Beginselen van Programmeren
Exercise Session 9:
Complexity analysis

Ex 1. Maximum

Analyse the time complexity of the algorithm that calculates the maximum of a list. Can you think of a more efficient algorithm?

```
def maximum(a):
    i = 0
    max = a[i]
    while i < len(a):
        if a[i] > max:
            max = a[i]
        i = i + 1
    return max
```

Ex 2. Binary Search

In the previous exercise session you implemented a binary search algorithm. The following is a possible implementation. Analyse its time and space complexity.

```
def binary_search(sortedlist, needle):
    indexmin = 0
    indexmax = len(sortedlist)-1
    while indexmin < indexmax:
        indexmid = int(math.floor((indexmax + indexmin)/2.0))
        if needle <= sortedlist[indexmid]:
            indexmax = indexmid
        else:
            indexmin = indexmid + 1
    if needle == sortedlist[indexmax]:
        return indexmax
    else:
        return None
```

Ex 3. Polynomial

Compare the time complexity of the following three algorithms which compute the value of a polynomial for a certain x, in terms of the degree of the polynomial. The polynomial is represented as a list of coefficients. E.g. $[3, 5, 7]$ is the representation of the polynomial $3 + 5x + 7x^2$.

```
def eval1(coef, x):
    res = 0.0
    for i in range(0,len(coef)):
        term = float(coef[i])
        for j in range(0,i):
            term = term * x
        res = res + term
    return res
```

```
def eval2(coef, x):
    res = 0.0
    term = 1.0
    for i in coef:
        res = res + term * i
        term = term * x
    return res

def eval3(coef, x):
    res = float(coef[-1])
    for i in range(len(coef)-2, -1, -1):
        res = res * x + coef[i]
    return res
```

Ex 4. Count doubles

Write a program, that given a list of integers, counts all duplicate values that are present in the list. E.g. for the list $[4, 6, 3, 2, 4, 1, 8, 3, 8]$, there are 3 duplicate values: 4, 3, and 8.

If the list would contain more than 2 items with the same value, count each possible pair of duplicates. E.g. in the list $[2, 2, 1, 2, 2]$, count 6 duplicates.

What is the time complexity of your algorithm? Can you do better if the list would be sorted? Try implementing it.

Ex 5. Delta

Given a function for identifying if a sorted list contains two elements with a given delta (a numeric value for the difference between two numbers).

```
def containsdelta(lst, delta):
    j = 0
    for i in range(0, len(lst)):
        while (j < len(lst)-1) and (lst[i]-lst[j] > delta):
            j = j + 1
        if lst[i]-lst[j] == delta:
            return True
    return False
```

Estimate the time complexity of the function and justify your answer.

Ex 6. Saddle points

Given the code below. Analyse the time and space complexity of the algorithm in the function *get_saddle_points* in the assumption that the matrix m is a square $n \times n$ matrix.

```
def is_saddle (m, x, y):

    if (m[x][y] > m[x-1][y] and m[x][y] > m[x+1][y] and m[x][y] < m[x][y+1] and m[x][y] < m[x][y-1]) or\
        (m[x][y] < m[x-1][y] and m[x][y] < m[x+1][y] and m[x][y] > m[x][y+1] and m[x][y] > m[x][y-1]):
        return True
    else:
        return False

def get_saddle_points(m):

    saddles = []

    for i in range (1, len(m)-1):
        for j in range (1, len(m[i])-1):
            if is_saddle(m,i,j):
                saddles.append((i,j))

    return saddles
```

```
def main():

    m = [[1, 2, 3, 4, 5],
         [5, 3, 0, 1, 0],
         [6, 4, 2, 7, 8],
         [5, 7, 5, 4, 2],
         [4, 5, 1, 9, 1]
         ]

    saddle_points = get_saddle_points(m)

    for sdl in saddle_points:
        print(sdl)

main()
```