Advanced Algorithms (DAT6/SW6/DE8/MI8)

Exam Assignments

Bin Yang

10.00 - 13.00, 7 June 2016

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 this paper. 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 user.

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.*
- **CLRS** *refers to the textbook—T.H. Cormen, Ch. E. Leiserson, R. L. Rivest, C. Stein,* Introduction to Algorithms *(3rd edition).*

Exercise 1 [50 points in total]

1. Given a directed, weighted graph G, we run Floyd-Warshll algorithm on G. After the second iteration (k=2) of the Flord-Warshall algorithm, the **distance matrix** $D^{(2)}$ looks like the following.

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 5 & 2 & 6 \\ +\infty & 0 & 4 & -1 & 4 \\ +\infty & +\infty & 0 & +\infty & 2 \\ +\infty & +\infty & 4 & 0 & 12 \\ 5 & 8 & 10 & -5 & 0 \end{pmatrix}$$

1.1. (*4 points*) Which of the following four matrices could be the **adjacency matrix** of G?

☐ **a)** $\begin{pmatrix} 0 & 3 & 5 & 1 & 6 \\ +\infty & 0 & 4 & -1 & 4 \\ +\infty & +\infty & 0 & +\infty & 2 \\ +\infty & +\infty & 4 & 0 & 12 \\ 5 & +\infty & +\infty & -5 & 0 \end{pmatrix}$ ☐ **b)** $\begin{pmatrix} 0 & 3 & 5 & +\infty & \\ +\infty & 0 & 4 & -1 & \\ +\infty & +\infty & 0 & +\infty & \\ +\infty & +\infty & 4 & 0 & \\ 5 & +\infty & +\infty & -5 & \end{pmatrix}$

☐ **c)** $\begin{pmatrix} 0 & +\infty & 5 & +\infty & 6 \\ +\infty & 0 & 4 & -1 & 4 \\ +\infty & +\infty & 0 & +\infty & 2 \\ +\infty & +\infty & 4 & 0 & 12 \\ 5 & +\infty & 7 & -5 & 0 \end{pmatrix}$ ☐ **d)** $\begin{pmatrix} 0 & 3 & 5 & +\infty & 6 \\ +\infty & 0 & 4 & -1 & 4 \\ +\infty & +\infty & 0 & +\infty & 2 \\ +\infty & +\infty & 4 & 0 & 12 \\ 5 & +\infty & +\infty & -5 & 0 \end{pmatrix}$

1.2. (*4 points*) Based on the chosen graph G, write down the corresponding **predecessor matrix** after the second iteration (k=2).

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

2. Suppose that we have a document with the following content.

$$edefafbcfdcfdfefffefe$$

Although it is a short document, we still try to use Huffman coding to compress this document. Let's use the pseudo code of HUFFMAN(C) shown in CLRS, on page 431 to encode. After running the pseudo code, assume that the codeword for a is 11111.

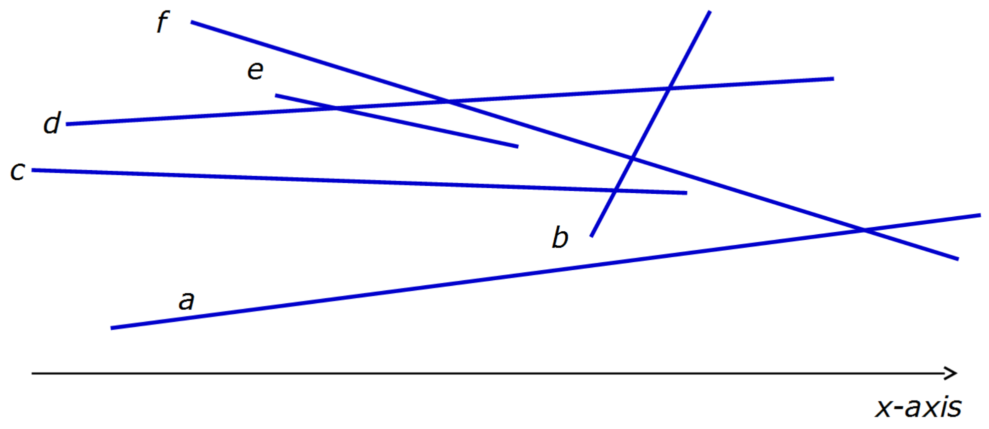
2.1. (*3 points*) What is the codeword for c?

☐ **a)** 1111 ☐ **b)** 1110 ☐ **c)** 110 ☐ **d)** 11111

2.2. (*3 points*) What is the codeword for d?

☐ **a)** 1111 ☐ **b)** 1110 ☐ **c)** 110 ☐ **d)** 11111

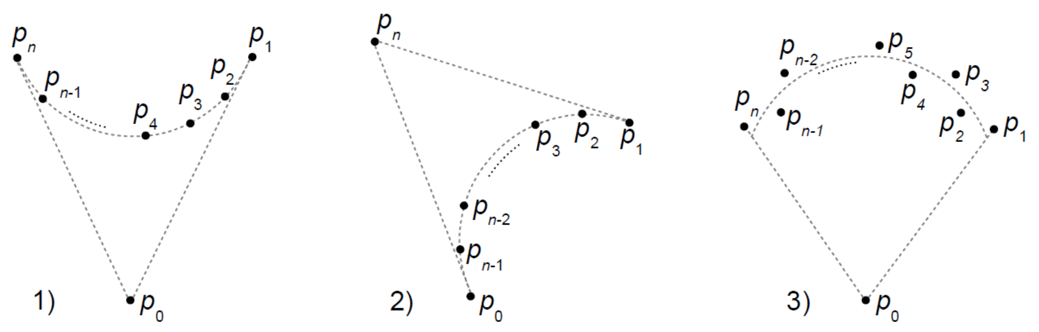
3. (*7 points*) Consider the following line segments a, b, c, d, e, and f, which are shown in the following figure.



Let's use the sweeping technique to check if there are line segments intersect, where an imaginary vertical sweep line passes from left to right according to the x-axis. In particular, we use the algorithm whose pseudo code ANY-SEGMENTS-INTERSECT(S) is given on page 1025, CLRS. Which intersection is the first intersection that the algorithm identifies?

☐ **a)** The intersection of a and f
☐ **b)** The intersection of b and c
☐ **c)** The intersection of b and f
☐ **d)** The intersection of b and d
☐ **e)** The intersection of d and e
☐ **f)** The intersection of d and f

4. Consider the following three sets of points (the dotted lines just illustrate the relative positions of points). We consider using either Graham's scan or the Jarvis's march to identify the convex hull of the points in each figure.



4.1. (*2 points*) Which of the following statements is correct for Figure 1)?

☐ **a)** Jarvis's march is asymptotically faster.
☐ **b)** Graham's scan is asymptotically faster.
☐ **c)** Both have the same asymptotic complexity.
☐ **d)** None of the above statements is correct.

4.2. (*2 points*) Which of the following statements is correct for Figure 2),

☐ **a)** Jarvis's march is asymptotically faster.
☐ **b)** Graham's scan is asymptotically faster.
☐ **c)** Both have the same asymptotic complexity.
☐ **d)** None of the above statements is correct.

4.3. (*2 points*) Which of the following statements is correct for Figure 3),

☐ **a)** Jarvis's march is asymptotically faster.

☐ **b)** Graham's scan is asymptotically faster.

☐ **c)** Both have the same asymptotic complexity.

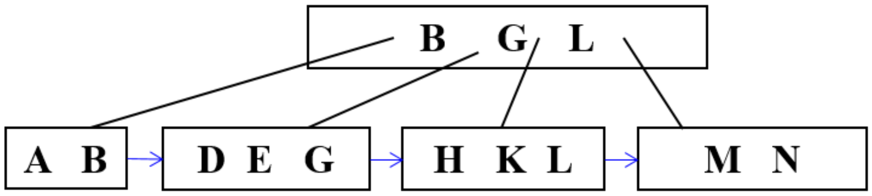
☐ **d)** None of the above statements is correct.

5. (*8 points*) Consider the following table that is maintained in a university's database. We build a kd-tree on the table. When building the kd-tree, we use the dimension on *Student Number* first, and then on the *Age* dimension, and finally on the *Grade* dimension. After building the kd-tree, a teacher asks a query to retrieve the information regrading to the students whose Student Numbers are between 140 and 250 and whose Grades are less than 50. How many leaf nodes in the kd-tree will be accessed to process the teacher's query?

Student Number	*Age*	*Grade*
101	24	80
112	16	70
123	20	60
150	24	75
210	15	55
243	26	60
248	27	70
290	23	80

Please write down the number here: ____________

6. (*8 points*) Consider the following B+-tree in an external memory setting, where nodes are stored on disk pages. Assume that a disk page can **at most hold 3 keys**. We insert two more data elements whose keys are X and Y into the B+-tree. After the insertions, we ask a range query [B, M] on the B+-tree to retrieve the elements whose keys are between B and M, and also including B and M. How many disk pages at least do we need to access? Note that we always keep the **root node** of the B+-tree in the **main memory**.



☐ **a)** 3 ☐ **b)** 4 ☐ **c)** 5 ☐ **d)** 6 ☐ **d)** 7

7. (*7 points*) Consider a multi-threaded algorithm whose **parallelism** is 20. Given the following setups, which one is the most likely setup to achieve a **perfect linear speedup**?

☐ **a)** A setup with slackness being 0.99.
☐ **b)** A setup with slackness being 0.01.
☐ **c)** A setup with slackness being 100.
☐ **d)** A setup with slackness being 10.
☐ **e)** A setup with 40 processors.
☐ **f)** A setup with 1000 processors.

Exercise 2 [50 points in total]

1 Consider the following two dynamic table expansion strategies.

- When the table is full, we make a new table whose size is three times of the size of the old table.
- When the table is full, we make a new table whose size is 10 more elements bigger than the size of the old table.

Assume that we only insert elements into the dynamic table.

- (*6 points*) What is the amortized cost of an insertion when the dynamic table is with the first strategy and second strategy, respectively?
- (*9 points*) Choose your favorite amortized analysis method to argue your decision is correct.

2 In Lecture 11, we have seen a 2-approximation algorithm (denoted as ALG1) for solving the **vertex cover** problem. We also briefly talked about a $(\ln |X| + 1)$-approximation algorithm (denoted as ALG2) for solving the **set cover** problem.

- (*10 points*) Actually, the **set cover** problem can be regarded as a generalization of the vertex cover problem. Show how can you transform a vertex cover problem into a set cover problem.
- (*10 points*) After the transformation, one can apply ALG2 to solve the vertex cover problem. Identify under which scenarios, transforming a vertex cover problem into a set cover problem and then applying ALG2 will give a better approximation ratio compared to applying ALG1 directly? Explain why your statement is correct.

3 Let's recall the spatial crowd-sourcing problem. We have a set of tasks $\{t_1, t_2, \dots, t_N\}$ and a set of workers $\{w_1, w_2, \dots, w_M\}$.

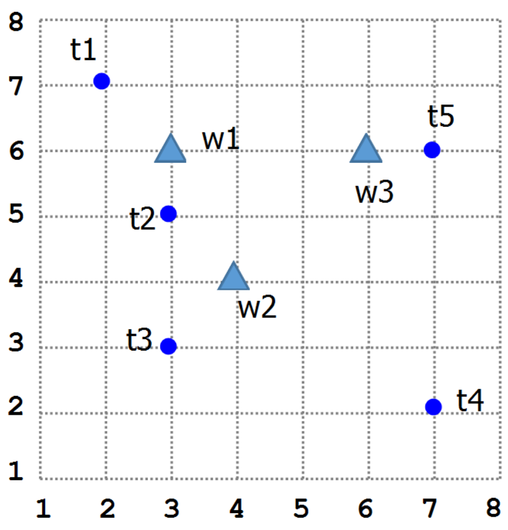
A task, t_i, where $1 \leqslant i \leqslant N$, is associated with a location, denoted as $t_i.location$, indicating where is the task.

A worker, w_j, where $1 \leqslant i \leqslant M$, is associated with the following attributes.

- A location, denoted as $w_j.location$, indicating where is the worker.
- A radius value, denoted $w_j.r$, indicating that worker w_j can only solve the tasks that are in the **active circle** with $w_j.location$ as the center and $w_j.r$ as the radius.
- A value indicating the maximum number of tasks the worker can solve, denoted as $w_j.maxTaskNum$.

A task t_i can be assigned to worker w_j if and only if $t_i.location$ is in worker w_j's active circle. The spatial crowd-sourcing problem is to assign as many tasks as possible to workers while satisfying that each worker w_j can have at most $w_j.maxTaskNum$ assigned tasks.

- (*7 points*) Show how to formulate a crowd-sourcing problem into a maximum-flow problem. In particular, describe how many vertices, how many edges, what does a vertex correspond to, and what is the capacity of an edge.
- (*8 points*) Consider the crowd-sourcing problem shown in the following figure. Draw the corresponding flow network. Use Edmonds-Karp algorithm to solve the problem and show the augmenting path that you choose in each step.



	Radius	maxTaskNum
w1	2	1
w2	1.8	3
w3	1.5	2