# Exam in Algorithms & Datastructures I (1DL210) Spring 2009

Prepared by Pierre Flener

1 June 2009, from 14:00 to 19:00, at Gimogatan 4, Room 2

**Rules:** This is a *closed*-book exam. The usage of electronic devices is *not* allowed.

| Question | G points | Your G | VG points | Your VG |
|---|---|---|---|---|
| 1 | 11 | | 17 | |
| 2 | 12 | | 0 | |
| 3 | 8 | | 0 | |
| 4 | 10 | | 0 | |
| 5 | 10 | | 0 | |
| 6 | 12 | | 0 | |
| 7 | 7 | | 13 | |
| Total: | 70 | | 30 | |

**Read carefully:** Provide only the requested information and nothing else. Unreadable, unintelligible, and irrelevant answers will not be considered. Be concise and write each answer immediately behind its question and **do not attach any extra solution or draft pages**: if your answer does not fit into the provided space, then it is unnecessarily long and maybe you should reread the question. Always show **all** the details of your reasoning, and make explicit **all** your assumptions. This question set is double-sided.
**Grading:** A mark of at least 49 G points earns 4 higher-education credit points (4 hp) and a 3 passing grade. Any other mark on the G points earns a U failure grade. Earned VG points do normally *not* compensate for missing G points, while every G point in excess of 49 G points counts as 0.5 VG points. If passing, then a mark from 11 to 25 VG points earns an upgrade to a 4 passing grade, and a mark of at least 26 VG points earns an upgrade to a 5 passing grade.
**Help:** Normally, the main instructor will attend this exam from 16:00 to 17:00.

**Name & Personal Number:** .....

**Question 1: Fact or Fiction?** (11 G points) (17 VG points)
Justify your yes/no answers to the following questions. If your answer is "no", then you must **exhibit a counter-example**, when relevant, that was **not discussed** in the course. Correct answers with a missing justification or counter-example get zero points.

a. **Complexity:** Alva claims she can solve the bingolotto problem for $n$ elements, a problem long known to take $\Theta(n^4 \cdot \sqrt{n})$ time at best, with an algorithm that takes $O(n^5 \cdot \log n)$ time at worst. Can Alva be right? (2 G points)

b. **Cutting da Crap?** Dan Brown, author of *The Da Vinci Code*, writes in Chapters 4 and 5 of his book *Digital Fortress* that a 64-bit key decrypting (using for instance the *xor* technique seen in the course) an encrypted text is routinely found by brute-force guessing in about ten minutes on the fictitious TRANSLTR computer, which has three million CPUs working in parallel, and that a particular million-bit key was found in three hours. Using a recurrence relation and closing it, answer the following question: Is the average time for million-bit keys higher than three hours? Identify any irrelevant technical facts in the question. (7 VG points)

c. **Sorting:** Does quicksort take at worst $O(n \cdot \log n)$ time, where $n$ is the number of elements? (1 G point)

d. **Sorting:** After stating the time complexities of dividing an $n$-element list in the middle and dividing an $n$-element array in the middle, answer the following question: When using such a division in the middle, does merge-sorting a list take the same asymptotic runtime as merge-sorting an array? (3 G points)

e. **Queues:** Ali claims he can implement both the *dequeue* and *enqueue* operations of first-in first-out queues to run in $O(\log \sqrt{n})$ time on average, using a new data structure he found in the *Wikipedia*. Can Ali be right? (2 G points)

f. **Trees:** Does a non-empty binary tree where all nodes have 0 or 2 children have one more leaf than non-leaf nodes? (3 VG points)

g. **Trees:** Is every binomial tree also a balanced tree, under some suitable generalisation of the AVL balancing property to $k$-ary trees, with $k \geq 2$? (2 VG points)

h. **Trees:** Is every binomial tree also a search tree, under some suitable generalisation of binary search trees to $k$-ary search trees, with $k \geq 2$? (2 VG points)

i. **Hashing:** Does search in a hash table under chaining take at most constant time? (2 G points)

j. **Hashing:** Does insertion into a non-full hash table under open addressing with quadratic probing but without rehashing never give an overflow? (3 VG points)

k. **Greediness:** Do greedy algorithms always give optimal solutions? (1 G point)

**Question 2: Analysis** (12 G points) (0 VG points)
Consider the following function, which takes an argument $n = 2^m$ for some $m > 0$:

```
fun g 2 = 3
  | g n = let val b = 3 * g (n div 2) in b*b*b end
```

Answer the following sub-questions:

a. Construct a recurrence (not for the time taken but) for the number $MD(n)$ of integer multiplications and integer divisions made when computing `g(`$n$`)`. First make explicit the costs of *dividing*, *conquering* (recurring), and *combining* of the underlying divide-and-conquer algorithm, and then simplify the expression. (3 G points)

b. Compute a tight asymptotic bound $\Theta(\dots)$ on $MD(n)$. Use the Master Theorem, if it is applicable, and assume the regularity condition holds, if need be, but do **not** use the table of common special cases seen in the course; otherwise, first state why the Master Theorem is not applicable and then use any other relevant method or theorem. (7 G points)

c. Briefly outline how your answers to sub-questions a and b change when the expression `3 * g (n div 2)` in the function declaration above is replaced by the equivalent expression `g (n div 2) + g (n div 2) + g (n div 2)`. (2 G points)

**Question 3: Complexity** (8 G points) (0 VG points)
Consider an abstract datatype for an integer set (*mängd*) with the following operations:

- *insert*$(s, i)$ returns a copy of set $s$ with integer $i$ inserted ($s \cup \{i\}$), but raises exception *Overflow* if not enough memory is left for this insertion
- *member*$(s, i)$ returns *true* if integer $i$ is in set $s$, and returns *false* otherwise ($i \in s$)
- *remove*$(s, i)$ returns a copy of set $s$ with integer $i$ removed ($s \setminus \{i\}$)
- *findMin*$(s)$ returns the minimum element in non-empty set $s$ ($\min s$)
- *extractMin*$(s)$ returns a copy of non-empty set $s$ with its minimum element removed ($s \setminus \{\min s\}$)
- *decSort*$(s)$ returns an array containing the elements of set $s$ in decreasing order

Recall that a set cannot contain duplicate elements.

Consider the following data structures for implementing this abstract datatype:

1. Array*
2. Decreasingly sorted array*
3. Hash table
4. AVL tree
5. Decreasingly sorted list
6. Binomial min-heap

*Consider only arrays that must always remain filled from the left, and where we maintain the index of the rightmost actually used cell.

A *cheap* operation on a data structure with $n$ elements has an **average**-case runtime of $O(1)$ or $O(\log n)$. For each application scenario below, list the numeric identifiers of **all** the data structures enumerated above that exhibit the desired behaviour, and state any assumptions you made:

a. We want at least *member* to be cheap. (3 G points)

b. We want at least *member*, *insert*, and *remove* to be cheap. (1 G point)

c. We want at least *findMin* and *extractMin* to be cheap. (3 G points)

d. We want at least *decSort* to be cheap. (1 G point)

**Question 4: AVL Trees** (10 G points) (0 VG points)
Starting from the empty AVL tree, perform the following sequence of operations, using algorithms seen in the course:

a. Insert 2, 6, 0, 7, 5, and 3, in this order. For each insertion, first identify any imbalance that temporarily occurs, and then re-balance the tree, showing the result of each elementary operation and explaining what you are doing. (We will **not** check whether you correctly continued from any error.) (5 G points)

b. Ignoring the AVL invariant, delete the 5 from your resulting AVL tree, showing the result of each elementary operation and explaining what you are doing. (5 G points)

**Question 5: Binomial Heaps** (10 G points) (0 VG points)
Starting from the empty binomial **min**-heap, perform the following sequence of operations, using the algorithms seen in the course:

a. Insert 4, 12, 8, 24, 6, 18, and 16, in this order. You just need to draw the **min**-heap after inserting the 24, and again after inserting the 16. (We will **not** check whether you correctly continued from any error.) (2 G points)

b. Delete the **minimum** from your resulting **min**-heap, showing the result of each elementary operation and explaining what you are doing. (8 G points)

**Question 6: Hash Tables** (12 G points) (0 VG points)
Given the initial hash table below of $m = 7$ cells (where $\perp$ denotes a cell that was never used) and using the ordinary hash function $hash'(key)$ = "the last digit of $key$" under **open addressing** with the **quadratic** probing function $f(i) = i^2$ and **with rehashing**, perform the **sequence** of operations below. For each operation, state at which indices probes are made for which values of probe number $i$, and give the resulting hash table, filling each cell with a numeric value or one of the special values $\perp$ and $\Delta$ (denoting a deleted value). At every operation, state any observations and assumptions you make.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $\perp$ | 11 | 31 | $\perp$ | $\perp$ | 21 | $\perp$ |

Delete 31:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| | | | | | | |

Insert 41:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| | | | | | | |

Insert 25:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| | | | | | | |

Insert 15:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| | | | | | | |

**Question 7: Compression** (7 G points) (13 VG points)
Consider an alphabet consisting of just the characters in the text below, **including** the space symbol between the words, denoted by "·" in the tables further down:

NOT TO BE OR TO BE OR NOT OR TO BE

Answer the following sub-questions (we will **not** check whether you correctly continued from any error):

a. In the table below, give the greedily constructed Huffman code for compressing the text above. Toward this, start from the list of character-frequency pairs in **alphabetical** order on the characters (with · before B), and use a priority queue that orders elements of equal priority in their **temporal** order of arrival. (7 G points)

| · | B | E | N | O | R | T |
|---|---|---|---|---|---|---|
| | | | | | | |

b. Give the uncompressed text for the following Ziv-Lempel codeword sequence over the same alphabet

6 4 0 1 2 0 4 5 0 3 4 6 0 7 9 11 13 19 8 10 12 14 16 6

and fill in the relevant cells of the dictionary below. (9 VG points)

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| · | B | E | N | O | R | T | | | |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| | | | | | | | | | |
| 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 |
| | | | | | | | | | |
| 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | etc |
| | | | | | | | | | |

c. Which of the Huffman and Ziv-Lempel algorithms gives the smallest compressed file for the uncompressed texts in sub-questions a and b above? State any assumptions you make. (4 VG points)