Department of Informatics

Prof. Dr. Michael Böhlen

Binzmühlestrasse 14
8050 Zurich
Phone: +41 44 635 4333
Email: boehlen@ifi.uzh.ch

University of Zurich[UZH]

Informatics II — Midterm1

Spring 2019 — 22.03.2019

Name: ______________________ Matriculation number: ______________

Advice

You have 90 minutes to complete the exam of Informatics II. The following rules apply:

- Answer the questions in the space provided.
- Additional sheets are provided upon request. If you use additional sheets, put your name and matriculation number on each of them.
- Check the completeness of your exam (20 numbered pages).
- Use a pen in blue or black colour for your solutions. Pencils and pens in other colours are not allowed. Solutions written in pencil will not be corrected.
- Stick to the terminology and notations used in the lectures.
- Only the following materials are allowed for the exam:
  - One A4 sheet (2-sided) with your personal notes (handwritten/ printed/ photocopied).
  - A foreign language dictionary is allowed. The dictionary will be checked by a supervisor.
  - No additional items are allowed except a pocket calculator without text storage(memory) like TI-30 XII B/S. Computers, pdas, smart-phones, audio-devices or similar devices may not be used. Any cheating attempt will result in a failed test (meaning 0 points).
- Put your student legitimation card ("Legi") on the desk.

Signature:

Correction slot — Please do not fill out the part below

Exercise	1	2	3	4	Total
Points Achieved					
Maximum Points	10	17	15	8	50

Exercise 1 5+5 = 10 Points

Arrays and Sorting

1.1 Consider an array `A[1,...,n]` of integers. The array contains values between 0 and `m-1` and may contain duplicate elements. Implement an algorithm that prints the elements of array `A` in **ascending** order. The complexity of your solution must be $O(n+m)$. Use either C or pseudocode for your solution.

Example:

```
Input:  n=16, m=7, A=[5,0,2,4,3,6,1,1,5,5,0,6,0,0,2,4]
Output:  0 0 0 0 1 1 2 2 3 4 4 5 5 5 6 6
```

Hint: Use an auxiliary data structure for the frequencies of the elements in `A`.

Algorithm: CountingSort (A,n,m)

```
frequency[1..m] = [0,..,0];
for i = 1 to n do
    val = A[i];
    frequency[val+1]= frequency[val+1] + 1;
for i = 1 to m do
    for j = 1 to frequency[i] do
        print "i-1 ";
```

1.2 Consider algorithm B1 shown below. The input is an array `A[1,...,n]` of integers and $n \geq 2$.

```
Algorithm: B1(A,n)
1  index = n-1;
2  while index ≥ 1 do
3      if index == n then
4          index = index - 1;
5      if A[index] ≥ A[index + 1] then
6          index = index - 1;
7      else
8          tmp = A[index];
9          A[index] = A[index+1];
10         A[index+1] = tmp;
11         index = index + 1;
```

Answer the questions below. Write your solution into the answer boxes.

(a) What does algorithm B1 do?

Sorts the elements in descending order

(b) What is the asymptotic complexity of algorithm B1 in the worst case?

$\Theta(n^2)$

(c) What is the asymptotic complexity of algorithm B1 in the best case?

$\Theta(n)$

(d) What will algorithm B1 do if **line 5** is changed as follows?

if $A[index] \leq A[index+1]$ **then**

Sorts the elements in ascending order

Exercise 2	6+5+6 = 17 Points

Asymptotic Complexity

2.1 Calculate the simplest possible asymptotic tight bound for the following functions. Write your solution into the answer boxes.

a) $f_1(n) = \sqrt{n} + n \log_2 n + \log_2 n^2$

$f_1(n) = \sqrt{n} + n \log_2 n + \log_2 n^2$
$f_1(n) = \sqrt{n} + n \log_2 n + 2 \log_2 n$
$f_1(n) \in \Theta(n \log_2 n)$

$\boxed{\Theta(n \log_2 n)}$

b) $f_2(n) = \log_2 (8n^3 log_2 4) + \sqrt{n}$

$f_2(n) = \log_2 (8n^3 log_2 4) + \sqrt{n}$
$f_2(n) = \log_2 8 + \log_2 n^3 + \log_2 (\log_2 4) + \sqrt{n}$
$f_2(n) = 3 + 3 \log_2 n + \log_2 2 + \sqrt{n}$
$f_2(n) = 3 + 3 \log_2 n + 1 + \sqrt{n}$
$f_2(n) = 4 + 3 \log_2 n + \sqrt{n}$
$f_2(n) \in \Theta(\sqrt{n})$

$\boxed{\Theta(\sqrt{n})}$

c) $f_3(n) = 2^{7n} + 10n + \log_2 24$

$f_3(n) = 2^{7n} + 10n + \log_2 24$
$f_3(n) \in \Theta(128^n)$

$\boxed{\Theta(128^n)}$

d) $f_4(n) = 7n^{max(\log_2 n^{\log_2 4\cdot 2}, \sqrt{n})}$

$max(\log_2 n^{\log_2 4\cdot 2}, \sqrt{n}) = max(\log_2 n^{\log_2 8}, \sqrt{n})$
$max(\log_2 n^3, \sqrt{n}) = max(3\log_2 n, \sqrt{n}) => max = \sqrt{n}$
$f_4(n) = 7n^{max(\log_2 n^{\log_2 4\cdot 2}, \sqrt{n})}$
$f_4(n) = 7n^{\sqrt{n}}$
$f_4(n) \in \Theta(n^{\sqrt{n}})$

$\boxed{\Theta(n^{\sqrt{n}})}$

e) $f_5(n) = 1 + n^{\frac{\log_3 32}{\log_3 2}} + n^2 + n^3 + n^4$

$f_5(n) = 1 + n^{\frac{\log_3 32}{\log_3 2}} + n^2 + n^3 + n^4$
$f_5(n) = 1 + n^5 + n^2 + n^3 + n^4 \in \Theta(n^5)$

$\boxed{\Theta(n^5)}$

f) $f_6(n) = \log_{\log 5}(\log^{\log 100} n)$

$f_6(n) = \log_{\log 5}(\log^{\log 100} n)$
$f_6(n) = \log_{\log 5}(\log 100 \cdot \log n)$
$f_6(n) = \log_{\log 5}(\log 100) + \log_{\log 5}(\log n)$
$f_6(n) \in \Theta(\log(\log n))$

$\boxed{\Theta(\log(\log n))}$

Name: Matriculation number:

2.2 Calculate the asymptotic tight bound of the following recurrences. If the Master Theorem can be used write down the case (1-3). Write your solution into the answer boxes. Assume T(1) = 0 for all cases.

a) $T(n) = 5T(\frac{n}{7}) + (\log n)^2$
a=5, b=7, f(n)= $(\log n)^2$
Case 1: $T(n) = \Theta(n^{log_7 5})$

Case:	Complexity:
1	$\Theta(n^{log_7 5})$

b) $T(n) = T(n-3) + \frac{n}{3}$

$$\begin{aligned} T(n) &= T(n-3) + \frac{n}{3} \\ &= T(n-6) + \frac{n-3}{3} + \frac{n}{3} \\ &= T(n-6) + \frac{2n-3}{3} \\ &= T(n-9) + \frac{n-6}{3} + \frac{2n-3}{3} \\ &= T(n-9) + \frac{3n-9}{3} \\ &= T(n-12) + \frac{n-9}{3} + \frac{3n-9}{3} \\ &= T(n-12) + \frac{4n-18}{3} \\ &= \dots \\ \Rightarrow T(n) &= T(n-3k) + \frac{k \cdot n}{3} - \sum_{i=1}^{k} i - 1 \end{aligned}$$

k grows until it reaches $k = \lfloor \frac{n}{3} \rfloor$, thus we have:

$$T(n) = \frac{n}{3} \cdot \frac{n}{3} - \sum_{i=1}^{\frac{n}{3}} i - 1 = \frac{n^2}{9} - \frac{n-3}{18} \cdot n = \frac{n^2}{9} - \frac{n^2}{18} + \frac{1}{6} = \frac{n^2}{18} + \frac{1}{6} \in \Theta(n^2)$$

Case:	Complexity:
-	$\Theta(n^2)$

c) $T(n) = 2T(\frac{n}{3}) + n \log n$
a=2, b=3, f(n)= $n \log n$
Case 3: $T(n) = \Theta(n \log n)$

Case:	Complexity:
3	$\Theta(n \log n)$

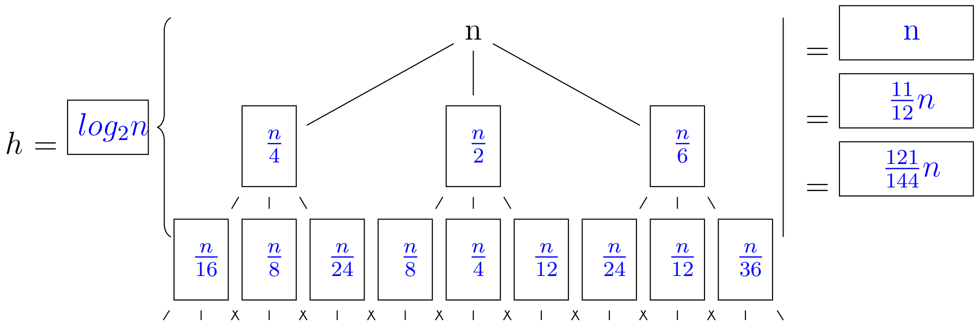
d) $T(n) = 8T(\frac{n}{2}) + n^3$
a=8, b=2, f(n)= n^3
Case 2: $T(n) = \Theta(n^3 \log_2 n)$

Case:	Complexity:
2	$\Theta(n^3 \log_2 n)$

2.3 Consider the recurrence:

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ T(n/4) + T(n/2) + T(n/6) + n & \text{if } n > 1 \end{cases}$$

Finish a recursion tree and use it to estimate the asymptotic upper bound of $T(n)$. Write your solutions into the answer boxes.



Complexity: $\boxed{\Theta(n)}$

Tree grows until $\left(\frac{1}{2}\right)^h n = 1 \implies h = \log_2 n$ To get an upper bound, we can use the sum $n \sum_{h=0}^{h} \left(\frac{11}{12}\right)^h$.
Guess: $O(n)$

Exercise 3 — $1+3+4+7 = 15$ Points

Runtime Analysis

Suppose `A[1,...,n]` is a **sorted** array of **unique integers**. A *fixed point* of an array is an index $i \in \{1, ..., n\}$ so that $A[i] = i$.

The goal of the following program is to return `True` if there is a fixed point, and to return `False` otherwise.

Function Call: isThereAFixedPoint(A,1,n)

Algorithm: isThereAFixedPoint(A, lower, upper)

mid = $\lfloor$(lower + upper)/2$\rfloor$;
if $A[mid] == mid$ **then**
  return *True;*
if $lower == upper$ **then**
  return *False;*
if $A[mid] > mid$ **then**
  return $isThereAFixedPoint(A, lower, mid)$;
if $A[mid] < mid$ **then**
  return $isThereAFixedPoint(A, mid + 1, upper)$;

3.1 What is the fixed point in array $B = [-1, -2, 0, 4, 5, 7]$

4 or 5

3.2 Suppose the algorithm `isThereAFixedPoint` is applied to input array [-2,-1,2,3,4,5,7,14]. What are values of `lower`, `upper`, `mid`, `A[mid]` each time after `isThereAFixedPoint` has executed `line 1`?

No.	**lower**	**upper**	**mid**	**A[mid]**
1	1	8	4	3
2	5	8	6	5
3	7	8	7	7

3.3 Consider the worst-case runtime of `isThereAFixedPoint`. Specify the recurrence relation and calculate the asymptotic complexity of `isThereAFixedPoint` for this case.

Reccurence relation:

$\boxed{\mathrm{T(n) = T(n/2) + \Theta(1)}}$

Complexity:

$\boxed{O(\log n)}$

The algorithm satisfies recurrence relation:
$\mathrm{T(n) = T(n/2) + \Theta(1)}$

because in each call to isThereAFixedPoint we recur on a set of size about n/2, and then there is $\Theta(1)$ overhead to increment the pointer. By, for example, Master Theorem, this means
$\mathrm{T(n)} \in O(\log n)$

3.4 Do an exact analysis and calculate the asymptotic tight bound of Algorithm D2.

Algorithm: D2(n)

```
1 result = 0;
2 for i = 1 to n do
3     sum = 0;
4     for j = 1 to i − 1 do
5         if i mod j == 0 then
6             sum = sum + j;
7     if sum == i then
8         result = result + 1;
9 return result;
```

line number	cost	number of times executed
line 1	c_1	1
line 2	c_2	$n+1$
line 3	c_3	n
line 4	c_4	$\sum_{i=1}^{n}(\sum_{j=1}^{i} 1) = n\frac{(n+1)}{2}$
line 5	c_5	$n\frac{(n-1)}{2}$
line 6	c_6	$\alpha\frac{n}{2}(n-1)$
line 7	c_7	n
line 8	c_8	βn
line 9	c_9	1

* $0 \leq \alpha \leq 1$, $0 \leq \beta \leq 1$

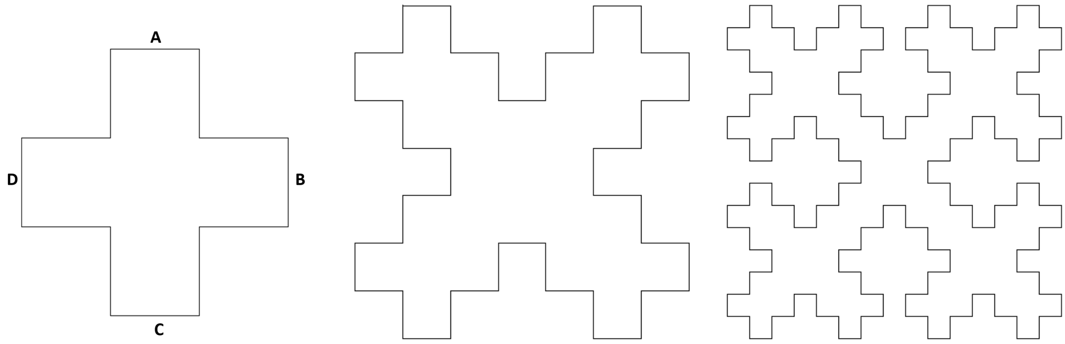
$$\mathrm{T}(n) \in \boxed{\Theta(n^2)}$$

$$\mathrm{T}(n) = c_1 + (n+1)c_2 + nc_3 + n\frac{(n+1)}{2}c_4 + n\frac{(n-1)}{2}c_5 + \alpha\frac{n}{2}(n-1)c_6 + nc_7 + \beta nc_8 + c_9$$

Exercise 4 2+4+2 = 8 Points

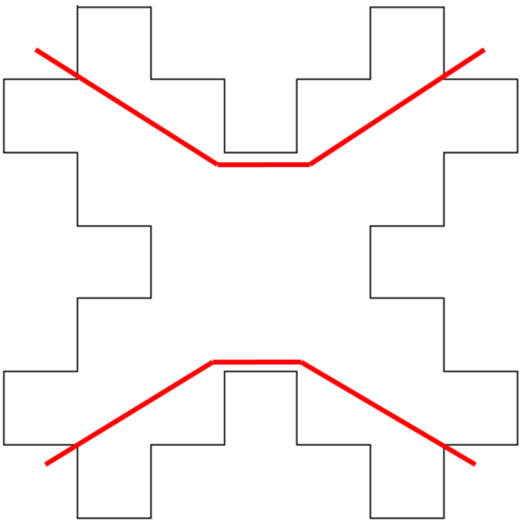
Divide and Conquer

Consider the space filling curves of, respectively, order 0, order 1 and order 2, illustrated below.

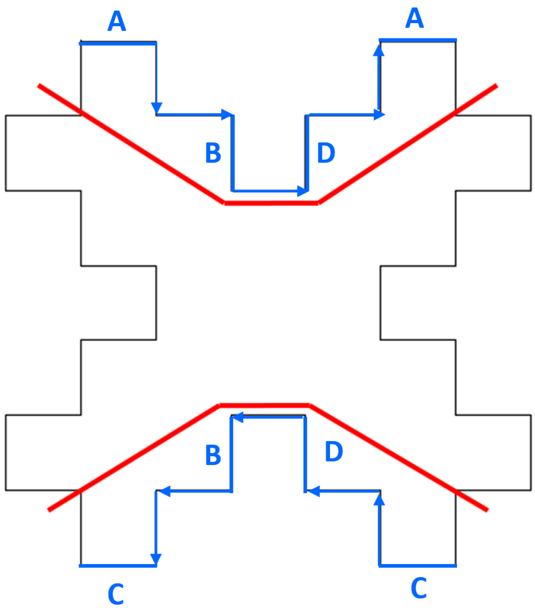


4.1 Illustrate in the indicated parts in figures below, how curves of order i are composed to curves of order i+1. Draw directly into the figures and label multi-line segments with, respectively, A, B, C and D to explain the solution. Precisely denote start and beginning of multi-line segments.

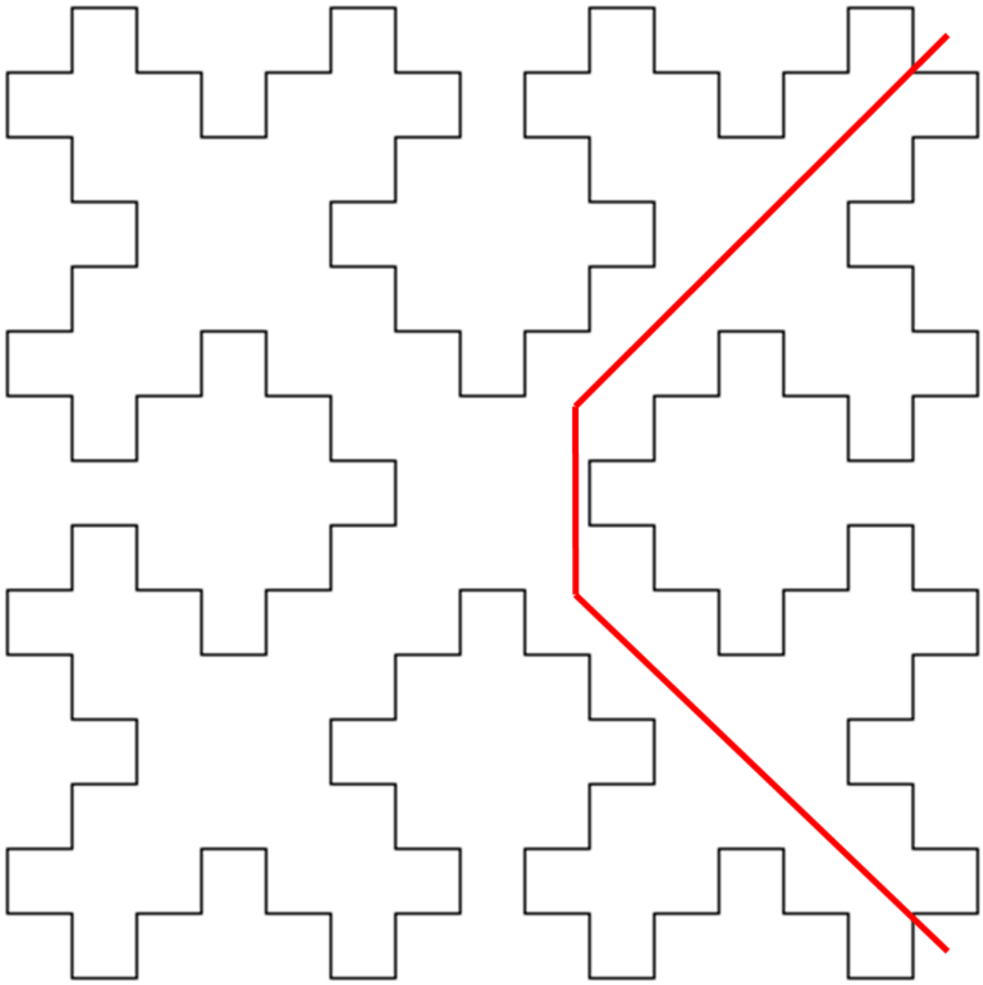
Order 1:



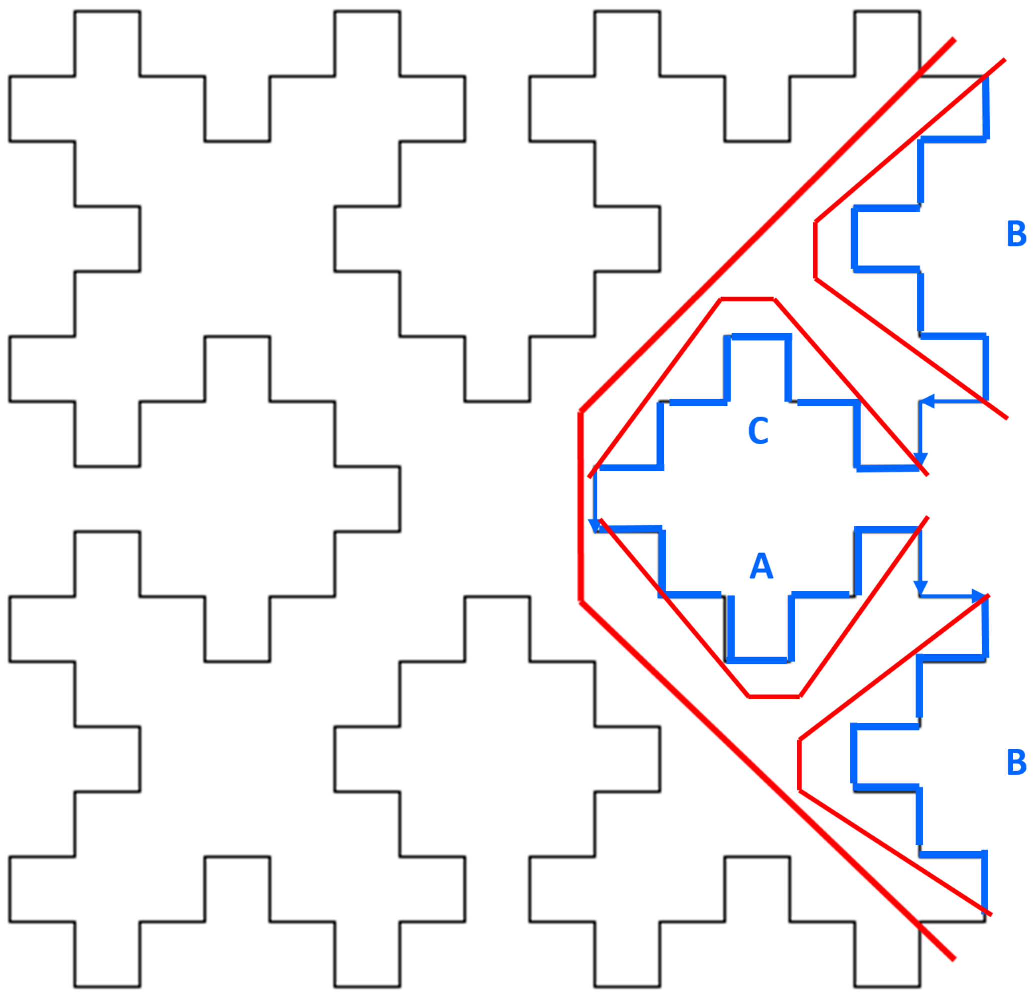
Order 1:



Order 2:



Order 2:



4.2 Recursively define curve C, i.e., define C of order `i+1` in terms of curves of order `i`.

$$C = C \uparrow \leftarrow D \leftarrow B \leftarrow \downarrow C$$