\leq$ `endRow` and `startCol` $\leq$ `endCol`. Write a function `blockMaxValue` that finds the value of the maximum entry of a given block assuming that the block satisfies the properties (1) and (2) above.

Hint: Use the divide-and-conquer strategy.

Question 3. Write a function `matrixMaxValue` that finds the maximum value of a matrix that satisfies properties (1) and (2) above, and provide a better upper bound for the time complexity of this function than $O(nm)$. Explain your answer.

Hint: The complexity of linear search is $O(nm)$, do better than that!

Submission

Submission is via Canvas, and it should contain two files:

- Java source code named '`solution.java`' containing a `class Solution` with the following methods:

```
public class Solution {
    public static int maxIndex(int[] row, int start, int end);

    public static int blockMaxValue(int[][] matrix,
        int startRow, int startCol, int endRow, int endCol);

    public static int matrixMaxValue(int[][] matrix);
}
```

  Do not rename the class or the methods, otherwise your solution will fail the test cases.

- A text/pdf file containing the explanation of the complexity of your code.

```
public class Test {

    static void printTest(int result, int output) {
        if (result == output) {
            System.out.println("Test passed.");
        } else {
            System.out.println("Test failed: Expected " + output + ", got " + resu
lt);
        }
    }

    static void testMaxIndex(int[] row, int start, int end, int output) {
        System.out.println("Testing maxIndex on row between " + start + " an
d " + end);
        int result = Solution.maxIndex(row, start, end);
        printTest(result, output);
    }

    static void testBlockMaxValue(int[][] matrix, int startRow, int startCol, int
endRow, int endCol, int output) {
        System.out.println("Testing blockMaxValue between (" + startRow +
"," + startCol + ") and (" + endRow + "," + endCol + ")");
        int result = Solution.blockMaxValue(matrix, startRow, startCol, endRo
w, endCol);
        printTest(result, output);
    }

    static void testMatrixMaxValue(int[][] matrix, int output) {
        System.out.println("Testing matrixMaxValue");
        int result = Solution.matrixMaxValue(matrix);
        printTest(result, output);
    }

    static void testCasesMaxIndex() {
        {
            int[] row = {1, 3, 2, -1, 0, 1}; int start = 0, end = 5, output = 1;
            testMaxIndex(row, start, end, output);
        }
```

```
        {
            int[] row = {3, 2, -1, 0, 1}; int start = 2, end = 4, output = 4;
            testMaxIndex(row, start, end, output);
        }
        {
            int[] row = {1, -1, -3, -2}; int start = 1, end = 3, output = 1;
            testMaxIndex(row, start, end, output);

        }
    }

    static void testCasesBlockMaxValue() {
        {
            int[][] matrix = {{1, 7, 6}, {3, 8, 5}, {4, 2, 9}};
            int startRow = 0, startCol = 0, endRow = 0, endCol = 2;
            int output = 7;
            testBlockMaxValue(matrix, startRow, startCol, endRow, endCol, outp
ut);
        }
        {
            int[][] matrix = {{3}, {2}, {-1}, {0}, {1}};
            int startRow = 2, startCol = 0, endRow = 4, endCol = 0;
            int output = 1;
            testBlockMaxValue(matrix, startRow, startCol, endRow, endCol, outp
ut);
        }
        {
            int[][] matrix = {{1, -1, -2, -4}, {-1, -3, -2, -5}, {1, 2, -2, -1}};
            int startRow = 0, startCol = 2, endRow = 2, endCol = 3;
            int output = -1;
            testBlockMaxValue(matrix, startRow, startCol, endRow, endCol, outp
ut);
        }
    }

    static void testCasesMatrixMaxValue() {
        {
            int[][] matrix = {{1, 7, 6}, {3, 8, 5}, {4, 2, 9}};
```

```
            int output = 9;
            testMatrixMaxValue(matrix, output);
        }
        {
            int[][] matrix = {{3}, {2}, {-1}, {0}, {1}};
            int output = 3;
            testMatrixMaxValue(matrix, output);
        }
        {
            int[][] matrix = {{1, -1, -2, -4}, {-1, -3, -2, -5}, {1, 2, -2, -1}};
            int output = 2;
            testMatrixMaxValue(matrix, output);
        }
    }

    public static void main(String[] args) {
        testCasesMaxIndex();
        testCasesBlockMaxValue();
        testCasesMatrixMaxValue();
    }
}
```

```
public class Solution {

    public static int maxIndex(int[] row, int start, int end) {
        // Returns the index of the maximum entry between start and end

    }

    public static int matrixMaxValue(int[][] matrix) {
        // Returns the maximum value in the matrix

    }


    public static int blockMaxValue(int[][] matrix, int startRow, int startCol, int
endRow, int endCol) {
        // Returns the maximum entry in the matrix between columns startCol,
```

```
endCol inclusive and between rows startRow, endRow inclusive
    }
}
```