Advanced Algorithms
(DAT6/SW6/DE8/MI8/IT8)

Exam Assignments

Bin Yang

10.00 - 13.00, 13 June 2017

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

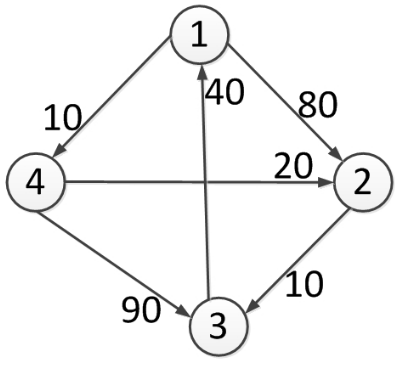
This exam consists of two exercises. Exercise 1 is a set of quizzes. Exercise 2 has a few open questions. When answering the quizzes in Exercise 1, mark the check-boxes or write down numbers, matrices, or sentences on these papers. When answering the questions in Exercise 2, remember to put your name and your student number on any additional sheets of paper you will need to use.

During the exam you are allowed to consult books, notes, and other written martials. 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 Exercise 2, it is important that your solutions are presented in a readable form. Make an effort to use a readable handwriting and to present your solutions neatly.*

Exercise 1 [50 points in total]

1. (*8 points*) We run Floyd-Warshall algorithm on the following graph.



Please write down the **distance matrix** after the second iteration, i.e., $D^{(2)}$.

$$\begin{pmatrix} & & & \\ & & & \\ & & & \\ & & & \end{pmatrix}$$

2. Consider the following three different dynamic table expansion strategies. Assume that the dynamic table is initialized with 10 elements in the beginning.

1. When the table is full, we make a new table whose size is 2,000 more elements bigger than the size of the old table.

2. When the table is full, assume that the table has N elements. We make a new table whose size is $\lceil \frac{N}{3} \rceil$ more elements larger than the size of the old table. For example, if the old table has 10 elements, the new table will have $\lceil \frac{10}{3} \rceil = 4$ more elements, i.e., 14 elements in total. If the old table has 9 elements, the new table will also have $\lceil \frac{9}{3} \rceil = 3$ more elements, i.e., 12 elements in total.

3. When the table is full, we make a new table whose size is 7 times of the size of the old table.

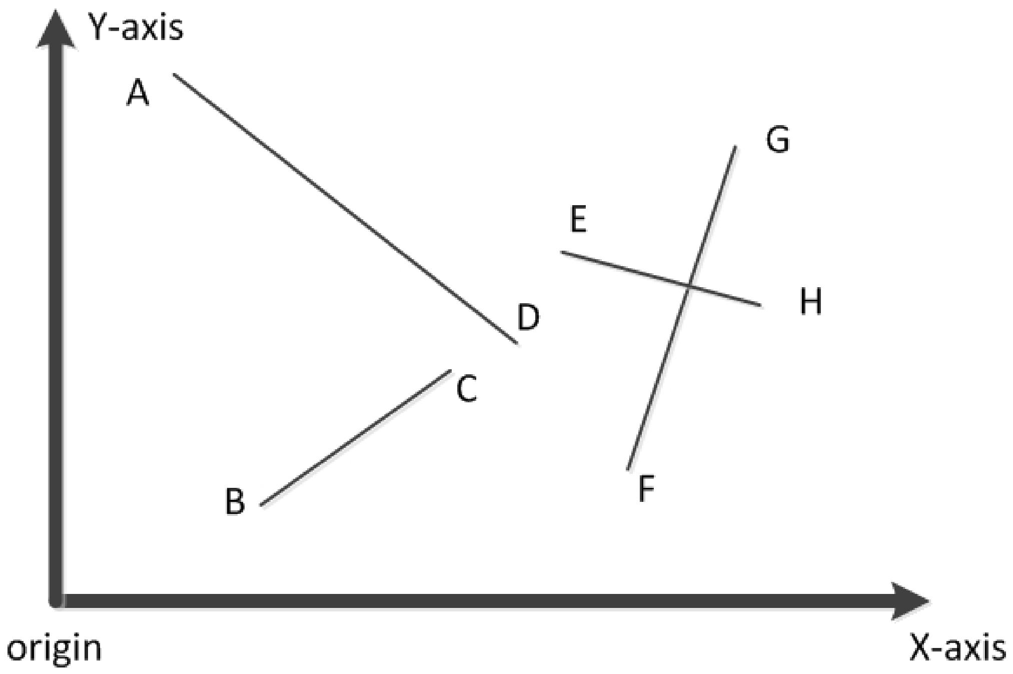
Assume that we only insert elements into the dynamic table. And consider a series of n insertions into the dynamic table. Please write down the amortized complexity of an insertion operation when using the three different strategies, respectively. Please use asymptotic notation.

2.1 (*3 points*) Strategy 1: ____________

2.2 (*3 points*) Strategy 2: ____________

2.3 (*3 points*) Strategy 3: ____________

3. Consider the following 4 line segments shown in the following figure.



Let's use the sweeping technique to check if there are line segments that intersect. We consider two different sweeping techniques using two different sweeping lines.

3.1 (*4 points*) We first consider an imaginary **vertical** sweep line that passes from the origin and keeps moving to the **right** according to the **x-axis**. Which point is the event point where the sweeping algorithm identifies the intersection?

☐ **1)** A ☐ **2)** B ☐ **3)** C ☐ **4)** D
☐ **5)** E ☐ **6)** F ☐ **7)** G ☐ **8)** H

3.2 (*4 points*) Next, we consider an imaginary **horizontal** sweep line that passes from the origin and keeps moving to the **top** according to the **y-axis**. Which point is the event point where the sweeping algorithm identifies the intersection?

☐ **1)** A ☐ **2)** B ☐ **3)** C ☐ **4)** D
☐ **5)** E ☐ **6)** F ☐ **7)** G ☐ **8)** H

4. Let's consider two dimensional (2D) range searches. Suppose that we have a set of n 2D points that are uniformly spread in a 10 km $\times$ 10 km space.

We are given three different workloads, where each workload consists of 100 2D range searches.

- **Workload 1**: each range search has a range that covers $\frac{1}{4}$ of the whole space.
- **Workload 2**: each range search has a range that covers $\frac{1}{100}$ of the whole space.
- **Workload 3**: each range search has a range that covers exactly 10 points in the space.

Now, we consider three different methods to process the three different workloads of 2D range searches: using a 2D range tree, using a kd-tree, and using two 1D Binary search trees (BSTs). We assume all the trees have been created so that you do not need to take into account the run time for constructing the trees.

4.1 (*3 points*) When processing the range searches in **Workload 1**, which method is the asymptotically fastest?

☐ **1)** Using a 2D range tree.
☐ **2)** Using a kd-tree.
☐ **3)** Using two 1D BSTs.
☐ **4)** They have the same asymptotic time complexity.

4.2 (*3 points*) When processing the range searches in **Workload 2**, which method is the asymptotically fastest?

☐ **1)** Using a 2D range tree.
☐ **2)** Using a kd-tree.
☐ **3)** Using two 1D BSTs.
☐ **4)** They have the same asymptotic time complexity.

4.3 (*3 points*) When processing the range searches in **Workload 3**, which method is the asymptotically fastest?

☐ **1)** Using a 2D range tree.
☐ **2)** Using a kd-tree.
☐ **3)** Using two 1D BSTs.
☐ **4)** They have the same asymptotic time complexity.

5. Assume that we have a computer with a large hard disk and a disk page of the hard disk is 1 GB (gigabyte). On the hard disk, there is a 1 TB (terabyte) file to be sorted.

Further, we define an I/O operation as either a read or a write of a disk page. Write down how many I/O operations are performed when using different external memory merge-sort algorithms.

5.1 (*3 points*) When using a **two-way merge-sort** algorithm, and assuming that the computer has **50 GB** (gigabyte) main memory, this requires
____________ I/O operations.

5.2 (*3 points*) When using a **multiway merge-sort** algorithm, and assuming that the computer has **50 GB** (gigabyte) main memory, this requires
____________ I/O operations.

5.2 (*3 points*) When using a **multiway merge-sort** algorithm, and assuming that the computer has **500 GB** (gigabyte) main memory, this requires
____________ I/O operations.

6. (*7 points*) Consider an approximation algorithm for solving a NP-complete problem P. The approximation ratio of the approximation algorithm is **1.5**. Which of the following statements is/are correct?

☐ **1)** If P is a maximization problem, and its optimal solution is 100, then, the approximation algorithm may return a value 110.

☐ **2)** If P is a maximization problem, and its optimal solution is 100, then, the approximation algorithm may return a value 20.

☐ **3)** If P is a minimization problem, and its optimal solution is 100, then, the approximation algorithm may return a value 110.

☐ **4)** If P is a minimization problem, and its optimal solution is 100, then, the approximation algorithm may return a value 20.

Exercise 2 [50 points in total]

1 We have seen the *Activity Selection* problem in the lecture a few times when we talked about dynamic programming and greedy algorithms. Let's now consider a slightly different problem called *Weighted Activity Selection.*

We are given a set of activities, where

- each activity a_i has a start time s_i and a finish time f_i, indicating that the activity lasts during the time interval $[s_i, f_i)$;
- each activity is also associated with a weight w_i.

We define that two activities are compatible if their intervals do not intersect. Now, our goal is to identify maximum weight subset of mutually compatible activities.

For example, given the activities shown in Table 1. The *Weighted Activity Selection* problem chooses subset $\{a_2, a_4\}$ as they are mutually compatible and the sum of weights of the subset is the largest. In contrast, *Activity Selection* chooses subset $\{a_1, a_3, a_5\}$ as they are mutually compatible and the subset has the largest cardinality.

Activity	a_1	a_2	a_3	a_4	a_5
Start time	1	5	8	12	15
Finish time	6	9	13	17	20
Weight time	10	25	20	30	10

Table 1: A Set of Activities with their start times, finish times, and weights

- **1.1** (*10 points*) Formalize the *Weighted Activity Selection* problem. Write down the recurrence.
- **1.2** (*10 points*) Design an algorithm using, e.g., Dynamic Programming or Greedy Algorithms, to solve the formalized problem. If you use greedy algorithms, please write explicitly what is your greedy strategy and argue why it is correct.

- **1.3** (*5 points*) Identify the time complexity of your algorithm.

2 A few families go out for dinner together. A family may have different numbers of members and a table may have different numbers of seats. When choosing seats, all families have agreed to follow the rule:

"Members from the same family should not sit at the same table."

Now, we want to solve the following two problems.

- Problem **P1**: Identify whether it is possible to find a seat arrangement such that the rule is satisfied.
- Problem **P2**: If it is possible to find a seat arrangement that satisfies the rule, describe such a seat arrangement, e.g., who should sit at which table.

To be more precise, assume that we have X families f_1, f_2, $\ldots$, f_X and the i-th family f_i has m_i members, where $1 \leq i \leq X$. Further, assume that we have Y tables t_1, t_2, $\ldots$, t_Y and the j-th table t_j has s_j seats, where $1 \leq j \leq Y$.

2.1 (*10 points*) Show how this problem can be formalized into a *maximum flow problem*. Write down what do vertices represent, which vertices should be connected by edges, and what is the capacity for each edge.
2.2 (*5 points*) Design an algorithm to solve the two problems **P1** and **P2**.

Consider a concrete example where we have 3 families f_1, f_2, f_3. Family f_1 has 2 members, family f_2 has 3 members, and family f_3 has 4 members; and a restaurant has 4 tables, where table t_1 has only 2 seats, where the remaining 3 tables all have 4 seats.

2.3 (*5 points*) Draw the flow network for this concrete example.
2.4 (*5 points*) Run the algorithm that you have designed on this concrete example and show the results to the two problems **P1** and **P2**.