Algorithms and Data Structures (DAT3/SW3)

Re-exam E 2014

Manfred Jaeger

16 February 2015

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

This exam consists of three problems and there are three hours to solve them. When answering the questions in problem 1, mark or fill in the boxes on this paper. Remember also to put your name and your CPR number on any additional sheets of paper you will use for problems 2 and 3.

- *Read* carefully *the text of each problem before solving it!*
- *For problems 2 and 3, it is important that your solutions are presented in a readable form. If you don't have enough time to give full solutions for problems 2 and 3, then give a solution outline in a few lines of text.*
- *Make an effort to use a readable handwriting and to present your solutions neatly.*

[ItoA] refers to T.H. Cormen, Ch. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms*. Page references are for the 3rd edition.

During the exam you are allowed to consult books, printed lecture slides, and notes. The use of any kind of electronic devices, including calculators, is not permitted.

Problem 1 [50 points in total]

1. (*8 points*)

1.1. $3(\lg n)^2 + \sqrt{n}n^2 + (1+n^3)/n$ is:

☐ **a)** $\Theta((\lg n)^3)$ ☐ **b)** $\Theta(n^{2.5})$ ☐ **c)** $\Theta(n^3)$ ☐ **d)** $\Theta(n^2 \lg n)$

1.2. $(n^2)^3 + 6n^2n^3 + 2\lg(n^9)$ is:

☐ **a)** $\Theta(n^5)$ ☐ **b)** $\Theta(n^6)$ ☐ **c)** $\Theta(n^9)$ ☐ **d)** $\Theta(n^{4.5})$

2. (*6 points*)
Consider the following recurrence relation:

$$T(n) \quad = \quad 4T(n/2) + 2n^2 \qquad (n > 1).$$

Mark the correct solution. $T(n) =$

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

3. (*8 points*)
Consider the following algorithm:

input : An Array A with elements indexed $1..n$, integers l, r with $1 \leq l \leq n,\ 1 \leq r \leq n$
output: An Integer Z
$\textsc{Compute}(A, l, r)$

```
if l=r then
    return A[l]
else
    m = ⌊(l+r)/2⌋
    return Compute(A, l, m) + Compute(A, m+1, r)
```

3.1 For

A=

2	1	4	0	3	1	3
1	2	3	4	5	6	7

enter the return value of $\textsc{Compute}(A, 1, 7)$ in the box below:

$\textsc{Compute}(A, 1, 7) =$ ☐

3.2 Complete the following statement by entering the correct expression within the parentheses: the complexity of COMPUTE$(A, 1, n)$ as a function of n is

$$\Theta\Big(\quad \Big)$$

4 (*7 points*)

Consider the operation of QUICKSORT on an array A of length 8. Which of the following *can not* be the configuration of the array A, after line 2 of QUICKSORT$(A, 1, 8)$ has completed

☐ a)	4	2	5	3	4	7	11	8
☐ b)	4	2	4	3	5	8	7	11
☐ c)	4	2	5	3	4	8	7	11
☐ d)	4	2	5	3	4	8	11	7

5 (*7 points*)

We maintain a hashtable of size $m = 7$ using open addressing with auxiliary hash function

$$h'(k) = k \bmod 7$$

and linear probing. The current contents of the hashtable are:

0	1	2	3	4	5	6
12		9	16		5	13

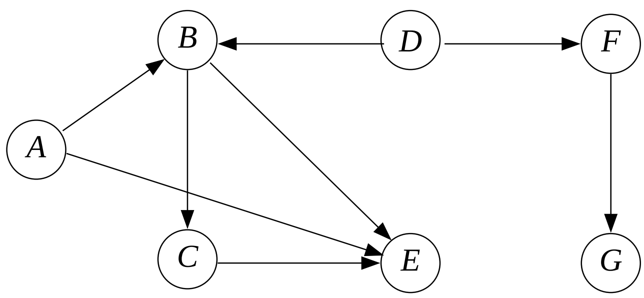
Write into the boxes below the contents of the hash table after the following sequence of operations has been performed:

$$\text{insert}(10),\ \text{delete}(16),\ \text{insert}(23)$$

0	1	2	3	4	5	6

6 (*6 points*)

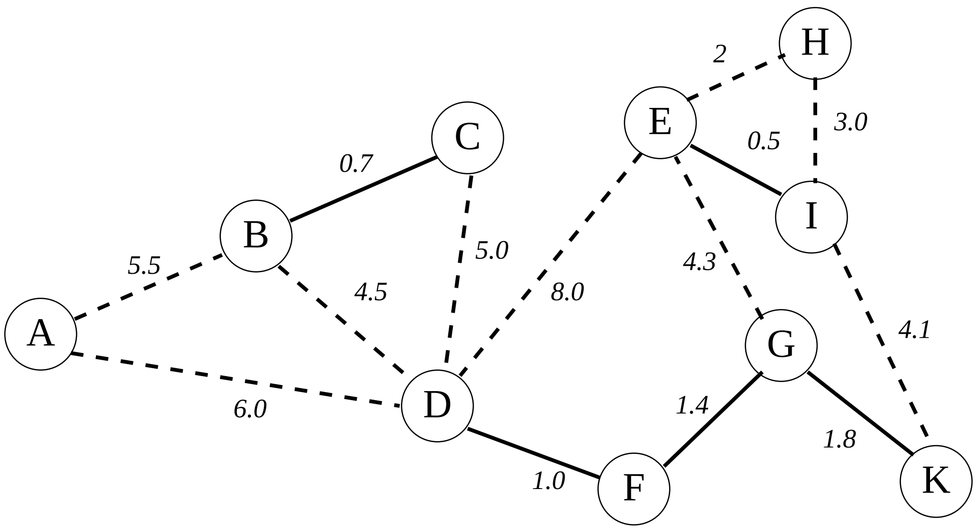
For the following directed graph:



write in the boxes below the labels of the nodes $A, ..., G$ in a topological sorted order:

7 (*8 points*)

The figure below shows an undirected weighted graph on which Kruskal's Minimum Spanning Tree algorithm ([ItoA], p. 631) is executed. The solid lines are edges that have already been added to the set A. The dotted lines are the remaining edges.



A Write in the boxes below the next three edges that will be added to A by the algorithm, in the order in which they are added:

B Write in the box below the number of iterations of the **for** loop of lines 5-8 that, starting from the configuration shown in the figure, are performed until the third additional edge is added to A (including the iteration in which the third edge is added):

Problem 2 [25 points]

For a set of n distinct integers $I = \{i_1, \ldots, i_n\}$ we define the *lower median* as the $\lceil \frac{n}{2} \rceil$th element when I is sorted in ascending order (if in doubt, see [ItoA] p. 54 for the $\lceil \ \rceil$ notation). For example, if $I = \{4, 2, 8, 5, 1\}$, then the lower median is 4: the 3rd element in the sorted order of I. For $I = \{4, 2, 6, 8, 5, 11, 1, 15\}$, the lower median is 5.

Following steps **A**-**C** below, describe a modification of the *Binary Search Tree (BST)* data structure that supports TREE-LOWER-MEDIAN(x) queries, where x is a node in a binary search tree. The method TREE-LOWER-MEDIAN(x) should return the lower median of the keys contained in the sub-tree rooted at x. You may assume that all keys contained in the tree are distinct (no duplicate keys).

In particular:

A Describe how the BST data structure should be modified.

B Describe briefly (in words, no pseudo-code) how the TREE-INSERT and TREE-DELETE methods must be modified.

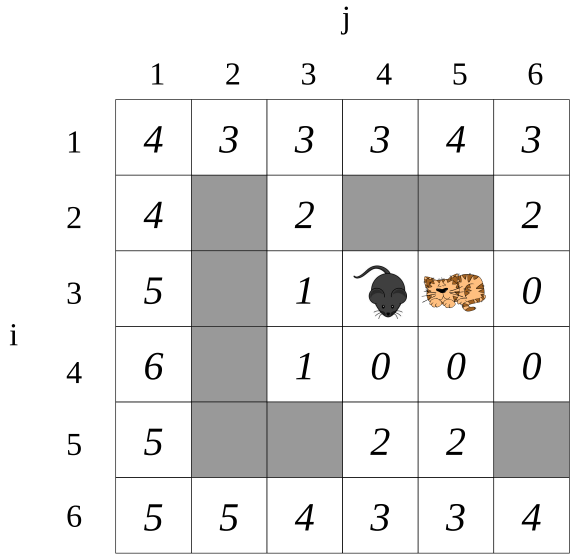
C Give a full pseudo-code description of the TREE-LOWER-MEDIAN(x) method.

Problem 3 [25 points]

Consider the following *Cat-and-Mouse* game: a cat and a mouse live in an environment consisting of an $n \times n$ square grid. Some of the squares in the grid are occupied by walls and therefore inaccessible. The mouse can move from any accessible square to any neighboring accessible square that is directly to the *left*, *right*, *above*, or *below* its current square. It takes the mouse 1 second to make any such move.

The mouse finds itself next to the cat, which, fortunately, is currently asleep. The cat will wake up in exactly T seconds. The mouse wants to move within these T seconds to a square that provides as much *protection* from the cat as possible. Each square has a *protection value*, which is a non-negative integer, and which depends on the square's distance from the cat, and on how shielded it is by walls.

The following is an example for a 6×6 grid. Inaccessible squares are shaded. Accessible squares are marked with their protection values. The squares currently occupied by the cat and the mouse have protection value 0.



A For the scenario shown in the figure, and $T = 5$, mark in the figure the square with maximal protection value that the mouse can reach, and the path it will take to that square.

B The general input for the *Cat-and-Mouse* problem is given by:

- An $n \times n$ matrix *ProtectMap* with integer entries representing the environment. If the square with coordinates (i, j) is inaccessible, then $ProtectMap[i, j] = -1$. If the square (i, j) is accessible, then $ProtectMap[i, j]$ is the non-negative protection value of that square.
- The coordinates (i_0, j_0) of the current position of the mouse (the coordinates of the cat need not be given explicitly – it is enough that the *ProtectMap* matrix has a 0 protection value at the cat's position).
- The available time T

The output of the problem consists of the coordinates (i, j) of the square with maximal protection value that is reachable for the mouse within at most T seconds, and the path leading from (i_0, j_0) to that square.

Give a specification in pseudo-code of an efficient algorithm FINDPROTECTION$(ProtectMap, i_0, j_0, T)$ for solving the *Cat-and-Mouse* problem. What is the complexity of your algorithm?