# Algorithms and Data Structures (DAT3/SW3)

*Exam Assignments*

Bin Yang

15 January 2018

| | |
|---|---|
| Full name: | |
| Student number: | |
| E-mail<br>at student.aau.dk: | |

This exam consists of four exercises and there are **three** hours (10.00 a.m. to 13.00 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. However, you can bring old fashion calculators.

- *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.** (*4 points*) $\lg n^2 + \lg 2^n + 1^n + \sqrt{n}$ is:

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

**1.2.** (*4 points*) $558 \cdot n \lg n + 0.0001 \cdot n^3 + (n \lg n)^2$ is:

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

**2.** (*7 points*) Consider the following functions.

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

Please choose the function(s) which make the statement $\lg(f(n)) = \Theta(\lg n)$ be true.

**3** (*8 points*) Consider the following recurrence:

$$\begin{aligned} T(1) &= \Theta(1) \\ T(n) &= 9 \cdot T(n/3) + n^{1.9} \qquad (n > 1), \end{aligned}$$

Which of the following is correct? T(n) is:

☐ **a)** $\Theta(n^{1.9} \lg n)$ ☐ **b)** $\Theta(n^{1.9})$ ☒ **c)** $\Theta(n^2)$ ☐ **d)** $\Theta(n^3)$

**4.** After learning HeapSort from the lecture 6, Mike starts trying HeapSort on the following array:

$$11, 19, 2, 7, 5, 3, 8, 9.$$

Assume that Mike would like to sort the above array in an ascending order, i.e., from small numbers to big numbers.

**4.1** (*4 points*) Mike first builds a max-heap based on the array. What should the built max-heap look like?

- ☐ **a)** 19, 11, 8, 9, 2, 3, 5, 7
- ☒ **b)** 19, 11, 8, 9, 5, 3, 2, 7
- ☐ **c)** 11, 9, 8, 19, 7, 3, 2, 5
- ☐ **d)** 19, 8, 9, 7, 5, 11, 3, 2

**4.2** (*4 points*) Then, after the first two biggest numbers are placed into the correct positions and the Max-Heap-Property has been reestablished, what should the array look like now?

- ☐ **a)** 19, 11, 8, 9, 7, 3, 5, 2
- ☒ **b)** 9, 7, 8, 2, 5, 3, 11, 19
- ☐ **c)** 11, 8, 9, 7, 5, 2, 3, 19
- ☐ **d)** 9, 8, 3, 7, 5, 2, 11, 19

**5.** (*7 points*) Consider the following MERGESORT$(A, p, r)$ algorithm.

```
MERGESORT(A, p, r)
 1  if p < r
 2      q← ⌊(p+r)/2⌋
 3      MERGESORT(A, p, q)
 4      MERGESORT(A, q + 1, r)
 5      MERGE(A, p, q, r)
```

Given a sequence of numbers $A = (7, 3, 5, 9, 8, 2)$, and we call MERGESORT$(A, 1, 6)$. In the following sequences, which one is the sequence $A$ after the MERGE() procedure has been executed **3 times**?

- ☐ **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. Write the correct one here: _____

**6.** Consider the following algorithm StackOperations(n). Note that indentations indicate block structures.

```
StackOperations(Integer n)
 1  Initialize an empty stack S;
 2  for i ← 1; i < n − 1; i++ do
 3      for j ← n; j ≥ i; j−− do
 4          S.push(i);
 5  for k ← 1; k ≤ n^2; k++ do
 6      S.pop();
```

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

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

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

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

**7** (*7 points*) Consider a hash table with 9 slots with the following hashing function that uses double hashing to address conflicts.

$$h(k, i) = (k + i \cdot h'(k)) \mod 9$$

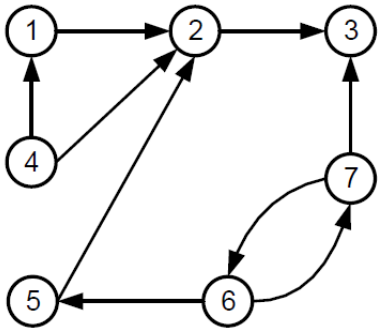
Here, $k$ is the key and $i$ is the probe number (where $i = 0, 1, \ldots, 8$) and we have

$$h'(k) = 2 + (k \mod 7).$$

After inserting 3, 10, 8, 2, and then 11 into an empty hash table, what does the hash table look like now?

☐ **a)** `11, 10, 2, 3, empty, empty, empty, empty, 8`
☐ **b)** `empty, 10, 2, 3, empty, empty, 11, empty, 8`
☒ **c)** `empty, 10, 2, 3, empty, 11, empty, empty, 8`
☐ **d)** None of the above is correct. Write the correct one here: _____

**8** (*7 points*) Consider the following graph. Which of the following sequence corresponds to the vertices in topologically sorted order?




☐ **a)** `4, 1, 6, 7, 5, 2, 3`
☐ **b)** `6, 7, 4, 5, 1, 2, 3`
☐ **c)** `6, 5, 4, 1, 2, 7, 3`
☒ **d)** None of the above is correct.

# Exercise 2 [10 points]

We have seen how to use a binary heap to implement a priority queue in Lecture 7. Now, let's try to use a doubly linked list $L$ to implement a priority queue.

1. (*3 points*) Briefly describe how to implement operation EXTRACT-MAX(*List* $L$). This operation removes the element with the largest key from the priority queue. What is the complexity of your implementation of operation EXTRACT-MAX(*List* $L$).

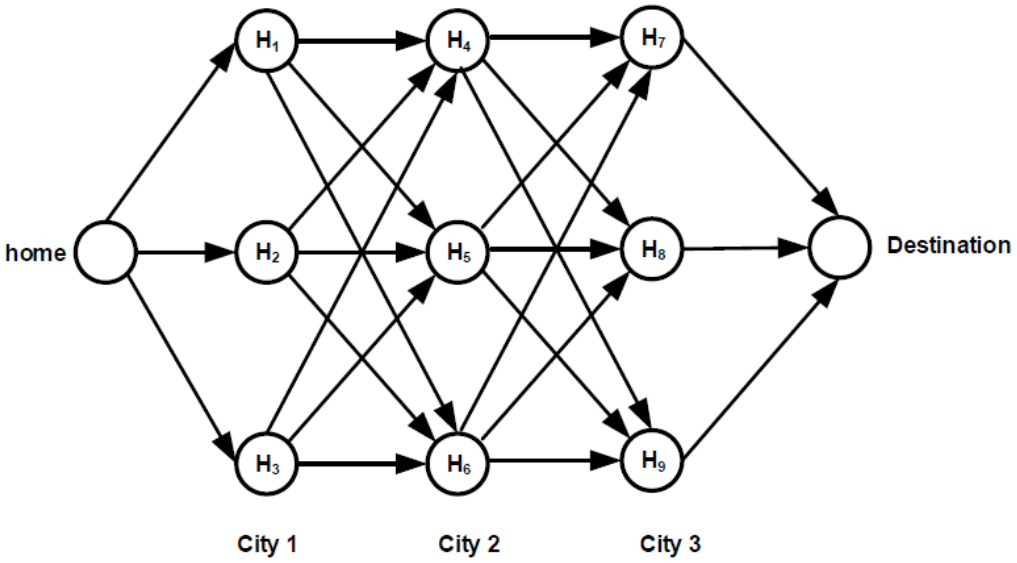
2. (*3 points*) Briefly describe how to implement operation INSERT(*List* $L$, *int* $x$). This operation inserts an element whose key is $x$ into the priority queue. What is the complexity of your implementation of operation INSERT(*List* $L$, *intx*).

3. (*4 points*) Recall that Dijkstra's algorithm employs a priority queue to identify shortest paths in a graph. Thus, the complexity of Dijkstra's algorithm is dependent on how priority queues are implemented. Assume that Dijkstra's algorithm uses the priority queue based on your above implementations, what is then the complexity of Dijkstra's algorithm? Briefly explain how you get the complexity.

# Exercise 3 [20 points]

Let us consider a trip planing problem. Kasper wants to travel by car from his home in city A to a hotel in city B as the destination. Since the distance from A to B is very large, e.g., from Aalborg to Beijing, Kasper has to stopover at a few cities on the way. Assume that there is a sequence of **n** cities that Kasper needs to stop at. In each city, there are **3** hotels in different locations. Each hotel has a unique price. Assume that the fuel cost is equal to the Euclidean distance. The total cost of Kasper's trip is the fuel cost plus the hotel prices.

The following figure shows an example where there are $n = 3$ cities on the way. If Kasper travels from home to $H_1$, $H_5$, and then $H_8$, to the destination, we have




- The fuel cost is the distance home to $H_1$, plus distance from $H_1$ to $H_5$, plus distance from $H_5$ to $H_8$, and plus distance from $H_8$ to the destination.
- The hotel cost is the price of $H_1$, plus price of $H_5$, and plus price of $H_8$.

Now, our target is to figure out a trip that has the smallest total cost.

1. (*5 points*) Consider a naive algorithm to solve the problem. The naive algorithm enumerates all possible trips, compute their total costs, and return the trip with the smallest cost. What is the asymptotic complexity of the naive algorithm? For the specific example shown in the above figure, how many trips the naive algorithm needs to enumerate?

2. (*15 points*) Design a more efficient algorithm (compared to the naive algorithm) to solve the problem. You may need to use dynamic programming. Briefly write down your ideas first and then write down pseudo code. What is the asymptotic complexity of your algorithm?

# Exercise 4 [10 points]

Consider a rural area with some villages. There are some roads connecting these villages. Some of the roads are paved with asphalt, but others are unpaved roads. Each road starts and ends at a village.

A set of villages is called a ***civilization island*** if it is possible to go from any village to any other village in the set using only asphalt roads. A civilization island may contain a single village if it is not possible to go from this village to any other village using only asphalt roads.

Given such a rural area, we would like to count how many civilization islands there are in the rural area.

1. (*3 points*) We would like to solve the problem as a graph problem. Show how to model this problem as a graph.

2. (*7 points*) Design a graph algorithm to solve the problem. Analyze the worst-case running time of your algorithm.

Ex. 2 Always insert an element at the head of the list. Θ(1). ExtratMax goes through the entire list Θ(n). Modifykey can be done by searching for old key, removing the element with the old key and inserting a new element with the new key, so also Θ(n). Dijkstra's alg: |V |TExtractM in + |E|Tmodif ykey = |V^2 + |E||V |

Ex. 3 Niave alg: Θ(3^n). Example, 3^3trips.
Dynamic programming: Θ(n). At each hotel, compute the least cost to achieve the hotel.

Ex. 4 Undirected graph. Villages as vertices, roads as edges. Each edge has a label/weight to indicate if it is paved or not.
Run DFS, but only using paved edges. the number of DFS trees in the DFS forest is the number of civilization islands. Θ(V + E).