uniandes

# Ingeniería de Sistemas y Computación

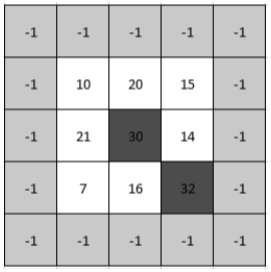
## Diseño y análisis de algoritmos

Profesor: Mateo Sanabria Ardila

Examen 1

Fecha de entrega: 01/Marzo | 2024-10 | Nota máxima: **100**

1. Un elemento cumbre en una matriz 2D es un elemento que es estrictamente mayor que todos sus 4-vecinos (izquierda, derecha, arriba, abajo). Dada una matriz de enteros positivo, se debe devolver las coordenadas de **UN** elemento cumbre. Se asume que la matriz esta rodeada en su perímetro con el valor -1 en cada celda, como se muestra en la imagen. Por ejemplo, para el siguiente caso de prueba se tiene:




```
Input: matrix = [[10,20,15],[21,30,14],[7,16,32]]
Output: [1,1]
Explanation: 30 y 32 som ambos elementos cumbre entonces [1,1] y [2,2] son
respuesta validas.
```

La función `findPeakGrid` implementa un algoritmo en Python que soluciona el problema:

```python
def findPeakGrid(matrix: list[list[int]]):
        return _find_peak_grid(matrix, 0, len(matrix[0]) - 1)

def _find_peak_grid(matrix: list[list[int]], left: int, right: int):
        mid = left + (right - left) // 2
        max_row = 0
        for row in range(len(matrix)):
                if matrix[row][mid] > matrix[max_row][mid]:
                        max_row = row

        left_is_big = mid - 1 >= left
        and matrix[max_row][mid - 1] > matrix[max_row][mid]
        right_is_big = mid + 1 <= right
        and matrix[max_row][mid + 1] > matrix[max_row][mid]

        # compare mid to left and right
        if not left_is_big and not right_is_big:
        # found peak in the middle
                return [max_row, mid]
        elif left_is_big:
        # peak on the left
                return _find_peak_grid(matrix, left, mid)
        else:
        # peak on the right
                return _find_peak_grid(matrix, mid + 1, right)
```

(a) **(10pts)** Explique la idea de la solución del algoritmo, es decir por que el algoritmo sirve para dar solución al problema, **NO HAGA REFERENCIA A CÓDIGO**. El algoritmo utiliza una estrategia de D&C?

(b) **(5pts)** Proponga la función $T(n)$ para `findPeakGrid`.

(c) **(30pts)** Resuelva la ecuación de recurrencia del punto anterior. Si lo necesita, recuerde que la solución para $T(n) = T(n-1) + m$ es $T(n) = n * m$.

(d) **(5pts)** Basado en lo anterior, cual es la complejidad temporal de findPeakGrid?

2. Dadas dos strings (w1,w2) se quiere conocer cual es la longitud de la subsecuencia común mas larga entre w1 y w2. Por ejemplo, si w1='ABC' y w2='MATEOBASIC' la respuesta debería ser 3.

(a) **(10pts)** Identifique cuales pueden ser los subproblem repetidos para este problema, explique que estructura de datos le puede ayudar a no repetir problemas. De un ejemplo, expliquelo y justifique porque esa forma de guardar subproblemas puede ayudar a solucionar el problema.

(b) **(30pts)** Proponga un algoritmo `Python,Java,...` (**No es valido respuesta es 'palabras'**) basada en programación dinámica que solución el problema. Complejidad esperada $\mathcal{O}(|w1||w2|)$, el punto no es valido si no es alcanza esta complejidad.

(c) **(5pts)** Explique cual es la complejidad temporal de la solución del punto anterior.

(d) **(5pts)** La solución que propuso es Top-Down (TD) o Button-Up (BU)? Por que? Si su solución es TD que ventajas tiene contra una solución BU? Si su solución es BU que ventajas tiene contra una solución TD?