DAVIDE IROVA

Ex. 1.1

```
DEF PEAK_ELEMENT(A):
    FOR i IN RANGE (LEN(A)):
        IF i >0 AND i <(LEN(A)-1) AND A[i] > A[i-1] AND A[i] > A[i+1]:
            RETURN i
        ELIF i==0 AND A[i] > A[i+1]:
            RETURN i
        ELIF i==(LEN(A)-1) AND A[i] > A[i-1]:
            RETURN i

    RETURN "NO PEAK ELEMENT"
```

Ex. 1.2:

COMPLEXITY OF 1.1 IS $\Theta(n)$ IN THE WORST CASE.
HERE IS THE SAME ALGORITHM BUT WITH COMPLEXITY $O(\log n)$:

```
DEF IS_PEAK(A, START, END):
    IF START == END:
        IF START == 0:
            IF A[START] > A[START+1]:
                RETURN START
            ELSE:
                RETURN -1
        ELIF START == LEN(A):
            IF A[START] > A[START-1]:
                RETURN START
            ELSE
                RETURN -1
    ELSE:
        LEFT_RES = IS_PEAK(A, START, END-1)
        IF LEFT_RES != -1:
            RETURN LEFT_RES
        RIGHT_RES = IS_PEAK(A, END+1, (LEN(A)-1))
        IF RIGHT_RES != -1:
            RETURN RIGHT_RES
    IF LEFT_RES == RIGHT_RES AND LEFT_RES == -1:
        RETURN -1
```

```
DEF PEAK_ELEMENT(A):
    RES = IS_PEAK(A, 0, (LEN(A)-1))
    IF RES == -1:
        RETURN "NO PEAK ELEMENT"
    ELSE
        RETURN RES
```

DAVIDE FAOVA

Ex. 2.1

```
COUNT_BATTLESHIPS(B):
    C = [FALSE] * LEN(B[0])
    FOR i IN RANGE (LEN(C)):
        C[i] = [FALSE] * LEN(B)
    VERTICAL_TOT = 0
    FOR n IN RANGE (LEN(B[0])):
        STATE = FALSE

        FOR m IN RANGE (LEN(B)):
            IF B[n][m] == 'O':
                IF STATE:
                    VERTICAL_TOT += 1
                    STATE = FALSE
            ELSE:
                C[n][m] = TRUE
                STATE = TRUE

    HORIZ_TOT = 0
    FOR m IN RANGE (LEN(B)):
        STATE = FALSE
        FOR n IN RANGE (LEN(B[0])):
            IF B[n][m] == 'O':
                IF STATE:
                    HORIZ_TOT += 1
                    STATE = FALSE
            ELIF B[n][m] == 'X' AND C[n][m] == FALSE:
                STATE = TRUE

    RETURN VERTICAL_TOT + HORIZ_TOT
```

Ex. 2.2

COMPLEXITY = $O((n \cdot m) + (m \cdot n)) = O(2nm) = O(n \cdot m)$

Ex. 3.1

ALGO-X RETURNS TRUE IF ALL THE ELEMENTS INSIDE THE ARRAY B ARE PRESENT IN THE ARRAY A.

IT DOES THAT BY COMPARING EACH ELEMENT OF B WITH ALL THE ELEMENTS IN A, AVOIDING THE ELEMENTS ALREADY FOUND.

BEST-CASE COMPLEXITY:

IF THE ELEMENT IN B IS NOT PRESENT IN THE ARRAY A THE ALGORITHM STOPS RETURNING FALSE.

$O(LEN(A))$, $\Omega(1)$

WORST-CASE COMPLEXITY:

ALL THE ELEMENTS OF B ARE INSIDE A.

$\Theta(LEN(B) * LEN(A))$

Ex. 3.2

```
DEF BETTER_ALGO_X(A,B):
    C=[0] * LEN(A)
    ATOTAL=0
    FOR i IN A:
        C[i] += 1
    FOR i IN RANGE(LEN(A)):
        IF C[i]!=0:
            ATOTAL += 1

    D=[0] * LEN(B)
    BTOTAL=0
    FOR i IN B:
        D[i] += 1
    FOR i IN RANGE(LEN(B)):
        IF D[i] !=0:
            BTOTAL += 1
    RETURN ATOTAL == BTOTAL
```

WORST-CASE: IF LEN(B) > LEN(A) $\Theta(LEN(B))$

BEST-CASE: IF LEN(A) > LEN(B) $\Theta(LEN(A))$

DAVIDE FROVA

DAVIDE FROVA

Ex. 4.1

```
DEF FIND_MIN_RECTANGLE (POINTS):
    POINTS.SORT()
    C = [ ] * LEN(POINTS)
    # FIND ELEMENTS THAT HAVE MORE THAN 1 ON SAME ROW
    FOR i IN RANGE (LEN(POINTS)):
        TEMP = 0
        FOR J IN RANGE (i+1, LEN(POINTS)):
            IF POINTS[i][0] == POINTS[J][0]:
                C.APPEND(POINTS[J])
                TEMP += 1
        IF TEMP != 0:
            C.APPEND(POINTS[i])

    # AFTER THIS I CAN FILTER OUT THE ONES THAT ARE ALONE IN
    # THEIR COLUMN.
    # THEN I FIND THE SMALLEST COMBINATION OF SAME ROW
    # THAT HAS A PAIRING COMBINATION IN ANOTHER ROW.
```