Algorithms and Data Structures (INF5/BAIT5/IT7)

Exam Assignments

Bin Yang

12 January 2017

Full name:	
Student number:	
E-mail at student.aau.dk:	

This exam consists of four exercises and there are **three** hours (13.30 p.m. to 16.30 p.m.) to solve them. When answering the questions in exercise 1, mark the check-boxes on this paper. Remember also to put your name and your student number on any additional sheets of paper you will use for exercises 2, 3, and 4.

During the exam you are allowed to consult books and notes. However, the use of any kind of electronic devices with communication functionalities, e.g., laptops, tablets, and mobile phones, is **NOT** permitted.

- *Read* carefully *the text of each exercise before solving it! Pay particular attentions to the terms in* **bold**.
- *For exercises 2, 3, and 4, it is important that your solutions are presented in a readable form. In particular, you should provide precise descriptions of your algorithms using pseudo-code or reference existing pseudo-code from the textbook CLRS. To get partial points for not completely correct answers, it is also worth to write two or three lines in English to describe informally what the algorithm is supposed to do.*
- *Make an effort to use a readable handwriting and to present your solutions neatly.*
- **CLRS** *refers to the textbook—T.H. Cormen, Ch. E. Leiserson, R. L. Rivest, C. Stein,* Introduction to Algorithms *(3rd edition).*

Exercise 1 [60 points in total]

Note that some of the following questions may have more than one correct solution. Mark the checkboxes for **ALL** correct solutions.

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

1.1. (*3 points*) $2^{\lg(n^2)} + n^{\lg 2}$ is:

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

1.2. (*3 points*) $0.3 \cdot \sqrt{64n^4} + 300 \cdot \lg(\lg n) + 1000 \cdot n$ is:

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

1.3. (*4 points*) $(\lg n) \cdot (5 \cdot n^2 - 80000 \cdot n + 200)$ is:

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

2. Consider the following recurrence:

$$\begin{aligned} T(1) &= 50000 \\ T(n) &= 4 \cdot T(n/4) + \sqrt{n} \qquad (n > 1), \end{aligned}$$

2.1. (*2 points*) Is this recurrence in the format that can be solved by the master method? If yes, choose which case should be used?

☐ **a)** No, the master method cannot solve the recurrence
☐ **b)** Yes, case 1 should be used
☐ **c)** Yes, case 2 should be used
☐ **d)** Yes, case 3 should be used

2.2. (*4 points*) Choose the correct solution for $T(n)$.

☐ **a)** $\Theta(n \lg n)$ ☐ **b)** $\Theta(n)$ ☐ **c)** $\Theta(\sqrt{n})$ ☐ **d)** $\Theta(\sqrt{n} \lg n)$

3. (*7 points*) Let's consider a scenario that Jakob uses a sorting algorithm to sort the following numbers (25, 10, 22, 17, 15). The algorithm produces the following sequences of numbers as it proceeds:
(10, 25, 15, 22, 17), (10, 15, 25, 17, 22), (10, 15, 17, 25, 22), (10, 15, 17, 22, 25).
The sorting algorithm that Jakob used is:

- [ ] **a)** Quick Sort
- [ ] **b)** Selection Sort
- [ ] **c)** Insertion Sort
- [ ] **d)** Bubble Sort

4. (*8 points*) Let A be an integer array of length 8 (indexed 1, 2, ..., 8), and define A.heap-size=8. After executing first Min-Heapify(A, 4), and then Min-Heapify(A, 2), which of the following statements is correct?

- [ ] **a)** It is possible to get A=(`17, 13, 15, 2, 9, 18, 19, 8`)
- [ ] **b)** It is possible to get A=(`5, 2, 7, 3, 1, 8, 9, 28`)
- [ ] **c)** It is possible to get A=(`15, 3, 7, 9, 8, 6, 5, 13`)
- [ ] **d)** It is possible to get A=(`13, 3, 9, 15, 17, 5, 4, 8`)
- [ ] **e)** We cannot execute Min-Heapify(A, 4) and/or Min-Heapify(A, 2) as the trees rooted at 4 and/or 2 may not be heaps.

5. Consider the MERGESORT(A, p, r) algorithm on p. 34, CLRS, which is also shown below.

MERGESORT(A, p, r)
if $p < r$
  q← $\lfloor \frac{(p+r)}{2} \rfloor$
  MERGESORT(A, p, q)
  MERGESORT$(A, q+1, r)$
  MERGE(A, p, q, r)

Given a sequence of numbers $A = (8, 7, 6, 5, 4, 3, 2, 1)$, and we call MERGESORT$(A, 1, 8)$.

5.1. (*4 points*) In the following sequences, which one is the sequence A after the MERGE() procedure has been executed **3 times**?

- [ ] **a)** `1, 2, 3, 4, 5, 6, 7, 8`
- [ ] **b)** `7, 8, 5, 6, 3, 4, 2, 1`
- [ ] **c)** `5, 6, 7, 8, 4, 3, 2, 1`
- [ ] **d)** `7, 5, 8, 6, 4, 3, 2, 1`

5.2. (*3 points*) After the MergeSort algorithm finishes, how many times in total has the MERGE() procedure been executed?

☐ **a)** 4 ☐ **b)** 5 ☐ **c)** 6 ☐ **d)** 7 ☐ **e)** 8

6. Consider the following algorithm DOSOMETHING(n).

```
DoSomething(Integer n)
  Initialize an empty stack S;
  for i ← 1 to n do
      S.push(i);
  Play(S, 1, n)

Play(Stack S, Integer x, Integer y)
  if x = y then
      for i ← 1 to x − 1 do
          S.pop();
  else
      for i ← 1 to x do
          S.push(i);
      Play(S, x+1, y)
```

Assume that passing a stack as an argument to a function takes constant time. Assume that $n \geq 1$.

6.1. (*4 points*) What is the run time of DOSOMETHING(n)?

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

6.2. (*4 points*) What is the size of the stack S after calling DOSOMETHING(n)?

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

7 (*7 points*) Consider a hash table with 7 slots and a quadratic hash function

$$h(k, i) = (2 \cdot k + 3i^2) \mod 7,$$

where k is the key and i is the probe number ($i = 0, 1, \ldots, 6$). After inserting 20, 12, 5, 3, 1 into an empty hash table. What does the hash table should look like?

☐ **a)** `1, empty, 3, 12, empty, 20, 5`

☐ **b)** `empty, 1, 3, 12, 5, 20, empty`

☐ **c)** `1, empty, 5, 12, empty, 20, 3`

☐ **d)** None of the above is correct. Write the correct one here: ____________

8 (*7 points*) Consider a directed graph $G = (V, E)$, where $V = \{a, b, c, d, e, f, g, h\}$. Its adjacency-matrix representation is given as follows.

	a	b	c	d	e	f	g	h
a	0	1	0	0	0	1	0	1
b	0	0	0	0	1	0	0	0
c	0	0	0	0	0	0	0	0
d	0	0	0	0	0	0	0	0
e	0	0	1	0	0	0	0	0
f	0	0	1	0	0	0	0	0
g	0	0	0	1	0	1	0	0
h	0	0	1	0	0	0	0	0

Let's run a topological sorting algorithm on the graph G. Which of the following choices are possible sequences of topologically sorted vertices?

☐ **a)** `g d a h f b e c`
☐ **b)** `a b e c f h g d`
☐ **c)** `d a b e f h c g`
☐ **d)** `a b e g f d h c`
☐ **e)** None of the above is correct as the graph is not a DAG.

Exercise 2 [5 points]

Consider an empty binary search tree. We do the the following sequence of insertions and deletions:
insert(5), insert(2), insert(3), insert(8), insert(10), delete(5), insert(9), delete(8)
How does the binary search tree look like? Please draw the binary search tree.

Exercise 3 [15 points]

Given an array A that consists of n integers, we want to identify the smallest integer in the array.

1. (*5 points*) Describe a **divide-and-conquer** algorithm to solve the problem. Write the pseudo code. Note that ***ONLY*** **divide-and-conquer** algorithms will be accepted.

2. (*5 points*) Write down the recurrence of your divide-and-conquer algorithm. Explain why you have the recurrence. In particular, how many sub-problems do you get at each step, what is the size of each sub-problem, and what is the cost of combining solutions from sub-problems?

3. (*3 points*) Solve the recurrence and identify the run-time complexity of your divide-and-conquer algorithm.

4. (*2 points*) Identify the asymptotic space overhead of your divide-and-conquer algorithm.

Exercise 4 [20 points]

We consider how graphs are helpful for solving some problems related to transportation.

First, let's consider all airports in the world. Suppose that we are given the information of all flights from all airline companies. For each flight, we know the source airport and destination airport. Note that when having a flight from airport A to airport B, it does not mean that there must exist another flight from airport B to airport A.

When a passenger uses a flight from airport A to airport B and then takes another flight from airport B to airport C, we say that the passenger transfers one time. Further, if the passenger continues to take a third flight from airport C to airport D, we say that the passenger transfers twice in total.

For any given airport S, Anders is willing to find all airports that are reachable from S using at most k times of transfers.

1. (*5 points*) Describe how to model the flight information and airports as a graph.

2. (*5 points*) Describe an algorithm which can help Anders to answer his question using the graph which you just modeled. Write down pseudo code. Analyze the asymptotic worst-case running time of your algorithm.

Next, let's consider another application scenario that concerns road transportation. Assume that we have a road network in a hilly region. For each direction of a road, we have two attributes: the length of the road and whether the road is downhill.

When an electrical vehicle travels on a flat or uphill road, the vehicle consumes electricity that is 0.5 times of the length of the road. In contrast, when an electrical vehicle travels on a downhill road, the vehicle recharges electricity that is 0.2 times of the length of the road.

For example, assume that we have a road that connects road intersections A and B. The length between A and B is 100, and from A to B is downhill and thus from B to A is uphill. When travelling from A to B, an electrical vehicle recharges 20 units of electricity, and when travelling from B to A, an electrical vehicle consumes 50 units of electricity.

Suppose Lene has an electrical vehicle that has m units of electricity in the beginning and she starts from a road intersection. She would like to know all road intersections that her vehicle can reach from the starting road intersection.

3. (*5 points*) Describe how to model the road network as a graph.

4. (*5 points*) Describe an algorithm that helps Lene answer her question. Write down pseudo code. Analyze the asymptotic worst-case running time of your algorithm.