Algorithms and Data Structures (DAT3/SW3/INF5/BAIT5)

Re-exam Assignments

Simonas Šaltenis

8 March 2013

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, mark the check-boxes on this paper. Remember also to put your name and your CPR number on any additional sheets of paper you will use for exercises 2 and 3.

- *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. (*7 points*)

1.1. $\sqrt{3}n^3 + \sqrt{3^n} + \lg n^{10}$ is:

☐ **a)** $O(\sqrt{2^n})$ ☑ **b)** $O(2^n)$ ☐ **c)** $O(n^{10})$ ☐ **d)** $O(n^3)$

1.2. $\lg(n5^n) + \lg n^5$ is:

☑ **a)** $\Theta(n)$ ☐ **b)** $\Theta(\lg n)$ ☐ **c)** $\Theta(n \lg n)$ ☐ **d)** $\Theta(5^n)$

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

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(\lceil n/3 \rceil) + 3 \qquad (n > 1). \end{aligned}$$

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

☐ **a)** $\Theta(n \lg n)$ ☑ **b)** $\Theta(\lg n)$ ☐ **c)** $\Theta(\sqrt[3]{n})$ ☐ **d)** $\Theta(n)$

3. (*8 points*) Consider a STACK ADT with the standard operations *push*(*x*:*int*) and *pop*():*int*, as well as an operation *top*():*int*, which simply returns an element at the top of the stack without removing it from the stack. Assume an efficient implementation of this ADT (for example, using a linked list). Also, assume that passing a stack as an argument to a function takes $O(1)$ time, i.e., it is passed "by reference" (without creating a new copy of the stack).

```
PLAY(n:int):int
 1  s:STACK
 2  for i ← 1 to n do s.push(i)
 3  for i ← 1 to n do PLAYAUX(s, s.top())
 4  return s.top()

PLAYAUX(s:STACK, k:int)
 1  t:STACK
 2  if k > 0 then
 3      for i ← 1 to k do t.push(s.pop())
 4      for i ← 1 to k do s.push(t.pop())
 5      PLAYAUX(s, k − 1)
```

3.1. PLAY(9) is:

☐ **a)** 1 ☐ **b)** 2 ☐ **c)** 8 ☑ **d)** 9

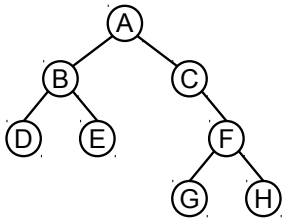
3.2. The running time of PLAY(n) is:

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

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 $q = 3$. Which of the following four arrays is a possible state of the array $A[p..r]$ just after QUICKSORT($A, p, q-1$), but before QUICKSORT($A, q+1, r$)?

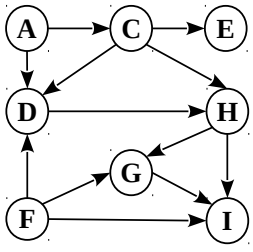
☐ **a)** `3, 1, 4, 6, 10, 7, 5, 14, 11`
☑ **b)** `1, 3, 4, 11, 10, 7, 5, 14, 6`
☐ **c)** `3, 5, 6, 4, 11, 10, 7, 1, 14`
☐ **d)** `3, 1, 4, 5, 6, 7, 10, 11, 14`

5. (*6 points*) Which one of the following four sequences is a sequence printed by a postorder tree walk of this binary tree?



☐ **a)** A, B, D, E, C, F, G, H ☐ **b)** D, B, E, A, C, G, F, H
☐ **c)** G, H, D, E, F, B, C, A ☑ **d)** D, E, B, G, H, F, C, A

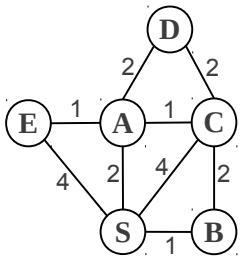
6. (*7 points*) Consider the following directed acyclic graph.



Which one of the following four sequences of vertices is topologically sorted?

- [x] **a)** F, A, C, D, E, H, G, I
- [ ] **b)** A, F, C, D, G, E, H, I
- [ ] **c)** F, A, D, C, H, G, I, E
- [ ] **d)** A, F, G, C, D, H, I, E

7. (*7 points*) Consider the Dijkstra's algorithm started at S in this weighted graph.



What are the d values of the vertices remaining in the priority queue after the *full two iterations* of the main loop of the algorithm, i.e., after S and B are removed from the priority queue? The d values are given in parentheses.

- [ ] **a)** $A(2), C(3), E(3), D(4)$
- [ ] **b)** $A(2), C(4), E(4), D(\infty)$
- [ ] **c)** $A(2), C(3), E(4), D(6)$
- [x] **d)** $A(2), C(3), E(4), D(\infty)$

Exercise 2 [25 points]

An array $A[1..n]$ of numbers is called a *centered heap*, if two conditions are satisfied. First, the element in the middle of the array $A[\lfloor (1+n)/2 \rfloor]$, which is called the *root*, is a largest number in the array. Second, the part of the array to the left of the root ($A[1\,..\,\lfloor (1+n)/2 \rfloor - 1]$) as well as the part of the array to the right of the root ($A[\lfloor (1+n)/2 \rfloor + 1\,..\,n]$) are both centered heaps (or empty). Note that this structure is similar to the standard max-heap, but its layout in the array is different.

1. (*5 points*) Arrange the following array of numbers into a centered h
   $[4, 7, 2, 6, 3, 1, 9, 8, 5]$.

2. (*10 points*) Design a procedure that removes a maximum element from a centered heap and inserts a new element v into the heap. More specifically, write an efficient algorithm that, given a centered heap $A[1..n]$ and a number v, changes the value at the root of the heap to v and "heapifies" the array, i.e., rearranges it, if necessary, so that it remains a centered heap. Analyze the worst-case running time of your algorithm.

3. (*10 points*) Write an efficient algorithm that, given an array $A[1..n]$, checks if the array is a centered heap. Analyze the worst-case running time of your algorithm.

Exercise 3 [25 points]

Consider a network of roads in an area of a developing country with some villages and roads connecting them. Only some of the roads are paved with asphalt, others are unpaved roads (dirt roads). Each road starts and ends at a road intersection which may be a village or a simple intersection without a village. The length of each road is known.

1. (*5 points*) Suggest a mathematical model of the problem.

2. (*10 points*) A set of villages is called a *civilization island* if it is possible to get from any village to any other village in the set without using unpaved roads. A civilization island contains a single village if it is not possible to go from this village to any other village using only asphalt roads. Given a road network, write an efficient algorithm that counts how many civilization islands it contains. Analyze the worst-case running time of your algorithm.

3. (*10 points*) For simplicity, let us assume that each road intersection is a village. Assume also that there are no asphalt roads in some area, i.e., all roads are unpaved. Given the road network of such an area, write an efficient algorithm that computes which roads should be paved with asphalt so that (i) all villages are joined into one civilization island and (ii) the smallest possible amount of kilometers of roads are paved with asphalt. Analyze the worst-case running time of your algorithm. If there are n villages, how many roads will your algorithm suggest to pave with asphalt?