19th May 2023 Session 12

Exercise 1. (f19, Ex. 229)

Write an algorithm *Contains-Square(A)* that takes an $\ell \times \ell$ matrix A of numbers, and returns true if and only if A contains a square pattern of equal numbers, that is, a set of equal elements $A_{x,y}$ whose positions, interpreted as points with Cartesian coordinates (x, y), lay on the perimeter of a square. A square pattern consists of at least four elements, so a single number is not a valid square pattern. For example, the following matrix contains a square pattern consisting of elements with value 3. Notice in fact that there are two such square patterns.

```
    def Contains-Square(A: List[List[Int]]) -> bool:
        ...
```

```
A = [
[7, 8, 3, 8, 8, 3],
[7, 8, 3, 3, 3, 3],
[1, 3, 3, 5, 8, 3],
[7, 6, 3, 5, 3, 3],
[0, 4, 3, 3, 3, 3],
[9, 9, 1, 3, 7, 3]
]
```

Also, analyze the complexity of your solution as a function of $n = \ell^2$.

Exercise 2. Word Search

Given an `m x n` grid of characters `board` and a string `word`, return `true` *if* `word` *exists in the grid.* The word can be constructed from letters of sequentially adjacent cells, where adjacent cells are horizontally or vertically neighboring. The same letter cell may not be used more than once. Assume that `board` and `word` consist of only lowercase and uppercase English letters.

```
    def wordExists(board: List[List[str]], word: str) -> bool:
        ...
```

```
Input:
board = [
["A","B","C","E"],
["S","F","C","S"],
["A","D","E","E"]
],
word = "ABCCED"
Output = True

Input:
board = [
["A","B","C","E"],
["S","F","C","S"],
["A","D","E","E"]
],
word = "SEE"
Output = True

Input:
board = [
["A","B","C","E"],
["S","F","C","S"],
["A","D","E","E"]
],
word = "ABCB"
Output = False
```