Self-Study Session 01

Algorithms and Data Structures (DAT2, SW2, DV2)

Instructions. You have to **work individually** on these questions from 8:15 to 10:00. From 10:00 to 12:00 you can join your group and discuss your solutions togther.

- Read carefully the text of each exercise before solving it! Pay particular attentions to the terms in bold.
- If a question seems ambiguous, write under which assumptions you are solving it.
- You can use **CLRS** to refer to the textbook T.H. Cormen, Ch. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms* (3rd edition) in your answers.
- Make an effort to use a readable handwriting and to present your solutions neatly.
- The TAs will be available from 10:00 to 12:00. Use the digital trashcan to ask for help.
- The points relative to each question give an indication of their complexity relative to this exercise sheet. These points **may not** correspond to the amount of points you may get for a similar exercise at the exam.

Question 1. 20 Pts

Identifying asymptotic notation. (Note: lg means logarithm in base 2)

(1.1) [5 Pts] Mark **ALL** the correct answers. $\lg n^2 + \lg 2^n + 1^n + \sqrt{n}$ is

☐ **b)** $\Theta(\lg n)$ ☐ **b)** $\Theta(n)$ ☐ **c)** $\Theta(\sqrt{n})$ ☐ **d)** $\Theta(n^2)$ ☐ **e)** $\Theta(1^n)$

(1.2) [5 Pts] Mark **ALL** the correct answers. $\lg n^2 + \lg 2^n + 1^n + \sqrt{n}$ is

☐ **a)** $O(\lg n)$ ☐ **b)** $O(n)$ ☐ **c)** $O(\sqrt{n})$ ☐ **d)** $O(n^2)$ ☐ **e)** $O(1^n)$

(1.3) [5 Pts] Mark **ALL** the correct answers. $558 \cdot n \lg n^2 + 0.0001 \cdot n^3 + (n \lg n)^2$ is:

☐ **a)** $\Theta(n \lg n)$ ☐ **b)** $\Theta(n^3)$ ☐ **c)** $\Theta(n^2)$ ☐ **d)** $\Theta(n^2 \lg n)$ ☐ **d)** $\Omega(n^2 \lg n)$

(1.4) [5 Pts] Mark **ALL** all the functions below that satisfy $\lg(f(n)) = \Theta(\lg n)$.

a) **a)** $f(n) = n^2$ b) **b)** $f(n) = 2^n$ c) **c)** $f(n) = 2^n \cdot n^2$ d) **d)** $f(n) = \sum_{i=1}^{n} i$

Solution 1.

(1.1)

$$\lg n^2 + \lg 2^n + 1^n + \sqrt{n} = 2 \lg n + n + 1 + n^{1/2} = \Theta(n)$$

Therefore only **b** is correct.

(1.2) We have seen that $\lg n^2 + \lg 2^n + 1^n + \sqrt{n} = \Theta(n)$. Therefore both **b** and **d** are correct.

(1.3)

$$558 \cdot n \lg n^2 + 0.0001 \cdot n^3 + (n \lg n)^2 = 2 \cdot 558 \cdot n \lg n + 0.0001 \cdot n^3 + n^2 \lg^2 n = \Theta(n^3)$$

Therefore **b** and **d** are correct.

(1.4) The correct answers are **a** and **d** as demonstrated below

$$\lg n^2 = 2 \lg n = \Theta(\lg n) \quad (\mathbf{a})$$

$$\lg 2^n = n = \Theta(n) \quad (\mathbf{b})$$

$$\lg 2^n \cdot n^2 = \lg 2^n + \lg n^2 = n + 2 \lg n = \Theta(n) \quad (\mathbf{c})$$

$$\lg(\textstyle\sum_{i=1}^{n} i) = \lg(\frac{n(n+1)}{2}) = (\lg n + \lg(n+1)) - 1 = \Theta(\lg n) \quad (\mathbf{d})$$

Question 2. 20 Pts

Consider the following recurrences

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ T(n/2) + n^4 & \text{if } n > 1 \end{cases} \qquad Q(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 9 \cdot Q(n/3) + n^{1.9} & \text{if } n > 1 \end{cases}$$

(2.1) [5 Pts] Mark **ALL** correct answers.

- ☐ **a)** $T(n)$ can be solved using Case 1 of the Master Theorem
- ☐ **b)** $T(n)$ can be solved using Case 2 of the Master Theorem
- ☐ **c)** $T(n)$ can be solved using Case 3 of the Master Theorem
- ☐ **d)** $T(n)$ cannot be solved using the Master Theorem

(2.2) [5 Pts] Mark **ALL** correct answers.

- ☐ **a)** $T(n) = \Theta(n^4 \lg n)$
- ☐ **b)** $T(n) = \Theta(n^2)$
- ☐ **c)** $T(n) = \Omega(n^3)$
- ☐ **d)** $T(n) = O(n^4)$

(2.3) [5 Pts] Mark **ALL** correct answers.

- ☐ **a)** The recurrence Q is in the form $Q(n) = a\,Q(n/b) + f(n)$ for some a, b and $f(n)$.
- ☐ **b)** $Q(n)$ can be solved using Case 1 of the Master Theorem
- ☐ **c)** $Q(n)$ can be solved using Case 4 of the Master Theorem
- ☐ **d)** $Q(n)$ cannot be solved using the Master Theorem

(2.4) [5 Pts] Mark **ALL** correct answers.

- ☐ **a)** $Q(n) = \Theta(n^{1.9} \lg n)$
- ☐ **b)** $Q(n) = \Omega(n^{1.9})$
- ☐ **c)** $Q(n) = \Theta(n^2)$
- ☐ **d)** $Q(n) = O(n^3)$

Solution 2.

(2.1) Note that the recurrence is of the form $T(n) = aT(n/b)+f(n)$ where $a = 1$, $b = 2$, and $f(n) = n^4$. We can solve the recurrence using the master method. This recurrence falls into the third case, because $f(n) = n^4 = \Omega(n) = \Omega(n^{\log_b a} + \epsilon)$ for $\epsilon = 1$. Moreover the "regularisation" condition $af(n/b) \leq cf(n)$ holds for $c = 1/16$ as shown below

$$af(n/b) = \left(\frac{n}{2}\right)^4 = \frac{1}{16}n^4 = cn^4\,.$$

Therefore, by the Master Theorem (Case 3) we can conclude that $T(n) = \Theta(f(n)) = \Theta(n^4)$.

(2.2) As proven before $T(n) = \Theta(n^4)$, therefore **c** and **d** are the correct answers.

(2.3) Note that the recurrence is of the form $Q(n) = aQ(n/b) + f(n)$ where $a = 9$, $b = 3$, and $f(n) = n^{1.9}$. We can solve the recurrence using the master method. This recurrence falls into the first case of the Master theorem because $f(n) = n^{1.9} = O(n^{1.9}) = O(n^{\log_b a} - \epsilon)$ for $\epsilon = 0.1$. Therefore, by the Master Theorem (Case 1) we can conclude that $Q(n) = \Theta(n^{\log_b a}) = \Theta(n^2)$.

(2.4) As proven before $Q(n) = \Theta(n^2)$, therefore, **b**, **c**, and **d** are correct answers.

Question 3. 20 Pts

(3.1) [5 Pts] Consider the INSERTION-SORT algorithm on the following input $A = [91, 71, 29, 43, 97, 59, 17, 93, 61, 13]$. Select among the followings, the configuration of the array after **four** iterations of the main loop of the INSERTION-SORT algorithm have been performed.

☐ **a)** $[29, 43, 71, 91, 97, 59, 17, 93, 61, 13]$ ☐ **b)** $[3, 17, 29, 43, 59, 61, 71, 91, 97, 93]$

☐ **c)** $[71, 29, 43, 59, 17, 61, 13, 91, 97, 93]$ ☐ **d)** $[29, 43, 91, 71, 97, 59, 17, 93, 61, 13]$

☐ **e)** None of the above is correct.

(3.2) [5 Pts] Consider an execution of MERGE-SORT$(A, 1, A.length)$ on the array $A = [7, 3, 5, 9, 8, 2]$. Select among the following options, the configuration of the array after **three** calls of the subroutine MERGE have been performed.

☐ **a)** $[5, 3, 7, 9, 8, 2]$ ☐ **b)** $[7, 3, 5, 9, 8, 2]$ ☐ **c)** $[3, 5, 7, 8, 9, 2]$

☐ **d)** $[2, 3, 5, 7, 8, 9]$ ☐ **e)** None of the above is correct.

(3.3) Consider the algorithm STACKSTUFF, which takes as input an integer $n > 2$ and performs some operations on a stack S which is initially empty.

```
STACKSTUFF(n)
1  Initialise an empty stack S
2  for i = 1 to n − 2
3       for j = n downto i
4            PUSH(S,i)
5  for k = 1 to n^2
6       POP(S)
```

(a) [5 Pts] What is the asymptotic running time of STACKSTUFF(n)? Justify your answer.

(b) [5 Pts] What is the size of the stack S after the execution of STACKSTUFF(n)?

☐ **a)** $\Theta(1)$ ☐ **b)** $\Theta(n)$ ☐ **c)** $\Theta(n \lg n)$ ☐ **d)** $\Theta(n^2)$ ☐ **e)** $\Theta(n^3)$

Solution 3.

(3.1) The correct answer is **a**.

(3.2) The correct answer is **c**.

(3.3) (a) The loop in lines 2–4 runs in $\Theta(n^2)$, and the loop in lines 5–6 runs in $\Theta(n^2)$. Therefore the overall running time is $\Theta(n^2)$.

(b) Note that the number of elements pushed in S during the execution of lines 2–4 is smaller than n^2. Therefore after performing n^2 POP operations the stack S is empty. Therefore, at the end of the execution of STACKSTUFF(n), the size of S is $\Theta(1)$. Hence, the correct answer is **a**.

Question 4. 40 Pts

The Fibonacci sequence is the series of numbers $1, 1, 2, 3, 5, 8, 13, 21, \ldots$ where the next number in the sequence is found by adding up the two numbers before it. The n-th Fibonacci number in the sequence is typically defined by the following recurrence (see CLRS pp. 99) for $n \in \mathbb{N}$

$$F(n) = \begin{cases} 1 & \text{if } n \in \{0, 1\}, \\ F(n-1) + F(n-2) & \text{if } n > 1 \end{cases}$$

(a) [15 Pts] Consider the following iterative algorithm FIB-ITER(n) which computes the n-th Fibonacci number. Prove that FIB-ITER(n) is correct.

```
FIB-ITER(n)
1  Initialise the array F[1..n+1]
2  F[1] = 1
3  F[2] = 1
4  for i = 3 to n+1
5      F[i] = F[i-2] + F[i-1]
6  return F[n+1]
```

(b) [5 Pts] What is the asymptotic running time of FIB-ITER(n)? Justify your answer.

(c) [20 Pts] Consider the recursive algorithm FIB-REC(n) which computes $F(n)$ by implementing its recurrence. Prove that the running time of FIB-REC(n) is $O(2^n)$.

```
FIB-REC(n)
1  if n < 2
2      return 1
3  else
4      return FIB-REC(n-1) + FIB-REC(n-2)
```

Solution 4.

(a) For proving the correctness of FIB-ITER we use the following loop invariant:

> At the beginning of each iteration of the for-loop of lines 4–5, $F[j] = F(j-1)$ for all $j = 1, .. i-1$

Initialisation Before the first loop iteration $i = 3$, $F[1] = F(0)$ and $F[2] = F(1)$, and the loop invariant trivially holds.

Maintenance In line 5 it is performed the assignment $F[i] = F[i-2] + F[i-1]$. Since the loop invariant holds right before starting the current iteration, we have that

$$F[i-2] + F[i-1] = F(i-3) + F(i-2) = F(i-1)\,.$$

Hence, after executing line 5 $F[i] = F(i-1)$. Therefore, after incrementing i the invariant is restored.

Termination When the loop terminates $i = n+2$. According to the loop invariant we have $F[i-1] = F[n+2-1] = F[n+1] = F(n)$.

Then, we can conclude the algorithm correctly returns $F(n)$.

(b) The initialisation of F in lines 1–3 takes $\Theta(1)$. The for-loop of lines 4–5 iterates $\theta(n)$ times and each iteration takes constant time. Therefore, the overall running time of FIB-ITER is $\Theta(n)$. Note that the algorithm uses $\Theta(n)$ space to store the Fibonacci numbers $F(0), \ldots, F(n)$ in an array.

We leave as exercise to adapt the algorithm so that it only uses $\Theta(1)$ space to perform the same task.

(c) The running time of FIB-REC(n) is described by the following recurrence

$$T(n) = \begin{cases} d_0 & \text{if } n \in \{0,1\} \\ T(n-1) + T(n-2) + d_1 & \text{if } n > 1 \end{cases}$$

for some constants $d_0, d_1 > 0$.

We show, by induction on n, that $T(n) \leq c2^n - d$ for some constants $c, d > 0$.

Base Case ($n \in \{0,1\}$): Assuming that $c \geq d + d_0$ we get

$$T(0) = d_0 \leq c2^0 - d = c - d\,, \qquad \text{and} \qquad T(1) = d_0 \leq c2^1 - d = 2c - d\,.$$

Inductive Step ($n > 1$): Assume that the inductive hypothesis holds for all $m < n$. Then we have

$$\begin{aligned} T(n) &= T(n-1) + T(n-2) + d_1 && \text{(def. } T\text{)} \\ &\leq 2T(n-1) + d_1 && (T(i+1) > T(i) \text{ for all } i \in \mathbb{N}) \\ &\leq 2(c2^{n-1} - d) + d_1 && \text{(Inductive hypothesis)} \\ &= c2^n - 2d + d_1 \\ &\leq c2^n - d && \text{(assume } d \geq d_1\text{)} \end{aligned}$$

Therefore, for $c \geq d + d_0$ and $d \geq d_1$ we have that $T(n) \leq c2^n - d$ for all $n \in \mathbb{N}$. This proves that $T(n) = O(2^n)$.