Advanced Algorithms

(DAT6/SW6/IT8)

Exam Assignments

Bin Yang

10.00 - 13.00, 6 June 2018

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

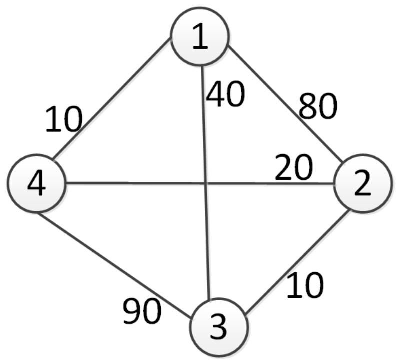
This exam consists of two exercises. Exercise 1 is a set of quizzes. Exercise 2 has a few open questions. When answering the quizzes in Exercise 1, mark the check-boxes or write down numbers, matrices, or sentences on these papers. When answering the questions in Exercise 2, remember to put your name 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. Old-fashion calculators are 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. We run Floyd-Warshall algorithm on the following undirected graph.



After the **1st iteration**:

(*5 points*) Please write down the **3rd row** of the **distance matrix**:

$$\langle \quad , \quad , \quad , \quad \rangle$$

(*5 points*) Please write down the **3rd row** of the **predecessor matrix**:

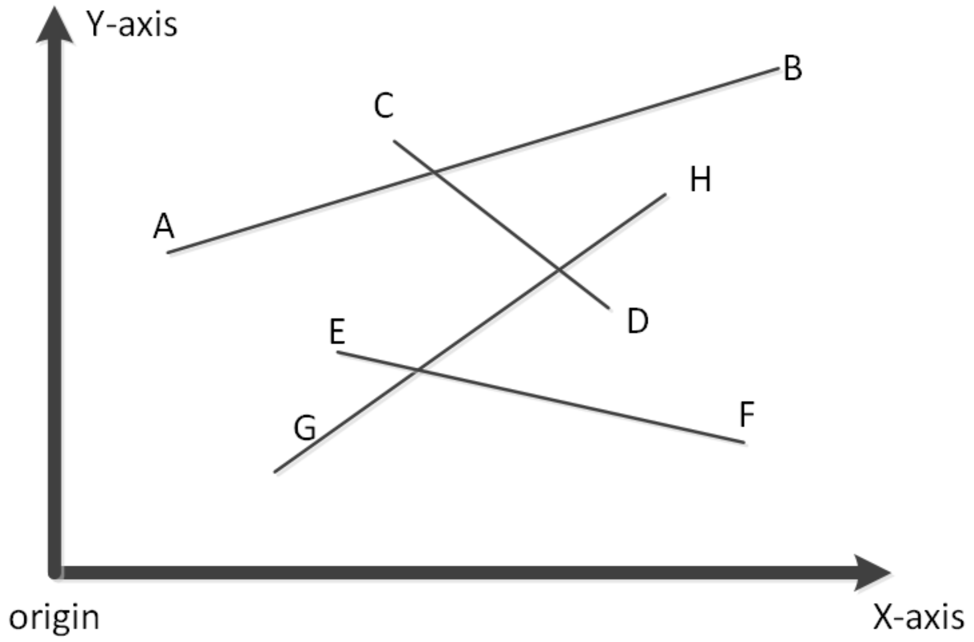
$$\langle \quad , \quad , \quad , \quad \rangle$$

2. (*10 points*) In Lecture 5, we have seen the **multipop** operation for stacks. Let's now consider a new operation for stacks—**multipush**. Specifically, **multipush**(S, k, x) pushes element x into stack S k times.

Now, consider a sequence of n stack operations, including **push**, **pop**, **multipop**, and **multipush**. Try to use one of the three amortized analysis techniques to identify the **amortized cost** for a stack operation. The amortized cost of a stack operation is:

☐ **1)** $\Theta(1)$
☐ **2)** $\Theta(n)$
☐ **3)** $\Theta(k^2)$
☐ **4)** $\Theta(n^2)$
☐ **5)** None of the above is correct. Please write down the correct answer: ________.

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



Let's use the efficient sweeping technique from Lecture 6 (i.e., the algorithm with $\Theta(n \log n)$ runtime where n is the number of line segments) to check if there are line segments that intersect.

3.1 (*5 points*) Consider the following four possible usages of sweeping lines. Please mark all the cases where the first identified intersection is the intersection between segment AB and segment CD.

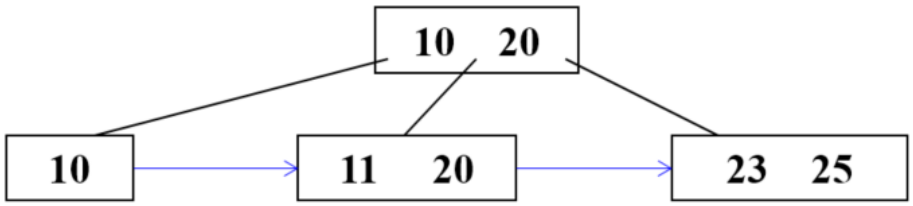
☐ **1)** Using a vertical sweeping line from left to right
☐ **2)** Using a vertical sweeping line from right to left
☐ **3)** Using a horizontal sweeping line from top to bottom
☐ **4)** Using a horizontal sweeping line from bottom to top

3.1 (*5 points*) Consider the following four possible usages of sweeping lines. Please mark all the cases where the first identified intersection is the intersection between segment EF and segment GH.

☐ **1)** Using a vertical sweeping line from left to right
☐ **2)** Using a vertical sweeping line from right to left
☐ **3)** Using a horizontal sweeping line from top to bottom
☐ **4)** Using a horizontal sweeping line from bottom to top

4. (*10 points*) Consider the following B$^+$-tree in an external memory setting, where nodes, except the root node, are stored on disk pages and the root node is always stored in **main memory**. Assume that a disk page can at most hold 2 keys.

Now, we start modifying the B$^+$-tree. We first delete 10, and then insert 28 into the B$^+$-tree. After the modifications, we ask a range query [24, 30] on the B$^+$-tree to retrieve the numbers that are within the range [24, 30]. How many disk pages do we need to access in order to process the query?



Please write down the number here:___________

5. Take a careful look at the following statements and decide if they are correct.

5.1 (*2 points*) Consider an approximation algorithm with approximation ratio 1.1 for solving a NP-complete problem P. Assume that P is a **maximization** problem and its optimal solution is 100. Then, the approximation algorithm may return a value 105.

☐ **1)** Correct ☐ **2)** Wrong

5.2 (*2 points*) Consider an approximation algorithm with approximation ratio 2 for solving a NP-complete problem P. Assume that P is a **minimization** problem and its optimal solution is 100. Then, the approximation algorithm may return a value 201.

☐ **1)** Correct ☐ **2)** Wrong

5.3 (*3 points*) Consider a multi-threaded algorithm A. If the **slackness** of A is 2 on a computer with **2 computation units**, then A's **parallelism** is 8.

☐ **1)** Correct ☐ **2)** Wrong

5.4 (*3 points*) Consider a multi-threaded algorithm A. If the **parallelism** of A is 10, then it is possible to achieve **perfect linear speedup** on a computer with **5 computation units**.

☐ **1)** Correct ☐ **2)** Wrong

Exercise 2 [50 points in total]

1 Suppose we have two sequences A and B, where each sequence has n positive integers. You can choose to reorder each sequence whatever you like. After reordering, let a_i be the i-th integer in the reordered sequence A, and let b_i be the i-th integer of the reordered sequence B. The profit of the reordering is defined as $\prod_i^n a_i^{b_i} = a_1^{b_1} \cdot a_2^{b_2} \cdot \dots \cdot a_n^{b_n}$.

For example, assume that we have two sequences $A = \langle 10, 20, 30 \rangle$ and $B = \langle 1, 2, 3 \rangle$. After the first reordering, we have $A = \langle 20, 10, 30 \rangle$ and $B = \langle 1, 2, 3 \rangle$. Then, we have profit $20^1 \cdot 10^2 \cdot 30^3 = 20 \cdot 100 \cdot 27000 = 5.4 \cdot 10^7$. After the second reordering, we have $A = \langle 10, 30, 20 \rangle$ and $B = \langle 2, 1, 3 \rangle$. Then, we have profit $10^2 \cdot 30^1 \cdot 20^3 = 10 \cdot 30 \cdot 8000 = 2.4 \cdot 10^6$. Comparing these two reorderings, the first reordering gives a higher profit.

- (*5 points*) Design an algorithm that maximizes the profit. Please first write down the basic ideas of your algorithm and then write down pseudo code of your algorithm.
- (*7 points*) Prove that your algorithm maximizes the profit.
- (*3 points*) Identify the asymptotic time complexity of your algorithm.

2 Consider an NP-complete problem—*Subset Sum*. Given a set S of n distinct positive integers and target integer T, identify whether there exists a subset of integers in S that add up to T. Notice that there can be more than one such subset. Consider the following two cases:

- Case 1: $S = \{2, 4, 6, 9\}$ and $T = 15$;
- Case 2: $S = \{11, 5, 1\}$ and $T = 15$.

For case 1, the answer is *True*, because the integers in subset $\{2, 4, 9\}$ add up to 15, and the integers in subset $\{6, 9\}$ also add up to 15. On the other hand, for case 2, the answer is *False*, because there does not exist a subset whose integers add up to 15.

- (*5 points*) Design a naive algorithm that does not use backtracking to solve the subset sum problem. Identify the asymptotic complexity.

- (*7 points*) Design an algorithm that uses backtracking to solve the subset sum problem. Define what is a configuration, i.e., what is the remaining subproblem to be solved and what is the set of choices made so far. What kind of configuration corresponds to a deadend and what kind of configuration corresponds to a solution.

- (*5 points*) Draw the search space for Case 2 and show the corresponding configuration for each node in the search space.

- (*3 points*) Identify the asymptotic complexity for the algorithm using backtracking.

3 Our department is going to host the 19th IEEE International Conference on Mobile Data Management (a.k.a., MDM) in Aalborg in the end of June 2018. Let's consider an very important problem when organizing a conference—*article-reviewer assignment*.

Researchers all over the world submit N articles to the MDM conference. The MDM conference organizers appoint an evaluation committee that consists of M expert reviewers. Each article must be reviewed by **3** expert reviewers. Each reviewer can at most review **5** articles. Further, each article has a particular topic, e.g., Algorithms, Databases, and Machine Learning. Each reviewer may have expertise on multiple topics. For example, reviewer R1 has expertise on both Algorithms and Databases; reviewer R2 has expertise on only Machine Learning. Then, articles on a specific topic only get reviewed by those reviewers who have the corresponding expertise. For example, an article with topic Machine Learning can only be reviewed by R2 but not R1. For each paper, based on the 3 evaluations provided by the expert reviewers, the MDM conference organizers decide whether the article should be accepted or be rejected.

Based on the above, the conference organizers would like to know the following:

- P1: Whether or not it is possible to assign articles to reviewers such that each article is with 3 reviewers whose expertise matches the article's topic;

- P2: If the answer to P1 is Yes, the conference organizers would also like to know which articles should be assigned to which reviewers.

- (*10 points*) Show how to a model the article-reviewer assignment problem as a *flow network*. In particular, write down what do vertices represent, e.g., articles or reviewers? Which vertices should be connected by edges, and what is the capacity for each edge?

- (*5 points*) Show how to answer P1 and P2 by solving the *maximum flow* problem on the flow network that you just constructed.