Coding challenges

Mohamed Boucetta

Exercise 0.1 *A peak of a matrix M of type $n \times p$ is an item $M[i][j]$ such that $M[i][j] \geq M[i+1][j]$ (if $M[i+1][j]$ exists), $M[i][j] \geq M[i-1][j]$ (if $M[i-1][j]$ exists), $M[i][j] \geq M[i][j+1]$ (if $M[i][j+1]$ exists) and $M[i][j] \geq M[i][j-1]$ (if $M[i][j-1]$ exists). For instance, the matrix*

$$M = \begin{pmatrix} 5 & 6 & 4 & 3 \\ 4 & 8 & 11 & 8 \\ -1 & 12 & 7 & 9 \end{pmatrix}$$

has three peaks 11, 12, 9.
Build a function which takes a matrix of integers of type $n \times p$ and returns a peak.

Solution : There is a brute-force solution which consists in scanning the matrix and comparing each item matrix[i][j] to its neighboors in the top, bottom, left an right. This solution has a time complexity $O(n \times p)$ and $O(1)$ space complexity. (n is the number of rows and p the number of columns).
There is another solution which has time complexity $O(n \ln p)$ and $O(n \times p)$ space complexity.
We denote by C_i the column of i column of the matrix and for $i < j$, $M(i,j)$ the the matrix composed by the columns i to j. For instance if

$$M = \begin{pmatrix} 5 & 6 & 4 & 3 \\ 4 & 8 & 11 & 8 \\ -1 & 12 & 7 & 9 \end{pmatrix}, \; M(0,2) = \begin{pmatrix} 5 & 6 & 4 \\ 4 & 8 & 11 \\ -1 & 12 & 7 \end{pmatrix}.$$

The idea is to choose the column $C_{p/2}$ and matrix[j][p/2] the maximum of $C_{p/2}$. Then :

1. If matrix[j][p/2] $\geq$ matrix[j][p/2-1] and matrix[j][p/2] $\geq$ matrix[j][p/2+1] then matrix[j][p/2] is a peak,
2. If matrix[j][p/2] $<$ matrix[j][p/2-1] then you are certain that there is a peak in matrix(0,p/2-1) and you apply your function to this matrix,

3. If matrix[j][p/2] $<$ matrix[j][p/2+1] then you are certain that there is a peak in matrix(p/2+1,p-1) and you apply your function to this matrix,

It is a recursive algorithm and the base cases are when $p = 1$ the function returns the maximum of the matrix and when $p = 2$ the function returns the maximum of m_0 and m_1 where m_0 is the maximum of C_0 and m_1 is the maximum of C_1.

```swift
func submatrix0fmatrix(matrix: [[Int]], firstColumn: Int, secondColumn: Int) -> [[Int]] {

    var submatrix = Array(repeating: [0], count: matrix.count)

    for i in 0..<matrix.count {
        submatrix[i] = Array(matrix[i][firstColumn...secondColumn])
    }
    return submatrix
    }

func column(matrix:[[Int]], index: Int) -> [Int] {

    var column = Array(repeating: 0, count: matrix.count)

    for i in 0..<matrix.count {
        column[i] = matrix[i][index]
    }

    return column
}

let M = [[1, 2, 3], [6, 3, 4], [8, 9, 3]]
print(submatrix0fmatrix(matrix: M, firstColumn: 0, secondColumn: 1))
print(column(matrix: M, index: 0))
```

```
[[1, 2], [6, 3], [8, 9]]
[1, 6, 8]
```

```
func peakMatrix(_ matrix: [[Int]] ) -> Int {

    let n = matrix[0].count / 2
    let p = matrix[0].count

     if p == 1 {
        return column(matrix: matrix, index: 0).max()!
    }
    if p == 2 {
        return max(column(matrix: matrix, index: 0).max()!,column(matrix: matrix, index: 1).max()!)
    }

    let m = column(matrix: matrix, index: n).max()!
    let j = column(matrix: matrix, index: n).firstIndex(of: m)!

    if matrix[j][n-1] <= m && matrix[j][n+1] <= m {
        return m
    }
    if matrix[j][n-1] > m {
        return peakMatrix(submatrix0fmatrix(matrix: matrix, firstColumn: 0, secondColumn: n-1))
    }

    if matrix[j][n+1] > m {
        return peakMatrix(submatrix0fmatrix(matrix: matrix, firstColumn: n+1, secondColumn: p-1))
    }
    return 0
}


let L = [[1, 0, 9], [2, 5, 8], [27, 26, 13], [4, 25, 8], [17, 24, 7]]

peakMatrix(L)
```