Algorithms and Data Structures (DAT1/SW3)

Autumn Semester 2003

Exam Assignments

Simonas Šaltenis

21 January 2004

Full name:	
CPR-number:	

This exam consists of three exercises. When answering the questions from exercise 1, write in the provided fields on this paper.

- *Read carefully the text of the exercises before solving them.*
- *For exercises 2 and 3, it is important that you argue for your answers, and that your solutions are presented in a readable form. In particular, you should provide precise descriptions of your algorithms using pseudo-code.*
- *Make an effort to use a readable handwriting and to present your solutions neatly. This will make it easier to understand your answers, and thus to mark your exams.*

In exercise 1, "the course textbook" refers to T.H. Cormen, Ch. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms*.

During the exam you are allowed to consult books and notes. The use of pocket calculators is also permitted.

Exercise 1 [50 points in total]

1. (*6 points*)
$3^n + n^3 \log n$ is:

☐ **a)** $\Theta(n^3 \log n)$ [✓ **b)** $\Theta(3^n)$] ☐ **c)** $\Theta(n^3)$ ☐ **d)** $\Theta(9^n)$

2. (*8 points*)
Consider a recurrence relation (for simplicity, assume that n is a power of 2):

$$\begin{aligned} T(1) &= 4 \\ T(n) &= 4T(n/2) + 4n \qquad (n > 1)\ . \end{aligned}$$

$T(n)$ is

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

3. (*6 points*)
Here is the Heapsort algorithm, as presented in the course book (assume that *A.length* is the length of the array A and *A.heap-size* is the number of elements arranged into a max-heap in the array A):

```
HEAPSORT(A)
1  BUILD-MAX-HEAP(A)
2  for $i \leftarrow A.length$ downto 2 do
3      exchange $A[1] \leftrightarrow A[i]$
4      $A.heap\text{-}size \leftarrow A.heap\text{-}size - 1$
5      MAX-HEAPIFY(A, 1)
```

As defined in the course book, Build-Max-Heap(A) rearranges all elements of A into a max-heap with the root at $A[1]$ and Max-Heapify(A, k) moves $A[k]$ into the right place in the heap in A. Which of the following arrays can possibly be an array after line 5 has been executed two times? Mark exactly one array.

☐ **a)** `6, 8, 9, 3, 4, 5, 2, 1`
[✓ **b)** `6, 4, 5, 2, 3, 1, 8, 9`]
☐ **c)** `1, 3, 2, 4, 5, 6, 8, 9`
☐ **d)** `5, 3, 4, 1, 2, 6, 8, 9`

4. (*5 points*)
Consider a Queue ADT with the following standard operations: *init*(), *enqueue*(x:*int*), *dequeue*():*int*, and *front*():*int*. Here, *dequeue*() both removes an element and returns it as a result, *front*() returns the element at the front (head) of the queue. Write the sequence of values stored in the queue Q after Play(Q) has been executed. The front of the queue is on the left.

```
Play($Q$ : Queue)
$Q.init()$
for $i \leftarrow 1$ to $4$ do
    $Q.enqueue(i)$
while $Q.front \neq 3$ do
    $Q.enqueue(Q.dequeue())$
```

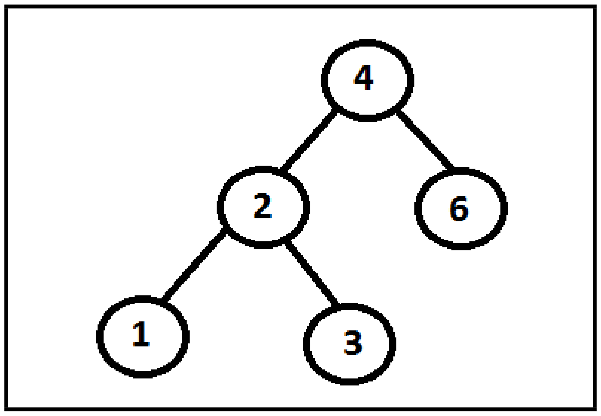
3 | 4 | 1 | 2

5. (*5 points*)
Write the result after inserting the numbers 7, 6, 1, 12 (in this order) into a hash table $A[0..4]$ with the hash function $h(n) = n \bmod 5$. Use linear probing to resolve collisions.

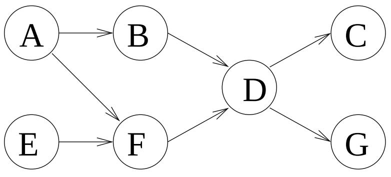
0	1	2	3	4
	6	7	1	12

6. (*5 points*)
Draw the resulting binary search tree after performing the following sequence of insert and delete operations:

insert 4,
insert 8,
insert 2,
insert 6,
insert 3,
insert 1,
delete 8.

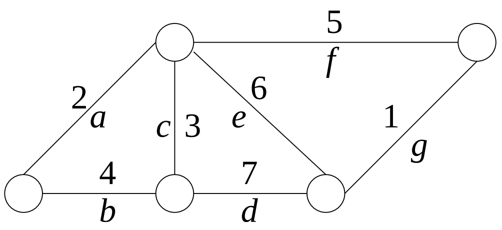


7. (*6 points*)
Write a topologically sorted sequence of vertices of the following directed acyclic graph.



E | A | B | F | D | C | G

8. (*7 points*)
From the set of the edges a, b, c, d, e, f, g list exactly those which are part of a minimum spanning tree of the following graph.



g | a | c | f

Exercise 2 [25 points]

Consider a country subdivided into a hierarchy of smaller administrative units. For example, Denmark is subdivided into counties (amter), which are further subdivided into municipalities (kommuner), which may be further subdivided into postal districts and so on.

A *geographic region* is a country or a smaller administrative region. We say that a geographic region r is subdivided into a number of *sub-regions* and each of these subregions has r as their *father-region*. For example, Nordjyllands amt is the father-region of Aalborg kommune, and Aalborg kommune is a sub-region of Nordjyllands amt.

A geographic region is represented by an instance of a GeoRegion abstract data type with the following operations:

area():*int*
: $r.area()$ returns a positive integer—the area of the geographic region r measured in square kilometers.

firstSubregion(): *GeoRegion*

r.fistSubregion() returns the first region in the list of sub-regions of the geographic region *r*. It returns *nil*, if the region *r* is not subdivided into smaller regions.

siblingSubregion(): *GeoRegion*

s.siblingSubregion() returns the next region (in an arbitrary order) in the list of sub-regions that share the same father-region as *s*. For example, Hals kommune could be a sibling sub-region of Aalborg kommune. It returns *nil*, if this is the last sub-region in this list or if *s* is not a part of any larger region.

removeSubregion(*s*: *GeoRegion*)

r.removeSubregion(*s*) removes the geographic region *s* from the list of sub-regions of *r* (if *s* is in that list).

addSubregion(*s*: *GeoRegion*)

r.addSubregion(*s*) adds the geographic region *s* to the end of the list of sub-regions of *r*. If *s* was a sub-region of another region r', the operation first removes *s* from the list of sub-regions of r'.

The operations listed above are the only ones that you are allowed to use to access the information about a geographic region or to modify it!

All of the operations are implemented so that they have $O(1)$ worst-case complexity. In your complexity estimates assume that there are in total n_R sub-regions in the whole hierarchy of sub-regions of a given region R.

1. Write an algorithm that counts how many large sub-regions a geographic region R has? A region is called large if its area is more than 1000km^2.

2. The area of a geographic region is *correctly computed* if its area is equal to the sum of the areas of its sub-regions *and* the areas of its sub-regions are all correctly computed. If the region does not have any sub-regions, its area is always said to be correctly computed. Write an algorithm that checks if the area of a geographic region R is correctly computed. What is the worst-case complexity of your algorithm?

3. If a geographic region S has only one sub-region r, the sub-region r is called a *non-dividing* sub-region. Write an algorithm that removes all non-dividing sub-regions from the hierarchy of sub-regions of a geographic region R. When a region r is removed, if it has sub-regions, they should become sub-regions of the father-region of r. What is the worst-case complexity of your algorithm?

Exercise 3 [25 points]

Let us consider a finite number of towns. Some of them are connected by roads. There are two types of roads: *one-way* roads (can be used to drive in one direction) and two-way roads (can be used in both directions). A town T is called a *metropolitan city* if and only if it is possible to drive from every other town to T.

Here is an example. Assume that there are four towns called Aalborg, Hadsund, Randers, and Skive. Aalborg and Hadsund are connected by a two-way road, as well as Aalborg and Randers are connected by a two-way road. There is also a one-way road from Skive to Hadsund and from Skive to Randers. There are no other roads.

In this example Aalborg is a metropolitan city, but Skive is not.

1. In the given example, is Hadsund a metropolitan city?
2. Suggest a mathematical formalization of the problem. If needed, you can choose different formalizations suitable for algorithms in questions 3 and 4. Show, how the example above is represented in your model(s). A drawing is sufficient.
3. Give an algorithm that checks if a given town is a metropolitan city. What is the worst-case complexity of your algorithm?
4. Assume that the lengths of all roads are given. All towns that can be reached by driving less than 30km from a town T are said to be in the *suburbs* of T. Give an algorithm that returns the number of towns in the suburbs of a given town. What is the worst-case complexity of your algorithm?

In questions 3 and 4, specify clearly what are the inputs of your algorithms.