

Introduction to Computer Science, Winter Semester 2017
Practice Assignment 7

Discussion: 02.12.2017 - 07.12.2017

Exercise 7-1 To be discussed

Given the following three algorithms for finding the larger number among three numbers.

- Algorithm 1:

```
a = eval(input())
b = eval(input())
c = eval(input())

if (a >= b) and (a >= c):
    print(a)
if (b >= a) and (b >= c):
    print(b)
if (c >= b) and (c >= a):
    print(c)
```

- Algorithm 2:

```
a = eval(input())
b = eval(input())
c = eval(input())

if (a >= b):
    if (a >= c):
        print(a)
    else:
        print(c)
else:
    if (b >= c):
        print(b)
    else:
        print(c)
```

- Algorithm 3:

```
a = eval(input())
b = eval(input())
c = eval(input())
max = a

if (b > max):
    max = b
if (c > max):
    max = c
print(max)
```

- a) Compare the efficiency of the three algorithms. Please justify your answer.
- b) Determine the order of magnitude of the three algorithms.

Exercise 7-2 To be discussed

Given the following algorithms:

- a) Algorithm 1 computes the sum from 1 to n :

```
n = eval(input())
result = 0
i = 1
while (i <= n):
    result = result+i
    i = i+1
print(result)
```

- b) Algorithm 2 finds the smallest value in a list A_0, \dots, A_{n-1} .

```
list_A = eval(input())
k = len(list_A)
S = list_A[0]
i = 1
while (i < k):
    if (list_A[i] < S):
        S = list_A[i]
    i = i + 1
print(S)
```

- c) Algorithm 3 prints out 64, 32, 16, 8, 4, 2.

```
i = 64
while (i > 1):
    print(i)
    i = int(i/2)
```

Find the total number of executed instructions of the algorithms and determine their order of magnitude (the big-O).

Exercise 7-3

Find the total number of instructions and the order of magnitude of the following algorithms

- a) `import math`

```
m, n = eval(input()), eval(input())
a = ((m * m) - (n * n))
b = (2 * m * n)
c = (math.sqrt((a * a) + (b * b)))

print(" The Pythagorean Triple consists of the following sides: ")
print(a, b, c)
```

```

b) x, y, z = eval(input()), eval(input()), eval(input())
   if (x > 0):
       average = (x + y + z)/3
       print(average)
   else
       print("Bad data")
   endif

c) n = eval(input())
   F = [1, 1]
   i = 2
   while i < n:
       F = F + F[i-1] + F[i-2]
       print(F[i])
       i = i + 1

```

Exercise 7-4 To be discussed

Consider the following algorithm:

```

n = eval(input())
i = 1
sum = 0
while i <= n:
    sum = sum + (1/i - 1/(i+2))
    i = i + 4
print(sum)

```

- What is the output of the algorithm for $n = 10$? You do not need to calculate the final result.
- Calculate the total number of executed instructions of the algorithm and give its order of magnitude.

Exercise 7-5 To be discussed
Mystery

Consider the following algorithm:

```

n = eval(input())
m = eval(input())
y = 0

while(n>0):
    y += 1
    n -= 1

while(m>0):
    y += 1
    m -= 1
print(y)

```

- What is the output of the algorithm for $n = 5$ and $m = 3$?
- What is the functionality of the algorithm?
- Calculate the total number of executed instructions of the algorithm and give its order of magnitude.

Exercise 7-6 To be discussed

Find the total number of instructions and the order of magnitude of the following algorithm:

```
list_A = eval(input())
n = len(list_A)
i = 0
while (i < int(n/2)):
    tmp = list_A[i]
    list_A[i] = list_A[n-(i+1)]
    list_A[n-(i+1)] = tmp
    i = i + 1

print(list_A)
```

Exercise 7-7 To be discussed

Find the total number of instructions and the order of magnitude of the following algorithm and determine the best and worst case scenarios:

```
list_A = eval(input("Enter List"))
n = len(list_A)
x = eval(input("Enter Number"))
i = n - 1
c = 0
while(i>=0):
    if(list_A[i] < x):
        list_A[i] = 0
    else:
        list_A[i] = 1
        c +=1
    i-=1
print(list_A, ", ", c)
```

Exercise 7-8

Find the total number of instructions and the order of magnitude of the following algorithm and determine the best and worst case scenarios:

```
a = eval(input())
b = eval(input())
m = len(a)
k = len(b)
c = []
i = 0
if(m <= k):
    n = m
else:
    n = k
while(i < n):
    c = c + a[i]
    c = c + b[i]
    i += 1
if(i < m):
    while(i < m):
        c = c + a[i]
```

```
    i += 1
elif(i < k):
    while(i < k):
        c = c + b[i]
        i += 1
print(c)
```