Delft University of Technology
EEMCS Faculty

TU Delft

Resit exam A&D (TI1316/TI1316TW)

3 July 2018, 18:30–21:30

$\left(1\cdot 0{,}35 + \frac{18}{32}\cdot 0{,}35\right.$ [illegible] $\left. + 1\cdot 0{,}3\cdot 0{,}5\cdot 0{,}4\right)\cdot 0{,}8$
$+ 1 + 1$

Examiners:

Examiner responsible:	Joana Gonçalves and Robbert Krebbers
Examination reviewer:	Stefan Hugtenberg

Parts of the examination and determination of the grade:

Exam part	Number of questions	Question specifics	Grade
Multiple-choice (paper)	24 questions (equal weights)	One correct answer per question	35%
Open questions (paper)	3 questions	Multiple parts	35%
Implementation (Weblab)	2 questions (Weblab weights)	Multiple parts	30%

Use of information sources and aids:

- Any version of the book Algorithms & Data Structures in Java by Tamassia et al. is allowed.
- The use of of notes is not permitted.
- Scrap paper sheets are provided at the beginning of the exam. Additional scrap paper can be requested.
- The use of any devices other than the assigned computer is not permitted.

Additional instructions:

- Solve the exam on your own. Any form of collaboration is fraud.
- You may not leave the examination room during the first 30 minutes.
- You will not be allowed to start the exam if you arrive after the first 30 minutes.
- If you are eligible for extra time, show your "Verklaring Tentamentijd Verlenging" to the surveillants.

Instructions on the paper (analysis) part:

- Write your name and student number on every sheet of paper.
- Write the total number of sheets of paper you hand in.
- **For multiple-choice questions**, the order of the choices on the answer form **might not be A-B-C-D!**
- Tip: mark multiple-choice answers on this exam paper first, copy them to the answer form after revising.
- **For open questions**, provide all requested information and always give an explanation. Avoid irrelevant data, it could lead to deductions.
- **For proofs**, make sure your proof is properly structured and sufficiently explained. Statements or steps without justification could lead to point deductions.

Instructions on the Weblab (implementation) part:

- Make sure that all of your code compiles. If it does not compile, you get no points.
- **Computer login:** log in to the computer using the following details:

Username:	`EWI_Weblab`
Password:	`Tentamen01`

- **Weblab login:** log in to WebLab as usual using Single Sign-On.
- **Weblab exam:** access the exam assignment in the correct version of the course:

CS students:	`https://weblab.tudelft.nl/ti1315/2017-2018/` or
Math students:	`https://weblab.tudelft.nl/ti1315/TW+2017-2018/`

- **Weblab exam registration:** access your exam submission (pencil icon), register using your Weblab key.
- The Java API documentation can be found at `https://weblab.tudelft.nl/java8/api/`
- Busy Weblab server: if this happens, you might not be able to compile your solution immediately. Please save it, so you can compile and run it later. You can open other tabs and continue working while waiting.
- Do not close the browser after registering for the exam. If you do so accidentally, ask the surveillants.
- Questions: use the comment functionality of Weblab to ask for clarification about phrasing of the exam.

This page is intentionally left blank.

Multiple-choice questions (35%, 24 points)

1. (1 point) What is the asymptotic relationship between the functions $(c+2)^n$ and $(n+2)^k$? Assume constants $k \geq 1$ and $c > 1$. $n_0 \geq 1$

   A. $(c+2)^n$ is $\mathcal{O}((n+2)^k)$.

   [✓ B. $(c+2)^n$ is $\Omega((n+2)^k)$.]

   C. $(c+2)^n$ is $\Theta((n+2)^k)$.

   D. None of the above.

2. (1 point) Consider that the amortized time complexity of n operations is $O(nm)$. Which of the following statements **is false**?

   A. Some of the n operations can run in $\mathcal{O}(1)$ time.

   B. Each of the n operations takes $\mathcal{O}(m)$ time on average.

   C. Each of the n operations involves on average cm instructions, with constant $c \geq 1$.

   [✓ D. None of the n operations can run in $\mathcal{O}(m^2)$ time.]

3. (1 point) Consider the implementation of class DataS below.

```
public class DataS<E> {
  private int f;
  private E[] elements;

  public DataS<E>(int capacity) {
    elements = new E[capacity];
    f = -1;
  }

  public void insert(E x) {
    if (f >= elements.length)
      return;
    f = f+1;
    elements[f] = x;
  }

  public E remove() {
    if (f < 0)
      return null;
    f = f-1;
    return elements[f+1];
  }
}
```

   What type of data structure does class DataS above implement?

   A. Priority queue.

   B. Queue.

   C. Deque.

   [✓ D. Stack.]

4. (1 point) Consider class DataS from question 3. Assume that we insert all elements of a sequence of integers S into a data structure of type DataS and then perform an equal number of remove operations. In what order are the elements of S returned?

   A. Same order as in S.

   [✓ B. Reverse order.]

   C. Increasing order.

   D. Decreasing order.

5. (1 point) Consider an algorithm to move the last but one element of a sequence S with n elements to the front of that sequence. Example: if $S = \{1, 2, 3, 4, 5\}$, the algorithm should change S into $\{4, 1, 2, 3, 5\}$. What is the complexity of the most time-efficient algorithm for this operation when S is implemented by an array, a singly-linked list, or a doubly-linked list?

   A. **array:** $\mathcal{O}(1)$ **singly-linked list:** $\mathcal{O}(1)$ **doubly-linked list:** $\mathcal{O}(1)$
   B. **array:** $\mathcal{O}(1)$ **singly-linked list:** $\mathcal{O}(n)$ **doubly-linked list:** $\mathcal{O}(1)$
   [✓ C.] **array:** $\mathcal{O}(n)$ **singly-linked list:** $\mathcal{O}(n)$ **doubly-linked list:** $\mathcal{O}(1)$
   D. **array:** $\mathcal{O}(n)$ **singly-linked list:** $\mathcal{O}(n)$ **doubly-linked list:** $\mathcal{O}(n)$

6. (1 point) What is the best time complexity to access the last element (rightmost leaf) in a heap containing n nodes, implemented by a linked tree structure without an explicit reference to the last node (tip: algorithm using binary encoding of tree paths)?

   A. $\mathcal{O}(1)$
   [✓ B.] $\mathcal{O}(\log_2 n)$
   C. $\mathcal{O}(n)$
   D. $\mathcal{O}(n \log_2 n)$

7. (1 point) Consider the most time-efficient algorithm for finding the second largest value in a sequence of integers S using only $\mathcal{O}(1)$ additional space. On which of the following data structures storing S **is it always impossible** to do this in $O(\log_2 n)$ time?

   A. Max-heap. – bovenste 3
   B. Binary tree.
   C. AVL tree.
   D. Red-black tree.

8. (1 point) Consider that a recursive algorithm has run-time recurrence equation $T(n) = 4T(\lfloor n/4 \rfloor) + k_1$, with constant $k_1 \geq 1$. What is the recurrence equation $S(n)$ denoting the **minimum** space complexity used by the same algorithm, assuming constant $c_1 \geq 1$?

   [✓ A.] $S(n) = S(\lfloor n/4 \rfloor) + c_1$.
   B. $S(n) = 2S(\lfloor n/2 \rfloor) + c_1$.
   C. $S(n) = 2S(\lfloor n/4 \rfloor) + c_1$.
   D. $S(n) = 4S(\lfloor n/4 \rfloor) + c_1$.

9. (1 point) Which of the following statements on sorting algorithms **is false**?

   [✓ A.] Quick sort sorts in the partition step, while merge sort sorts in the combining step.
   B. Sorting with a priority queue can be done in $\mathcal{O}(n \log_2 n)$ time.
   C. To sort variable-length keys, radix sort applies bucket sort from the least to the most significant key (right to left).
   D. When sorting variable-length keys, radix sort may not need to process all individual keys.

10. (1 point) Which sorting algorithm is best suited to sort a sequence that **does not** fit into main memory?

    A. ~~Heap sort~~.
    B. Merge sort.
    [✓ C.] Quick sort.
    D. ~~Radix sort.~~

11. (1 point) Which algorithm has the best **expected** time complexity to find the median element of an unordered sequence S with n elements, assuming that n is odd?

    A. Binary search.
    B. Quick select. ← ?
    C. Quick sort followed by retrieval of the $\lceil n/2 \rceil^{th}$ element.
    D. Build a min-heap and extract $\lceil n/2 \rceil$ elements. The median is the $\lceil n/2 \rceil^{th}$ element.

12. (1 point) Consider you need to keep track of a large number of scores for two competitions, **FootLeague** and **CodingCup**. You will use a data structure per competition to store n pairs (t, s), where t is a team name and s the corresponding score. You will access and update scores, and users will often check the score of their favorite team as the competition progresses. For **FootLeague**, you are asked to keep the teams sorted in lexicographic/alphabetic order, so they can be retrieved in that order in $\mathcal{O}(n)$ time. For **CodingCup**, this is not necessary. Which data structures provide the best time complexity in each case?

    ~~A.~~ **FootLeague**: Sorted array list. **CodingCup**: Doubly-linked list.
    B. **FootLeague**: Array-based heap. **CodingCup**: Linked tree heap.
    [✓ C.] **FootLeague**: AVL tree. **CodingCup**: Hash table.
    D. **FootLeague**: Red-black tree. **CodingCup**: AVL tree.

13. (1 point) Which of the following statements about hash functions and hash tables **is false**?

    A. A hash function takes a key of arbitrary length and generates a fixed length code.
    B. A collision happens when `a.equals(b)` is false and `a.hashCode() == b.hashCode()` is true.
    [✓ C.] The load factor of a hash table is the ratio n/C between the number of entries n and the capacity of the hash table C.
    ~~D.~~ Open addressing with linear probing uses more space than chaining.

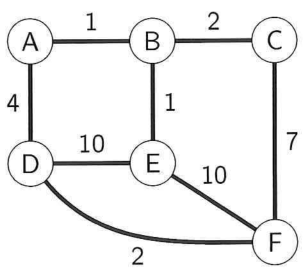
14. (1 point) Consider the following fixed size hash table using linear probing (associated values are omitted):

0	1	2	3	4
8		12	7	19

The hash function is $h(k) = k \mod 5$. Which of the following sequences denotes a valid order by which entries were inserted in an initially empty hash table?

    A. $\{19, 12, 8, 7\}$
    B. $\{12, 19, 8, 7\}$
    [✓ C.] $\{19, 12, 7, 8\}$
    D. $\{12, 8, 7, 19\}$

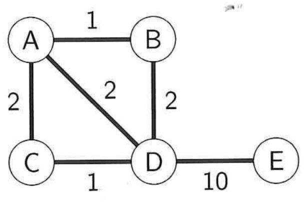
15. (1 point) Consider the following weighted graph:



When performing Dijkstra's algorithm starting from vertex A, how many **non-infinite** distinct labels will the vertex F have?

    [✓ A.] 1
    B. 2
    C. 3
    D. 4

16. (1 point) Consider the following weighted graph:



Applying Kruskal's algorithm on the above graph will identify a *single* minimum spanning tree. However, in general, a graph can have multiple minimum spanning trees. How many different minimum spannings trees does the above graph have?

A. 1

B. 2

C. 3

D. 4

17. (1 point) In general, a directed acyclic graph (DAG) can have multiple topological orders. If a DAG G has exactly one topological order, which of the following statements about properties of G is **true**?

A. G is a forest.

B. G is a tree.

C. G has no self-loops.

D. All of the above.

18. (1 point) Assume we have a large graph under the following conditions: fixed number of vertices, dynamic number of edges (frequent edge insertions and deletions), and frequent calls to method `getEdge(u,v)`. Which graph data structure is most time-efficient under this scenario?

A. Edge list.

B. Adjacency matrix.

C. Adjacency list.

D. Adjacency map.

19. (1 point) Consider a shortest path p from a vertex s to some other vertex t in a weighted undirected graph G, without negative weights. Under which conditions will p still be a shortest path from s to t?

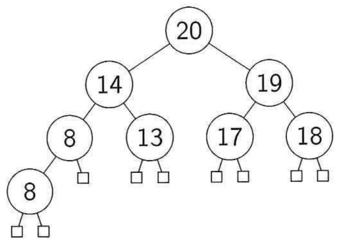
A. If the weights of all edges in G are increased by one (i.e. weight w is replaced by weight $w+1$).

B. If the weights of all edges in G are squared (i.e. weight w is replaced by weight w^2).

C. If the weights of all edges in G are increased by two (i.e. weight w is replaced by weight $w+2$).

D. If the weights of all edges in G are multiplied by two (i.e. weight w is replaced by $2w$).

20. (1 point) Consider the following tree:



Which of the following is **true**?

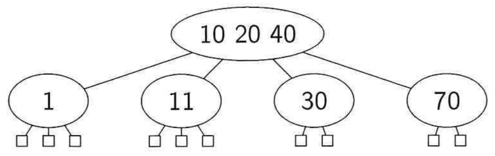
A. The tree is an AVL tree.

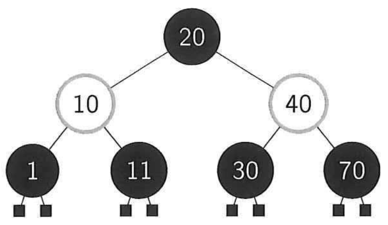
B. The tree can be colored as a red-black tree.

C. The tree is a minimum-oriented heap.

[✓ D.] The tree is a maximum-oriented heap.

21. (1 point) Consider the following (2,4) tree on the left and binary tree on the right. In the binary tree on the right, filled circles denote black nodes and unfilled circles denote red nodes.





Which of the following statements is **true**?

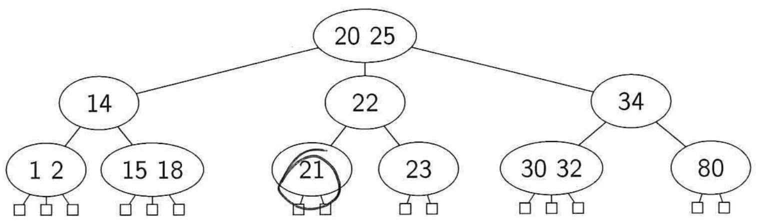
A. The binary tree is a red-black tree, which represents the given (2,4) tree. At the same time, there is at least one other red-black tree that represent the given (2,4) tree.

B. The binary tree is a red-black tree, which uniquely represents the given (2,4) tree.

[✓ C.] The binary tree is a red-black tree, which does not represent the given (2,4) tree.

D. The binary tree is not a red-black tree.

22. (1 point) Consider the following (2,4) tree:



When deleting **21** from the tree, how many underflows are caused in the tree?

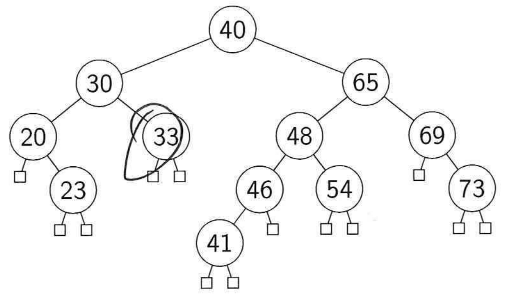
[✓ A.] One underflow, which needs to be fixed by a fusion.

B. Two underflows, which need to be fixed by a transfer and fusion.

C. Two underflows, which need to be fixed by two fusions.

D. Two underflows, which need to be fixed by two transfers.

23. (1 point) Consider the following AVL tree:



When deleting **33** from the given AVL tree, how many tri-node restructurings are caused in the tree? If the node to be deleted has two children, the node will be replaced with the in-order predecessor (i.e. the maximal node in the left child).

A. No tri-node restructuring.

B. One tri-node restructuring.

[✓ C.] Two tri-node restructurings.

D. Three tri-node restructurings.

24. (1 point) Given a binary search tree, what traversal should be used to obtain a sorted sequence of the keys in the tree?

    [✓ A.] Pre-order traversal.

    B. Post-order traversal.

    C. In-order traversal.

    D. Depth-first traversal.

Open questions (35%, 32 points)

25. Consider the following Java implementation of a recursive algorithm.

```
public static int precursive(int n) {                            1
  if (n == 0) return 1;                                          2
  return precursive(n-1) + precursive(n-1);                      3
}                                                                4
```

4

(a) (4 points) State the base and recurrence equations for the **time** complexity of method `precursive` as a function of n. Refer to the relevant parts of the code to justify your answer.

6 4

(b) (6 points) Derive the closed form of the given recurrence equation. You should **either**: guess the closed form and prove its correctness by induction, **or** derive the closed form by repeated unfolding.

(c) ~~(2 points)~~ State the tightest worst-case Big-$\mathcal{O}$ **time** complexity of `precursive` in function of n. Justify your answer.

(d) ~~(2 points)~~ Explain what the algorithm `precursive` calculates. Describe a faster solution to perform the same calculation and state its tightest worst-case **time** complexity.

26. Method `operateOnMatrix` below performs an operation on a symmetric n by n matrix (row x column).

```
public class SymmetricMatrix {                                   1
  private int[][] matrix;                                        2
  /* ... */                                                      3
  public static int[] operateOnMatrix(int x) {                   4
    for (int i = 0; i < matrix.length; i++) {                    5
      int j = operateOnArray(matrix[i], i, x);                   6
      if (j > -1)                                                7
        return new int[]{i,j};                                   8
    }                                                            9
    return null;                                                 10
  }                                                              11
                                                                 12
  private static int operateOnArray(int[] row, int startIdx, int x) {  13
    for (int j = startIdx; j < row.length; j++)                  14
      if (row[j] == x)                                           15
        return j;                                                16
    return -1;                                                   17
  }                                                              18
}                                                                19
```

Entries in a symmetric matrix are symmetric with respect to the main diagonal. Example: $M = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{pmatrix}$

(a) (4 points) Write the polynomial for the worst-case **time** complexity of method `operateOnMatrix` as a function of n. Define all constants. Explain each term of the polynomial, refer to lines of code.

1

(b) (4 points) Simplify your polynomial expression as much as possible. Derive and explain the tightest worst-case **time** complexity of method `operateOnMatrix` in Big-$\mathcal{O}$ notation.

(c) (3 points) Identify what `operateOnMatrix` does. Describe a faster solution to perform the same operation only on the upper triangle of the matrix assuming that the elements of each row are sorted, and state its tightest worst-case **time** complexity in Big-$\mathcal{O}$ notation.

27. Prove that $2^{n+3} + 2^{\log_2 n}$ is $\Theta(2^n)$.

3

(a) (3 points) State in detail the mathematical conditions that should be proved.

4

(b) (4 points) Prove that these conditions hold. Explain all the steps in your proof.

18

Implementation questions (30%)

There are two implementation questions on Weblab.

32

End of the exam