Self-Study Session 02

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. $n \lg n^5 + n \lg 2^n + n\sqrt{n}$ is

☐ **b)** $\Theta(n \lg n)$ ☐ **b)** $\Theta(n)$ ☐ **c)** $\Theta(\sqrt{n})$ ☐ **d)** $\Theta(n^2)$ ☐ **e)** $\Theta(n^{1.5})$

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

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

(1.3) [5 Pts] Mark **ALL** the correct answers. $700 \cdot n^2 + \frac{n^2 \lg n}{999} + \lg n^n$ is:

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

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

☐ **a)** $f(n) = n^2$, $g(n) = n^2$ ☐ **b)** $f(n) = 2^n$, $g(n) = 2^n$

☐ **c)** $f(n) = 2^n$, $g(n) = 4^n$ ☐ **d)** $f(n) = n \lg n$, $g(n) = 2^n$

Solution 1.

(1.1)

$$n \lg n^5 + n \lg 2^n + n\sqrt{n} = 5n \lg n + n^2 + n^{1.5} = \Theta(n^2)$$

Therefore only **d** is correct.

(1.2) We have seen that $n \lg n^5 + n \lg 2^n + n\sqrt{n} = \Theta(n^2)$. Therefore **a**, **c**, and **d** are correct.

(1.3)

$$700 \cdot n^2 + \frac{n^2 \lg n}{999} + \lg n^n = 700 \cdot n^2 + \frac{1}{999} n^2 \lg n + n \lg n = \Theta(n^2 \lg n)$$

Therefore **a**, **b**, **d**, and **e** are correct.

(1.4) The correct answers are **b**, **c**, and **d** as demonstrated below

$$\lg(n^2 \cdot n^2) = \lg n^4 = 4 \lg n = \Theta(\lg n) \tag{\textbf{a}}$$

$$\lg(2^n \cdot 2^n) = \lg(2^{2n}) = 2n = \Theta(n) \tag{\textbf{b}}$$

$$\lg(2^n \cdot 4^n) = \lg(2^n \cdot 2^{2n}) = \lg(2^{3n}) = 3n = \Theta(n) \tag{\textbf{c}}$$

$$\lg(n \lg n \cdot 2^n) = \lg n + \lg \lg n + n = \Theta(n) \tag{\textbf{d}}$$

Question 2. 20 Pts

Consider the following recurrences

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \in \{0,1\} \\ T(n-1) + T(n-2) + \Theta(1) & \text{if } n > 1 \end{cases} \qquad Q(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 8 \cdot Q(n/2) + n^4 & \text{if } n > 1 \end{cases}$$

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

- ☐ **a)** $T(n)$ can be solved using Case 1 of the Master Theorem
- ☐ **b)** $Q(n)$ can be solved using Case 2 of the Master Theorem
- ☐ **c)** $Q(n)$ 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)** $Q(n) = \Theta(n^2 \lg n)$
- ☐ **b)** $Q(n) = \Omega(n)$
- ☐ **c)** $Q(n) = \Theta(n^4)$
- ☐ **d)** $Q(n) = O(n^3)$

(2.3) [10 Pts] Prove that $T(n) = O(2^n)$ using the substitution method.

Solution 2.

(2.1) In the next point we show that $Q(n)$ can be solved using the Case 3 of the Master Theorem. In contrast, the recurrence $T(n)$ does not comply with the format required for the Master Theorem. Therefore correct answers are **c** and **d**.

(2.2) Note that the recurrence is of the form $Q(n) = aQ(n/b) + f(n)$ where $a = 8$, $b = 2$, and $f(n) = n^4$. We can solve the recurrence using the master method. This recurrence falls into the third case, because $f(n) = n^4 = \Omega(n^4) = \Omega(n^{\log_b a + \epsilon})$ for $\epsilon = 1$. Moreover the "regularisation" condition $af(n/b) \le cf(n)$ holds for $c = 1/2$ and $n \ge 1$ as shown below

$$af(n/b) = 8f(n/2) = 8\left(\frac{n}{2}\right)^4 = \frac{8}{16}n^4 = cn^4 \,.$$

By the Master Theorem (Case 3) we can conclude that $Q(n) = \Theta(f(n)) = \Theta(n^4)$.

Therefore, the correct answers are **c** and **b**.

(2.3) Note that we have already seen this recurrence in the first self-study session. We can rewrite $T(n)$ making explicit the constants hidden behind the Θ notation.

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

for some constants $d_0, d_1 > 0$.

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

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

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

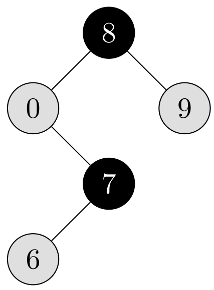
Inductive Step ($n > 1$): Assume that the inductive hypothesis holds for all $m < n$. Then we have

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

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

Question 3. 20 Pts

(3.1) [4 Pts] Mark **ALL** the correct statements. Consider the tree T depicted below



☐ **a)** The height of T is 3

☐ **b)** T satisfies the binary-search tree property

☐ **c)** T satisfies the red-black tree property (NIL nodes are omitted)

☐ **d)** T satisfies the max-heap property

☐ **e)** T corresponds to the array $[8, 0, 9, 7, 6]$ interpreted as a binary tree.

(3.2) [6 Pts] Consider the hash table $H = \text{NIL}, \text{NIL}, 46, 35, \text{NIL}, 15, 92, \text{NIL}, 52, \text{NIL}, 87$. Insert the keys $44, 84, 17, 20$ in H using *open addressing* with the auxiliary function $h'(k) = k$.

Mark the hash table resulting by the insertion of these keys using linear probing.

☐ **a)** $44, 17, 46, 35, \text{NIL}, 15, 92, 84, 52, 20, 87$ ☐ **b)** $44, \text{NIL}, 46, 35, 17, 15, 92, 84, 52, 20, 87$

☐ **c)** $44, 20, 46, 35, \text{NIL}, 15, 92, 84, 52, 17, 87$ ☐ **d)** none of the above

Mark the hash table resulting by the insertion of these keys using quadratic probing with $c_1 = 2$ and $c_2 = 4$.

☐ **a)** $44, 17, 46, 35, \text{NIL}, 15, 92, 84, 52, 20, 87$ ☐ **b)** $44, \text{NIL}, 46, 35, 17, 15, 92, 84, 52, 20, 87$

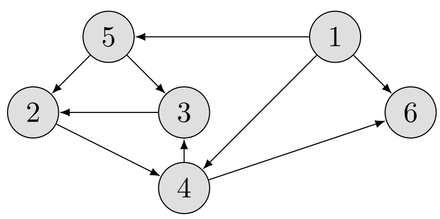
☐ **c)** $44, 20, 46, 35, \text{NIL}, 15, 92, 84, 52, 17, 87$ ☐ **d)** none of the above

Mark the hash table resulting by the insertion of these keys using double hashing with $h_1(k) = k$ and $h_2(k) = 1 + (k \mod (m-1))$.

☐ **a)** $44, 17, 46, 35, \text{NIL}, 15, 92, 84, 52, 20, 87$ ☐ **b)** $44, \text{NIL}, 46, 35, 17, 15, 92, 84, 52, 20, 87$

☐ **c)** $44, 20, 46, 35, \text{NIL}, 15, 92, 84, 52, 17, 87$ ☐ **d)** none of the above

(3.3) [10 Pts] Consider the directed graph G depicted below.



(a) Write the intervals for the discovery time and finishing time of each vertex in the graph obtained by performing a depth-first search visit of G. *Remark:* If more than one vertex can be chosen, choose the one with smallest label.

(b) Write the corresponding "parenthesization" of the vertices in the sense of Theorem 22.7 in CLRS

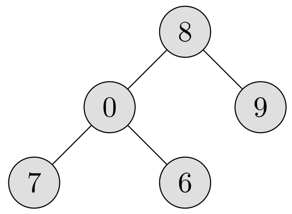
(c) Assign to each edge a label T (tree edge), B (back edge), F (forward edge), or C (cross edge) corresponding to the classification of edges induced by the DFS visit performed before.

(d) Show the components computed by STRONGLY-CONNECTED-COMPONENTS(G).

Solution 3.

(3.1) The height of the tree is 3 since its longest path from the root to a leaf has 3 edges. The tree satisfies the binary search tree property. Instead, it does not red-black tree property (under the assumption that NIL nodes are omitted) because the path from the root node 8 the node 9 has fewer black nodes than the path from 8 to 6. T does **not** satisfy the max-heap property because for instance the node 9 is a child than the node 8.

Finally, the binary tree interpretation of the array $[8, 0, 9, 7, 6]$ is



Therefore, the correct answers are **a** and **b**.

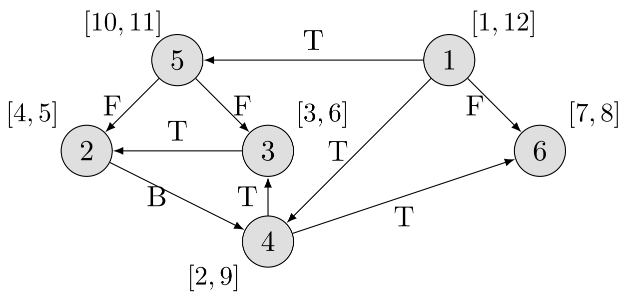
(3.2) The resulting hash tables are respectively:

linear probing: $44, 20, 46, 35,$ NIL, $15, 92, 84, 52, 17, 87$

quadratic probing: $44, 17, 46, 35,$ NIL, $15, 92, 84, 52, 20, 87$

double hashing: $44,$ NIL, $46, 35, 17, 15, 92, 84, 52, 20, 87$

(3.3) The correct answers for (a) and (c) are depicted in the graph below. There, each vertex $v \in V$ is associated with the interval $[v.d, v.f]$ as computed by DFS, and each edge is labelled according to the corresponding classification.



(b) the corresponding "parenthesization" of the vertices is

(1 (4 (3 (2 2) 3) (6 6) 4) (5 5) 1).

(d) After running STRONGLY-CONNECTED-COMPONENTS(G) we obtain the following strongly connected components: $\{\{1\}, \{5\}, \{4, 2, 3\}, \{6\}\}$.

Question 4. 20 Pts

A student participates in an exam which includes n different questions $Q_1, \ldots, Q_n$. Each question Q_i ($i = 1 \,..\, n$) has a positive value in points, denoted by p_i, and also the time (in minutes) to solve the question, denoted by t_i. Overall, the exam concludes after T minutes. Given this information, the student wants to find a selection $S \subseteq \{1, \ldots, n\}$ of questions to solve that maximises the total amount of points $\sum_{i \in S} p_i$ subject to the time constraint $\sum_{i \in S} t_i \leq T$.

(a) [5 Pts] Argue why brute-force enumeration of all possible selections of questions leads to an exponential algorithm?

(b) [10 Pts] Describe a bottom-up dynamic programming procedure PointsQuestions(p, t, T) that takes as input $p[1 \,..\, n]$, $t[1 \,..\, n]$, and $T > 0$ and returns the the total amount of points of an optimal selection of questions. Analyse the worst-case running time of your algorithm.

(c) [5 Pts] Describe a procedure Questions(p, t, T) that prints an optimal selection of questions.

Solution 4.

(a) The following pseudocode describes a solution of the problem based on a brute-force enumeration of all possible selections of the questions.

```
QUESTIONS(p, t, T)
P = 0
S* = ∅
for each S ⊆ {1, ... n}
      if P < Σ_{i∈S} p[i] and Σ_{i∈S} t[i] ≤ T
            P = Σ_{i∈S} p[i]
            S* = S
return S, P
```

The number of iterations of the for-loop is 2^n (because that is the number of subsets of $\{1, \ldots n\}$), and the body of the for-loop takes $\Theta(n)$. Therefore the overall running time is $\Theta(n \cdot 2^n) = \Theta(2^n)$.

(b) Note that this problem is an instance of the *knapsack problem*. Here, p corresponds to the items values, t to the items weight, and T the maximal weight. We simply reuse that procedure Knapsack from Exercise Session 9 to find the total amount of point of an optimal selection of questions.

```
POINTSQUESTIONS(p, t, T)
KNAPSACK(p, t, T)
```

The worst-case running time of this procedure is $O(T \cdot n)$

(c) As before, we can reuse the procedure Print-Knapsack from Exercise Session 9 to print an optimal selection of questions.

```
QUESTIONS(p, t, T)
PRINT-KNAPSACK(p, t, T)
```

Question 5. 20 Pts

Let A denote a finite set of labels. We extend the notion of directed graphs to labelled directed graphs $G = (V, E)$ were V is a set of vertices and $E \subseteq V \times A \times V$ is a set of labelled edges. Each labelled edge $(v, a, v') \in E$ is associated to a weight $w(v, a, v')$ by means of a weight function $w \colon E \to \mathbb{R}$.
Let $\pi = \langle v_0, a_1, v_1, a_1, \dots, a_k v_k \rangle$ be a path of G, the *weight* of p is the sum of the weights of its constituent edges, namely $w(\pi) = \sum_{i=1}^{k} w(v_{i-1}, a_i, v_i)$; and the *trace* of p is the sequence of labels of p, namely $tr(\pi) = a_1 \mathinner{..} a_k$.
All-pairs same-trace shortest paths problem: Given a labelled directed graphs $G = (V, E)$ and a trace $\tau = a_1 \mathinner{..} a_k$, we want to find for every pair of vertices $u, v \in V$ a shortest (least-weighted) path $u \rightsquigarrow_\pi v$ from u to v such that $tr(\pi) = \tau$.

Hint: By assuming $V = \{1, \dots, n\}$ and $A = \{1, \dots, m\}$, we can conveniently represent a labelled directed graph $G = (V, E)$ with weight function w by means of a sequence $W^1 \mathinner{..} W^m$ of $n \times n$ adjacency matrices. For $a \in A$ the entry for the pair $(i, j) \in V \times V$ of $W^a = (w^a_{ij})$ is

$$w^a_{ij} = \begin{cases} w(i, a, j) & \text{if } (i, a, j) \in E \\ \infty & \text{if } (i, a, j) \notin E \end{cases}$$

(a) [10 Pts] Describe an algorithm All-Pairs-Same-Sequence(W, τ) that, given an array $W[1 \mathinner{..} m]$ of $n \times n$ adjacency matrices and an array $\tau[1 \mathinner{..} k]$ of labels, returns an $n \times n$ matrix $D = (d_{i,j})$ such that $d_{i,j} = \min\{w(\pi) \mid i \rightsquigarrow_\pi j, \text{ and } tr(\pi) = \tau\}$.

(b) [10 Pts] Analyse the asymptotic worst-case running time and space usage of your solution.

Solution 5.

(a) Taking as input the matrices $\{W^a\}_{a \in A}$ and the trace $\tau = a_1 \mathinner{..} a_k$ we compute a series of matrices $D^{(0)}, \dots, D^{(k)}$ where for $h = 1 \mathinner{..} k$, we have $D^{(h)} = (d^{(h)}_{i,j})$. The final matrix $D^{(k)}$ will contain the actual shortest-path weights. The heart of the algorithm is the following recursive definition for $d^h_{i,j}$.

$$d^{(0)}_{i,j} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{otherwise,} \end{cases}$$

and for $1 \leq h \leq k$

$$d^{(h)}_{i,j} = \min_{v=1..n} \{d^{(h-1)}_{i,v} + w^{a_h}_{v,j}\} \,. \tag{1}$$

Intuitively, $d^{(h)}_{i,j}$ represents the shortest-path weight of a path π from i to j with $tr(\pi) = a_1 \mathinner{..} a_h$.

Note that the Equation (1) is implemented by the Extend-Shortest-Paths algorithm seen in CLRS p.688. Specifically, for $1 \leq h \leq k$, $D^{(h)} =$ Extend-Shortest-Paths$(D^{(h-1)}, W^{a_h})$.

The pseudocode for All-Pairs-Same-Sequence(W, τ) is described below.

All-Pairs-Same-Sequence(W, τ)
$n = W[1].rows$
$k = \tau.length$
let $D^{(0)}$ be an $n \times n$ matrix initialised with ∞
for $i = 1$ **to** n
    $D^{(0)}[i, i] = 0$
for $h = 1$ **to** k
    $D^{(h)} =$ Extend-Shortest-Paths$(D^{(h-1)}, W[\tau[h]])$
return $D^{(k)}$

(b) The worst-case running time for ALL-PAIRS-SAME-SEQUENCE(W, τ) is $\Theta(k \cdot n^3)$ because line 7 takes $\Theta(n^3)$ (see CLRS p.688) and the for-loop in lines 6–7 iterates exactly k times. Note that the running time of the initialisation is $\Theta(n^2)$ which is overruled by the for-loop in lines 6–7.

Note that we do not need to store all D matrices, we can perform the same algorithm by keep updating the same distance matrix, using $\Theta(n^2)$. EXTEND-SHORTEST-PATHS uses $\Theta(n^2)$ space. Therefore, the overall space required by ALL-PAIRS-SAME-SEQUENCE is $\Theta(n^2)$.