Exercise sheet in Computational Complexity

Krzysztof Bartoszek

732A94 Advanced R Programming

Department of Computer and Information Science, Linköping University

16 October 2019 (Vallfarten)

The exercises here are translated from the below university scripts.

1. Giaro, K., 2011, Exercises in Computational Complexity of Algorithms (Złożoność obliczeniowa algorytmów w zadaniach, in Polish). Published by The Prof. Tadeusz Kotarbiński University of Informatics and Management, Olsztyn, Gdańsk.

2. Kubale, M., 1999. A Gentle Introduction to the Analysis of Algorithms (Łagodne wprowadzenie do analizy algorytmów, in Polish). Published by the Gdańsk University of Technology, Gdańsk.

Exercise 1
Show that

$$\sum_{i=2}^{n} \binom{i}{2} = \binom{n+1}{3} \qquad \text{and} \qquad \sum_{i=0}^{n} i^3 = \left(\frac{n(n+1)}{2}\right)^2$$

Exercise 2
Which of the below statements are true?

a) $(x^2+3x+1)^3 = o(x^6)$

b) $\frac{(\sqrt{x}+1)}{2} = o(1)$

c) $e^{1/x} = o(1)$

d) $\frac{1}{x} = o(1)$

e) $x^3(\log(\log x))^2 = o(x^3 \log x)$

f) $\sqrt{\log x + 1} = \Theta(\log \log x)$

g) $2 + \sin x = \Omega(1)$

h) $\frac{\cos x}{x} = O(1)$

i) $\int\limits_4^x \frac{\mathrm{d}t}{t} = O(\ln x)$

j) $\sum\limits_{j=1}^{x} \frac{1}{j^2} = O(1)$

k) $\sum\limits_{j=1}^{x} 1 = \Theta(x)$

l) $\int\limits_0^x e^{-t^2}\mathrm{d}t = O(1)$

Exercise 3
Arrange the below functions according to their growth rates for large n, that is each function belongs to $o(\cdot)$ of the next.

a) $2^{\sqrt{n}},\ e^{\log n^3},\ n^{3.01},\ 2^{n^2}$

b) $n^{1.6},\ 1+\log^3 n,\ \sqrt{n!},\ n^{\log n}$

c) $n^3 \log n,\ (\log\log n)^2,\ 2^n\sqrt{n},\ (n+4)^9$

d) $2^{\sqrt{\log n}},\ 2n,\ \sqrt{n},\ \log n,\ \log\log n,\ \frac{n}{\log n},\ \sqrt{n}\log n,\ \left(\frac{1}{3}\right)^n,\ \left(\frac{3}{2}\right)^n,\ 17,\ \left(\frac{n}{2}\right)^{\log n}$

Exercise 4
Find the growth rate (up to a constant) of the below expressions.

a) $\left(\frac{2}{3}\right)^n + \sum\limits_{i=1}^{n} \sin^2 n + n^2 + \ln\left(\sum\limits_{i=1}^{n} \binom{n}{i}\right)$

b) $\binom{n}{2} + \sum\limits_{i=1}^{n} \log n + n^2 \sin n$

Exercise 5
Does the function $f(n) = 2^{\sqrt{n}}$ grow faster than

a) $\log n$ but slower than n?

b) $\sqrt{n}$ but slower than n?

c) n but slower than n^2?

d) n^2 but slower than $\sqrt{2^n}$?

e) $\sqrt{2^n}$ but slower than 2^n?

Exercise 6
What is the computational complexity of the below code?

```
for i := 1 to √n do
    k := 1, l := 1
    while l < n do
        k := k + 2
        l := l + k
    end while
end for
```

Exercise 7
What is the computational complexity of the below code?

```
x := 0.0
for d := 1 to n do
    for g := d to n do
        sumvalue := 0.0
        for i := d to g do
            sumvalue := sumvalue + A[i]
        end for
        x := max(x, sumvalue)
    end for
end for
```

Exercise 8
The following algorithm calculates the value of a polynomial given as

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0.$$

```
p := a_0
xpower := 1
for i := 1 to n do
    xpower := x * xpower
    p := p + a_i * xpower
end for
```

How many multiplication operations need to be done in the worst case? How many summations?
Note: there exists an algorithm that requires only n multiplications and n summations. This is the fastest possible algorithm, called Horner's method.

Exercise 9
What is the computational complexity of the below code?

```
x := 0.0
for i := 1 to n − 1 do
    for j := i + 1 to n do
        for k := 1 to j do
            1 + 1
        end for
    end for
end for
```

Exercise 10
What is the computational complexity of the below code?

```
x := 0
for i := n − 1 down to 1 do
    if  i is odd then
        for j := 1 to i do
            1 + 1
        end for
        for k := i + 1 to n do
            x := x + 1
        end for
    end if
end for
```

Exercise 11
What is the computational complexity of the below code?

```
x := 0
for i := n − 1 down to 1 do
    if  i is odd then
        for j := 1 to i do
            for k := i + 1 to n do
                x := x + 1
            end for
        end for
    end if
end for
```

Exercise 12
Consider the following matrix

$$\mathbf{M} := \begin{bmatrix} \mathbf{1} & 2 & 3 & \dots & n-1 & n \\ \mathbf{1} & \mathbf{2} & 3 & \dots & n-1 & n \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{1} & \mathbf{2} & \mathbf{3} & \dots & \mathbf{n-1} & n \\ \mathbf{1} & \mathbf{2} & \mathbf{3} & \dots & \mathbf{n-1} & \mathbf{n} \end{bmatrix}$$

Propose an algorithm that calculates the sum of the lower triangular (including diagonal) elements of the matrix (i.e. those in bold) using

a) $\Theta(n^2)$ summations

b) $\Theta(n)$ summations

c) $O(1)$ summations

Recursive algorithms

Exercise 13
What is the computational complexity of the below code snippets and what do they return?

```
procedure FX1(n)
    if n == 1 then return 1
    else return FX(n − 1)+FX(n − 1)
    end if
end procedure
```

```
procedure FX2(n)
    if n == 1 then return 1
    else return 2·FX(n − 1)
    end if
end procedure
```

```
procedure FX3(n)
    if n == 1 then return 1
    else return FX(n − 1)
    end if
end procedure
```

```
procedure FX4(n)
    if n == 1 then return 1
    else return FX(n − 1)+1
    end if
end procedure
```

Exercise 14
What is the computational complexity of the below code?

```
procedure FX(n)
    for i := 1 to n do
        for j := 1 to n do
            print(i)
            print(n)
        end for
    end for
    if n > 1 then
        for i := 1 to 8 do
            FX(⌈n/2⌉)
        end for
    end if
end procedure
```

Exercise 15
What does the below code do and what is its computational complexity?

```
procedure FX(n)
    if n == 1 then return 1
    else return FX(FX(n − 1))+1
    end if
end procedure
```