Exam - June 2022

Algorithms and Data Structures (DAT2, SW2, DV2)

Instructions. This exam consists of **five questions**, each divided into sub-questions. You must hand-in your solutions in digital exam as a **single pdf file**. You are encouraged to mark the multiple choice answers as well as the labelling of graphs directly in this exam sheet.

- Before starting solving the questions, read carefully the exam guidelines at `https://www.moodle.aau.dk/mod/page/view.php?id=1340499`.
- Read carefully the text of each exercise. Pay particular attention to the terms in bold.
- **CLRS** refers to the textbook T.H. Cormen, Ch. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms* (3rd edition).
- You are allowed to refer to results in the textbook as well as exercise or self-study solutions posted in Moodle to support your arguments used in your answers.
- Make an effort to present your solutions neatly and precisely.

Question 1. 15 Pts

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

(1.1) [5 Pts] Mark **ALL** the correct answers. $n^4 \sqrt[4]{n} + \lg 2^{n^5} + n(n^3 + n \lg n)$ is

☐ **a)** $O(n^5 \lg n)$ ☐ **b)** $\Omega(n)$ ☐ **c)** $\Theta(n^{4.5})$ ☐ **d)** $\Theta(n^5 \lg n)$ ☐ **e)** $\Theta(n^5)$

(1.2) [5 Pts] Mark **ALL** the correct answers. $n \lg 2^{\lg n} + n + n \lg n^{2000} + 2000$ is

☐ **a)** $O(n)$ ☐ **b)** $\Omega(n)$ ☐ **c)** $\Theta(n \lg n)$ ☐ **d)** $\Omega(\sqrt{n})$ ☐ **e)** $O(n^{200})$

(1.3) [5 Pts] Mark **ALL** the correct answers. $100n + n \lg m + n^2$ is:
Hint: CLRS p.52 Exercise 3.1–8

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

☐ **d)** $\Omega(n \lg m + n^2)$ ☐ **e)** $\Theta(n^2 + \lg m)$

Solution 1.

(1.1)

$$n^4 \sqrt[4]{n} + \lg 2^{n^5} + n(n^3 + n \lg n) = n^{4.25} + n^5 + n^4 + n^2 \lg n = \Theta(n^5)$$

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

(1.2)

$$n \lg 2^{\lg n} + n + n \lg n^{2000} + 2000 = n \lg n + n + 2000 n \lg n + 2000 = \Theta(n \lg n)$$

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

(1.3)

$$100n + n \lg m + n^2 = \Theta(n \lg m + n^2)$$

Therefore **c** and **d** are correct.

Question 2. 20 Pts

Consider the following recurrences

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

Answer the questions below concerning these two recurrences. For each question, pay close attention to whether it concerns $Q(n)$ or $T(n)$.

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

- ☐ **a)** $Q(n)$ 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)** $Q(n)$ cannot be solved using the Master Theorem

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

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

(2.3) [10 Pts] Can we prove $T(n) = \Omega(2^n)$ using the substitution method? If yes, then provide such a proof using the substitution method. If no, then argue why not.

Solution 2.

(2.1) Note that the recurrence is of the form $Q(n) = aQ(n/b) + f(n)$ where $a = \frac{1}{4}$, $b = 2$, and $f(n) = 2n$. As stated in the book, the Master Theorem (CLRS Thm. 4.1) requires $a \geq 1$. Therefore, the correct answer is **d**. We will see that, using the substitution method, one can prove $Q(n) = \Theta(n)$.

In this specific case, it turns out that relaxing the requirement that $a \geq 1$, one can still apply the master method (case 3) and obtain the same asymptotic bound. Therefore we decided to give full score also to those who answered **c**.

Relaxing the requirement that $a \geq 1$, This recurrence falls into the third case, because $f(n) = 2n = \Omega(n^{-2+\epsilon}) = \Omega(n^{\log_b a + \epsilon})$ for some value of ϵ such that $2 < \epsilon \leq 3$. Moreover the "regularisation" condition $af(n/b) \leq cf(n)$ holds for $c = 1/8$ and $n \geq 0$. We prove the inequality below

$$\begin{aligned} af(n/b) &= \\ &= \frac{1}{4} \cdot 2 \cdot \frac{n}{2} && (a = \tfrac{1}{4},\ b = 2,\ f(n) = 2n) \\ &= \frac{1}{8} \cdot 2n \\ &= c \cdot f(n) && (c = 1/8,\ f(n) = 2n) \end{aligned}$$

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

(2.2) As mentioned before, $Q(n) = \Theta(n)$. Therefore, the correct answers are **a**, **b**, and **c**.

By Theorem 3.1 CLRS we can split the proof in two parts. In the first part we prove that $Q(n) = O(n)$, in the second part we prove that $Q(n) = \Omega(n)$. Let us rewrite the definition of Q by making explicit the constant $d > 0$ hidden behind $\Theta(1)$.

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

Part 1. We prove that for some $c > 0$, $Q(n) \le cn$ for all $n \ge 1$.

Basis ($n = 1$). By choosing $c \ge d$ we have $Q(1) = d \le c$.

Inductive Step ($n > 1$). By choosing $c \ge 32/7$ we have that

$$\begin{aligned}
Q(n) &= \frac{1}{4}Q(n/2) + 4n && \text{(def. } Q\text{)}\\
&\le \frac{1}{4}\left(c\frac{n}{2}\right) + 4n && \text{(inductive hypothesis)}\\
&= \left(\frac{1}{4}c + 4\right)n \\
&\le cn && (c \ge 32/7)
\end{aligned}$$

By choosing $c \ge \max(d, 32/7)$ the proof works.

Part 2. We prove that for some $c > 0$, $Q(n) \ge cn$ for all $n \ge 1$.

Basis ($n = 1$). By choosing $c \le d$ we have $Q(1) = d \ge c$.

Inductive Step ($n > 1$). By choosing $c \le 32/7$ we have that

$$\begin{aligned}
Q(n) &= \frac{1}{4}Q(n/2) + 4n && \text{(def. } Q\text{)}\\
&\ge \frac{1}{4}\left(c\frac{n}{2}\right) + 4n && \text{(inductive hypothesis)}\\
&= \left(\frac{1}{4}c + 4\right)n \\
&\ge cn && (c \le 32/7)
\end{aligned}$$

By choosing $0 < c \le \min(d, 32/7)$ the proof works.

(2.3) To prove that $T(n) = \Omega(2^n)$, it suffices to show that for all $n \ge 3$, $T(n) \ge c2^n$ for some suitable constant $c > 0$ (notice that this corresponds to chose $n_0 = 3$ in the definition of Ω-notation).

Basis We have to consider two cases

($n = 3$) Then we have that $T(3) = 3 \cdot T(1) = 3 \ge c2^3$. The inequality holds true for any choice of c such that $0 < c \le 3/8$.

($n = 4$) Then we have that $T(4) = 4 \cdot T(2) = 4 \cdot 2 \cdot T(0) = 8 \ge c2^4$. The inequality holds true for any choice of c such that $0 < c \le 1/2$.

Inductive Step ($n > 4$). We have that

$$\begin{aligned}
T(n) &= n \cdot T(n-2) && \text{(def. T)}\\
&\ge n \cdot c2^{n-2} && \text{(inductive hypothesis)}\\
&\ge 4 \cdot c2^{n-2} && (n > 4)\\
&= 2^2 \cdot c2^{n-2} \\
&= c2^n\,.
\end{aligned}$$

Overall, by choosing $0 < c \le 3/8$ we can make all cases to work. This proves that $T(n) \ge c2^n$ for all $n \ge 3$, which implies $T(n) = \Omega(2^n)$.

Question 3. 25 Pts

Understanding of known algorithms.

(3.1) [5 Pts] Mark **ALL** the correct statements.

- ☐ **a)** QUICKSORT and INSERTION-SORT have the same worst-case asymptotic running time.
- ☐ **b)** MAX-HEAPIFY and MERGE-SORT work in place.
- ☐ **c)** COUNTING-SORT can sort any array of integer numbers in linear time.
- ☐ **d)** Insertion in a binary search tree with n elements takes always $\Theta(\lg n)$ time.
- ☐ **e)** Insertion in a red-black binary search tree with n elements takes always $\Theta(\lg n)$ time.

(3.2) [4 Pts] Mark **ALL** the correct statements. Consider the array $A = [10, 5, 20, 1, 7]$.

- ☐ **a)** The binary tree interpretation of A satisfies the binary search tree property
- ☐ **b)** The result of BUILD-MAX-HEAP(A) is $[10, 5, 7, 1, 20]$
- ☐ **c)** Let $A.\mathit{heap\text{-}size} = A.\mathit{length}$. Then, after MAX-HEAPIFY($A, 2$), $A = [10, 7, 20, 1, 7]$
- ☐ **d)** A satisfies the max-heap property

(3.3) [6 Pts] Consider the hash table $T = 14, 71, 29, \text{NIL}, 32, 75, \text{NIL}$. For the following insertions we use *open addressing* with the auxiliary function $h'(k) = k$.
Remark: The "computed probe sequence" is the smallest sequence $\langle h(k, 0), h(k, 1), \ldots, h(k, i)\rangle$ such that $T[h(k, i)] = \text{NIL}$.

i) Insert the key $k = 7$ in T using *linear probing*. Mark **ALL** the correct statements.

- ☐ **a)** The resulting table is $14, 71, 29, 7, 32, 75, \text{NIL}$
- ☐ **b)** The resulting table is $14, 71, 29, \text{NIL}, 32, 75, 7$
- ☐ **c)** The computed probe sequence is $\langle h(k, 0), h(k, 1), h(k, 2), h(k, 3)\rangle$
- ☐ **d)** The computed probe sequence is $\langle h(k, 0), h(k, 1)\rangle$

ii) Insert the key $k = 14$ in T using *quadratic probing* with $c_1 = 2$ and $c_2 = 4$. Mark **ALL** the correct statements.

- ☐ **a)** The resulting table is $14, 71, 29, \text{NIL}, 32, 75, 14$
- ☐ **b)** The resulting table is $14, 71, 29, 14, 32, 75, \text{NIL}$
- ☐ **c)** The computed probe sequence is $\langle h(k, 0), h(k, 1), h(k, 2), h(k, 3)\rangle$
- ☐ **d)** The computed probe sequence is $\langle h(k, 0), h(k, 1)\rangle$

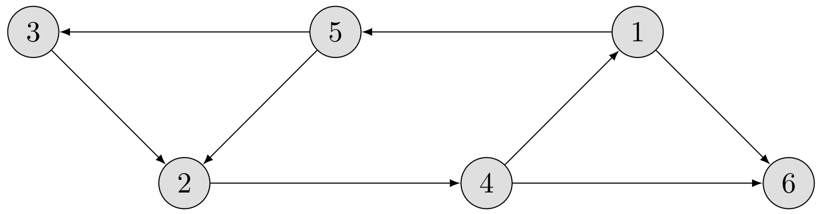
iii) Insert the key $k = 7$ in T using double hashing with

$$h_1(k) = k \qquad \text{and} \qquad h_2(k) = 1 + (k \mod (m-1))\,.$$

Mark **ALL** the correct statements.

- ☐ **a)** The resulting table is $14, 71, 29, 7, 32, 75, \text{NIL}$
- ☐ **b)** The resulting table is $14, 71, 29, \text{NIL}, 32, 75, 7$
- ☐ **c)** The computed probe sequence is $\langle h(k, 0), h(k, 1), h(k, 2), h(k, 3)\rangle$
- ☐ **d)** The computed probe sequence is $\langle h(k, 0), h(k, 1), h(k, 2), h(k, 3), h(k, 4)\rangle$

(3.4) [10 Pts] Consider the directed graph G depicted below.



(a) [2 Pts] 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 (see CLRS Sec 22.3). *Remark:* If more than one vertex can be chosen, choose the one with smallest vertex label.

(b) [2 Pts] 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.

(c) [4 Pts] Mark **ALL** the valid "parenthesis structures" of the discovery and finishing times in the sense of CLRS Theorem 22.7 resulting from some DFS visit performed on the above graph. *Remark:* In contrast with the above point, here vertices can be chosen arbitrarily.

- ☐ **a)** (5 (3 (2 (4 (6 6) (1 1) 4) 2) 3) 5)
- ☐ **b)** (5 (2 (4 (1 1) (6 6) 4) 2) (3 3) 5)
- ☐ **c)** (1 (5 (2 (4 (6 6) 4) 2) (3 3) 5) 1)
- ☐ **d)** (6 6) (3 (2 (4 (1 1) (5 5) 4) 2) 3)

(d) [2 Pts] If G admits a topological sorting, then show the result of TOPOLOGICAL-SORT(G) (see CLRS Sec 22.4). If it doesn't admit a topological sorting, briefly argue why.

Solution 3.

(3.1)

a) Correct. Both QUICKSORT and INSERTION-SORT have worst-case running time $\Theta(n^2)$ where n is the length of the input array.

b) Wrong. MAX-HEAPIFY works in place, but MERGE-SORT doesn't.

c) Wrong. Counting sort can sort arrays of non-negative integer numbers. E.g., the array $A = [-1, 3, 1, -2]$ in not a valid input for COUNTING-SORT as described in CLRS p. 195.

d) Wrong. If the binary search tree in unbalanced, insertion takes $\Theta(n)$.

e) Correct. Red-black binary search trees are balanced by construction. This allows insertion to be performed in $\Theta(\lg n)$ time.

(3.2)

a) Correct.

b) Wrong. After calling BUILD-MAX-HEAP(A), $A = [20, 7, 10, 1, 5]$.

c) Wrong. This is easily spot by the fact that 7 is repeated twice and 5 is missing. After calling MAX-HEAPIFY($A, 2$), $A = [10, 7, 20, 1, 5]$.

d) Wrong. This is easily spot by the fact that the first element of A is not the maximum element of the array.

(3.3) Recall that the hash functions under linear probing, quadratic probing, and double hashing are

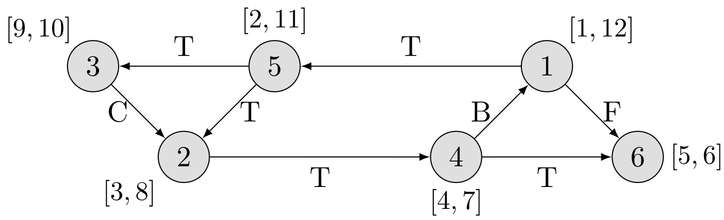
$$h(k,i) = (h'(k) + i) \mod m \qquad \text{(LINEAR PROBING)}$$
$$h(k,i) = (h'(k) + c_1 i + c_2 i^2) \mod m \qquad \text{(QUADRATIC PROBING)}$$
$$h(k,i) = (h_1(k) + i h_2(k)) \mod m \qquad \text{(DOUBLE HASHING)}$$

i) The correct answers are **a** and **c**. Indeed, $T[h(k,0)] = 14$, $T[h(k,1)] = 71$, $T[h(k,2)] = 29$, and $T[h(k,3)] = $ NIL.

ii) The correct answers are **a** and **d**. Indeed, $T[h(k, 0)] = 14$, and $T[h(k, 1)] = \textsc{Nil}$.

iii) The correct answers are **b** and **c**. Indeed, $T[h(k, 0)] = 14$, $T[h(k, 1)] = 29$, $T[h(k, 2)] = 32$, $T[h(k, 3)] = \textsc{Nil}$.

(3.4) The correct answers for (a) and (b) 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.



(c) The correct answers are **a** and **c**

(d) The graph contains a cycle, namely $1 \to 5 \to 4 \to 1$. Therefore it does not admit topological sorting. The presence of the cycle could also be spotted by the fact that the edge $(4, 1)$ was classified as a back-edge in (b).

Question 4. 20 Pts

Asymptotic runtime analysis.

Prof. Algo has been asked to analyse expected travelling times for a taxi company in Aalborg. To this end, he modelled the street map of Denmark as a weighted graph $G = (V, E)$ where each edge $e \in E$ represents a street segment whose weight $w(e)$ is the expected time required to drive through it.

(a) [10 Pts] Given a source location $s \in V$ and a target location $t \in V$, prof. Algo wants to determine which is the segment that takes the most time to drive through in a shortest path from s to t. For convenience he calls it the *bottleneck segment*. Then, he devises the procedure BSegment(G, w, s, t) that returns the expected time of the bottleneck segment from s to t.

```
BSEGMENT(G, w, s, t)
1  BELLMAN-FORD(G, w, s)
2  v = t
3  b = 0
4  while v.π ≠ NIL
5        b = max(b, w(v.π, v))
6        v = v.π
7  return b
```

Complete the following statements.
Remark: **ALL** the running times **must** be expressed a function of $|V|$ and $|E|$, where V and E are respectively the vertices and the edges of the input graph G.

a.1) The worst-case running time of line 1 in Θ-notation is: $\Theta(|V||E|)$

a.2) The worst-case running time of lines 2–3 in Θ-notation is: $\Theta(1)$

a.3) The worst-case running time of lines 4–7 in Θ-notation is: $\Theta(|V|)$

a.4) The worst-case running time of BSegment(G, w, s, t) in Θ-notation is: $\Theta(|V||E|)$

(b) [10 Pts] Given the location of a client $c \in V$ and a set $S \subseteq V$ of locations where available taxis are, Prof. Algo devises an algorithm to print the locations S ordered in non-decreasing expected arrival time to c.

```
RECOMMEND-TAXI(G, w, c, S)
1  let (G^T, w^T) be the transposed of (G, w)
2  DIJKSTRA(G^T, w^T, c)
3  Let T be an empty binary search tree
4  for each v ∈ S
5        if v.d < ∞
6              v.key = v.d
7              TREE-INSERT(T, v)
8  INORDER-TREE-WALK(T.root)
```

Assume that the input graph G is represented using adjacency lists, and that Dijkstra's algorithm uses the linear-array implementation of the min-priority queue.

Complete the following statements.
Remark: **ALL** the running times **must** be expressed as a function of $|V|$, $|E|$, and $|S|$, where V and E are respectively the vertices and the edges of the input graph G, and S is the input set of taxi locations.

b.1) The worst-case running time of line 1 in Θ-notation is: $\Theta(|V| + |E|)$

b.2) The worst-case running time of line 2 in O-notation is: $O(|V|^2 + |E|)$

b.3) The worst-case running time of line 3 in Θ-notation is: $\Theta(1)$

b.4) The worst-case running time of lines 4–7 in Θ-notation is: $\Theta(|S|^2)$

b.5) The worst-case running time of line 8 in Θ-notation is: $\Theta(|S|)$

b.6) The worst-case running time of RECOMMEND-TAXI(G, w, c, S) in O-notation is: $O(V|^2 + |S|^2)$

Solution 4.

(a) Line 1 takes $\Theta(|V||E|)$ (see CLRS p.651). Then, the execution of lines 2–3 takes $\Theta(1)$ time. A a single execution of the body of the while loop (lines 5–6) takes $\Theta(1)$ time. Thus the overall worst-case running time of lines 4–6 is determined by the maximum amount of iterations of the while loop. Note that the while loop is traversing (backward) a path from s to t in the shortest-path tree constructed by the call BELLMAN-FORD(G, w, s) made in line 1. The worst-case scenario occurs then the length of such path is maximal, that is when it has $|V| - 1$ edges. Therefore, in the worst-case, the execution of lines 2–7 takes $\Theta(|V|)$.

Summarising, we have that the worst-case running time of BSEGMENT(G, w, s, t) is

$$\Theta(|V||E|) + \Theta(|V|) = \Theta(|V||E|) .$$

(b) For the following analysis we assume that the input graph G is represented using adjacency lists, and that DIJKSTRA's algorithm uses the linear-array implementation of the min-priority queue.

Under the above assumptions, line 1 takes $\Theta(|V| + |E|)$ (see Exercise 1 from Exercise Session 10). Note that the transposed graph (G^T, w^T) has the same vertices as G and the same number of edges as G, therefore the execution of line 2 takes $O(|V|^2 + |E|)$ (see CLRS pp. 661–662). The construction of an empty BST performed in line 3 takes $\Theta(1)$ time. The for loop in lines 4–7 performs at most $|S|$ successive tree insertions, which in the worst-case will take $\sum_{i=1}^{|S|} \Theta(i) = \Theta(|S|^2)$ time. Finally, since after the execution of the for-loop the tree T will have at most $|S|$ elements, the execution of line 8 takes $\Theta(|S|)$.

Summarising, the worst-case running time of RECOMMEND-TAXI(G, w, c, S) is

$$\Theta(|V| + |E|) + O(|V|^2 + |E|) + \Theta(|S|^2) + \Theta(|S|) = O(V|^2 + |S|^2) .$$

Question 5. 20 Pts

Solving computational problems.

Alice is a dentist who runs a private ambulatory in Aalborg. She organises her working day in segments of 10 minutes each. For $i = 1 \mathinner{..} n$, Alice charges her clients p_i kr. for any task that takes i segments of her time, regardless from the actual type of task. Assuming that moving from one task to another takes no time, Alice wants to determine the **maximum** revenue obtainable in at most T working hours (here we assume that $6T \leq n$).

Example. We indicate a selection of k tasks ($1 \leq k \leq n$) taking respectively $i_1, \dots, i_k$ time segments, using the additive notation $i_1 + i_2 + \cdots + i_k$. The associated revenue for that selection is $r = p_{i_1} + p_{i_2} + \cdots p_{i_k}$. Consider the following price table

segments i	1	2	3	4	5	6	7	8	9	10
price p_i	100	500	800	900	1000	1550	1700	2000	2400	3000

The maximum revenue for 1 hour of work (corresponding to 6 segments of 10 min each) is 1600 kr. and is obtained via the task selection $3 + 3$, i.e., by performing 2 tasks of 30 minutes each.

(a) [10 Pts] Describe a **bottom-up** dynamic programming procedure MaxRevenue(p, T) that returns the maximum revenue obtainable in T working **hours**, according to the price table $p[1 \mathinner{..} n]$.

(b) [10 Pts] Describe a procedure PrintTasks(p, T) that **prints** a list of tasks that achieves the maximum revenue according to MaxRevenue(p, T).

Remarks:

- The description of the algorithmic procedures must be given **both** by providing the pseudocode and by explaining in detail how it works.
- Try to execute your algorithm on the above example. You may catch some errors you did not foresee while designing your algorithm.

Solution 5.

Note that the problem above described is an instance of the rod-cutting problem described in Lecture 9, where the length of the rod is $\lfloor 6T \rfloor$ and the price table is $p[1 \mathinner{..} n]$.

(a) The pseudocode for MaxRevenue(p, T) is

```
MaxRevenue(p, T)
    return Bottom-Up-Cut-Rod(p, ⌊6T⌋)
```

(b) The pseudocode for the procedure PrintTasks(p, T) is

```
PrintTasks(p, T)
    Print-Cut-Rod-Solution(p, ⌊6T⌋)
```