Algorithms and Data Structures (DAT3/SW3/INF5)

Re-exam Assignments

Simonas Šaltenis

26 March 2012

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

This exam consists of three exercises and there are three hours to solve them. When answering the questions in exercise 1, write directly in the provided fields on this paper. Remember also to put your name and your CPR number on any additional sheets of paper you will use.

- *Read* carefully *the text of each exercise before solving it!*
- *For exercises 2 and 3, 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. It is also worth to write two or three lines describing informally what the algorithm is supposed to do as well as to justify your complexity analyses.*
- *Make an effort to use a readable handwriting and to present your solutions neatly. This will make it easier to understand your answers.*

In exercise 1, CLRS refers to T.H. Cormen, Ch. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms* (both 2nd and 3rd editions).

During the exam you are allowed to consult books and notes. The use of pocket calculators is also permitted, but you are not allowed to use any electronic communication devices.

Exercise 1 [50 points in total]

1. (*8 points*)

1.1. $\lg 3n + n\sqrt{n} + n\lg(n\sqrt{n})$ is:

☐ **a)** $\Theta(n \lg n)$ [✓ **b)** $\Theta(n^{\frac{3}{2}})$] ☐ **c)** $\Theta(n^3)$ ☐ **d)** $\Theta(n^2)$

1.2. $2^n + 3n\lg n^3 + 9n^4$ is:

☐ **a)** $O(n^3)$ ☐ **b)** $O(n^4)$ ☐ **c)** $O(n \lg n)$ [✓ **d)** $O(3^n)$]

2. (*7 points*) Consider the following recurrence relation:

$$\begin{aligned} T(1) = T(2) = T(3) &= 1 \\ T(n) &= T(n-3) + n \qquad (n > 3). \end{aligned}$$

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

☐ **a)** $\Theta(n)$ [✓ **b)** $\Theta(n^2)$] ☐ **c)** $\Theta(n^3)$ ☐ **d)** $\Theta(n \lg n)$

3. (*6 points*) Consider a QUEUE ADT with the following standard operations: *enqueue*(x:*int*) and *dequeue*():*int*. Assume an efficient implementation of this ADT (for example, using a linked list). Assume also that passing a queue as an argument to a function takes $O(1)$ time, i.e., it is passed "by reference" (without creating a new copy of the queue). Consider the following algorithm:

```
PLAY(q:QUEUE, n:int):int
1  if $n = 1$ then
2      return $q.dequeue()$
3  else
4      $q.enqueue(n)$
5      return PLAY($q$, $\lfloor n/2 \rfloor$)
```

3.1. Assume that $n \geq 1$. For any q, the running time of PLAY(q, n) is:

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

3.2. Assume q is an empty queue. PLAY$(q, 8)$ is:

[✓ **a)** 8] ☐ **b)** 4 ☐ **c)** 2 ☐ **d)** 1

4. (*8 points*) Consider the QUICKSORT(A, p, r) algorithm (as defined in CLRS) and the last two lines in it: the recursive calls QUICKSORT$(A, p, q - 1)$ and QUICKSORT$(A, q + 1, r)$. Let $p = 1$ and $r = 9$. Assume that PARTITION(A, p, r) returned $q = 6$. Which of the following four arrays is a possible state of the array $A[p..r]$ just before the two recursive calls?

- ☐ **a)** `6, 3, 7, 4, 5, 12, 14, 1, 11`
- ☐ **b)** `5, 4, 3, 1, 7, 6, 11, 14, 12`
- [✓ **c)**] `4, 5, 6, 3, 1, 7, 14, 11, 12`
- ☐ **d)** `1, 3, 5, 6, 7, 4, 11, 12, 14`

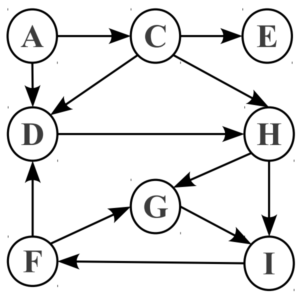
5. (*7 points*) Consider a double hashing method given by this hash function:

$$h(k, i) = (k + ih'(k)) \bmod 7, \quad \text{where} \quad h'(k) = 1 + (k \quad \bmod 6).$$

Here k is the key to be inserted and i is the probe number ($i = 0, 1, \ldots, 6$). Which of the following four arrays is the state of the hash-table after inserting keys $26, 15, 19, 14, 10$ (in this order) into an empty hash table $A[0..6]$?

	0	1	2	3	4	5	6
[✓ **a)**]	19,	15,	,	14,	,	26,	10.
☐ **b)**	10,	14,	15,	19,	26,	,	.
☐ **c)**	19,	15,	14,	,	10,	26,	.
☐ **d)**	19,	15,	10,	14,	,	26,	.

6. (*7 points*) Consider the following directed graph.

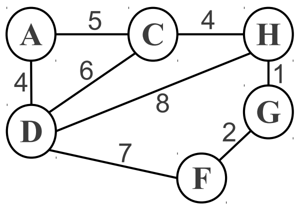


Assume that the depth-first search (DFS) algorithm starts from the vertex A and, for any vertex, the algorithm considers its adjacent vertices in alphabetical order. In which order does the DFS discover vertices (colors them gray)?

☐ **a)** A, C, D, E, H, G, I, F ☐ **b)** A, C, D, H, E, I, F, G

☐ **c)** A, C, D, H, I, G, F, E [✓ **d)** A, C, D, H, G, I, F, E]

7. (*7 points*) Consider the following weighted graph.



Consider Dijkstra's algorithm that starts from the vertex A. In which order does the algorithm extract vertices from the priority queue?

☐ **a)** A, D, C, F, H, G ☐ **b)** A, C, D, F, G, H

[✓ **c)** A, D, C, H, G, F] ☐ **d)** A, D, C, F, G, H

Exercise 2 [25 points]

Consider a square, two-dimensional array of numbers, $A[1..m, 1..m]$, representing an *elevation map* of a terrain. A cell in an array stores the elevation (height above sea level) of the corresponding square piece of the terrain.

1. (*8 points*) Write an algorithm, that given $A[1..m, 1..m]$, checks if there are two cells in it with the same elevation. Analyze the worst-case running time of your algorithm, expressed in terms of the total number of cells, $n = m^2$.

For simplicity, assume that $m = 2^k$ ($k \geq 0$). An array $A[1..m, 1..m]$ represents a *moderate terrain* if it satisfies the following property.

> If $m = 1$, the terrain is (trivially) moderate. When $m > 1$, consider the four quadrants of the array: the top left ($A[1..m/2, 1..m/2]$), the top right ($A[m/2{+}1\,..m, 1..m/2]$), the bottom left ($A[1..m/2, m/2{+}1\,..m]$), and the bottom right ($A[m/2+1\,..m, m/2+1\,..m]$). The terrain is moderate, if *both* of the following two conditions are satisfied. *First*, the average elevation in each of its four quadrants should be not more than double and not less than half the average elevation of the terrain, i.e., if e_1, e_2, e_3, e_4 are the average elevations in the four quadrants and $e = (e_1 + e_2 + e_3 + e_4)/4$ is the average elevation of the terrain, then $e/2 \leq e_i \leq 2e$ ($i = 1, \ldots, 4$). *Second*, each of the four quadrants should be a moderate terrain as well.

2. (*5 points*) Consider the following two 4x4 terrains, a) and b). For each of the two, tell if it is moderate or not. If not, explain why.

a)

2	3	5	9
1	2	5	5
4	4	3	3
4	4	9	1

b)

2	2	6	7
4	4	9	6
4	9	7	3
8	3	3	3

3. (*12 points*) Write an efficient algorithm that, given $A[1..m, 1..m]$, checks if it represents a moderate terrain. Analyze the worst-case running time of your algorithm, expressed in terms of the total number of cells, $n = m^2$.

Exercise 3 [25 points]

Consider a social-networking online service (similar, for example, to Facebook). One of the features of the service is the ability to press the "like" button on a posting of a friend. Consider analyzing a log of all "likes" by all the users of the system (as recorded for some period of time). We say that a user x *likes* a user y if x pressed "like" on at least one posting of y. More generally, a user x k-*likes* a user y, if x pressed "like" on exactly k postings of y.

1. (*5 points*) Suggest a mathematical model to capture the k-like relationships between users.

2. (*10 points*) A user y is in the m-*neighborhood* of a user x ($m \geq 1$), if there is a sequence of users $u_0 = x, u_1, \ldots, u_l = y$, such that $l \leq m$ and u_i likes u_{i+1} ($0 \leq i < l$). Provide an efficient algorithm that, given a pair of users x and y and a parameter m, checks if y is in m-*neighborhood* of x. Analyze the worst-case running time of your algorithm.

3. (*10 points*) A user y is r-*remote* from a user x, if r is the smallest number such that there is a sequence of users $u_0 = x, u_1, \ldots, u_p = y$ ($p \geq 1$), u_i k_i-likes u_{i+1} ($0 \leq i < p$), and $\sum_{i=0}^{p-1} 1/k_i = r$. To understand the intuition of the definition, consider an example with two pairs of users (a, b) and (c, d). User a 1-likes user b. There is no "like" relationship between c and d, but c 2-likes some other user u, which 2-likes d. To reward multiple likes, we say that d is as remote from c, as b is remote from a ($1/2 + 1/2 = 1$). Provide an efficient algorithm that, given a pair of users x and y, outputs all users which are not more remote from x than y. Analyze the worst-case running time of your algorithm.