Delft University of Technology
EEMCS Faculty

Final exam A&D (TI1316/TI1316TW)

18 April 2018, 13:30–16:30

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)	3 questions (Weblab weights)	Multiple parts	30%

Use of information sources and aids:

- Any version of the book Algorithms & Data Structures in Java by Tamassia, Goodrich, Goldwasser is allowed.
- The use of of notes is not permitted.
- Scrap paper sheets are provided in 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) Consider the functions f, g, and h, for which we know the following:

   - $f(n)$ is $\mathcal{O}(n^3)$
   - $g(n)$ is $\Theta(n^2)$
   - $h(n)$ is $\Omega(n)$

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

   A. There is an n_0 such that $f(n) \geq 2n^3$ for all $n \geq n_0$.

   B. There is an n_0 such that $g(n) \leq 2n^2$ for all $n \geq n_0$.

   C. There is an n_0 such that $g(n) \geq 2n^3$ for all $n \geq n_0$.

   D. There is an n_0 such that $h(n) \geq 2n$ for all $n \geq n_0$.

2. (1 point) Consider an algorithm with runtime $T(n)$. What can we say for sure about the space usage $S(n)$ of the algorithm?

   A. $S(n)$ is $\Omega(T(n))$.

   B. $S(n)$ is $\Theta(T(n))$.

   C. $S(n)$ is $\mathcal{O}(T(n))$.

   D. We can not say anything about the space complexity based on this information.

3. (1 point) Consider a dynamic array, whose capacity C is incremented by inc whenever the array is full (new capacity is $C + inc$). Which of the following values of inc **does not** lead to amortized time complexity $\mathcal{O}(n)$ for n push operations (insertions at the end)? Consider that the initial capacity is given by C_0, where $1 \leq C_0 \ll n$ (C_0 is a positive integer, much smaller than n).

   A. $inc = C$

   B. $inc = \lceil C/2 \rceil$

   C. $inc = C + 1$

   D. $inc = C_0$

4. (1 point) Consider the most time-efficient algorithm to find the last but one node in a list (just before the tail). Which of these list implementations **does not** enable $O(1)$ time complexity for this operation?

   A. Array list

   B. Circularly-linked list

   C. Doubly-linked list

   D. Doubly circularly-linked list

5. (1 point) Consider the most time-efficient algorithm for finding an element in a sequence S. On which of the following data structures storing S **is it impossible** to find the element in $O(\log n)$ time?

   A. Sorted array

   B. Sorted singly-linked list

   C. AVL tree

   D. Red-black tree

6. (1 point) Consider the implementation of class `MyNestedClass` below.

```
public class MyClass<E> {
  private int size;
  private E[] list;
  /* ... */

  private class MyNestedClass implements MyInterface<E> {
    private int j = 0;
    private boolean removable = false;

    public boolean hasNext() {
      return j < size;
    }

    public E next() throws NoSuchElementException {
      if (j == size) throw new NoSuchElementException("No next element");
      removable = true;
      return list[j++];
    }

    public void remove() throws IllegalStateException {
      if (!removable) throw new IllegalStateException("Nothing to remove");
      this.remove(j-1);
      j--;
      removable = false;
    }
  }
}
```

Which type of data structure does class `MyNestedClass` above implement?

A. Iterator.
B. Positional list.
C. Array list.
D. Singly-linked list.

7. (1 point) Consider the same class `MyNestedClass` from question 6. Which property and underlying data structure are correct for class `MyNestedClass`?

A. Lazy, on an array-based list.
B. Snapshot, on an array-based list.
C. Lazy, on a linked list.
D. Snapshot, on a linked list.

8. (1 point) Consider three implementations of a priority queue ADT respectively using: an unsorted list, a sorted list, and a heap. For each implementation, which operation(s) consider(s) the values of the keys?

A. **unsorted list:** insert **sorted list:** removeMin **heap:** both
B. **unsorted list:** removeMin **sorted list:** insert **heap:** both
C. **unsorted list:** removeMin **sorted list:** insert **heap:** removeMin
D. **unsorted list:** insert **sorted list:** removeMin **heap:** insert

9. (1 point) Which of the following statements about comparison-based sorting algorithms is **true**?

A. Selection sort is faster than insertion sort for all inputs.
B. Quick sort is always the fastest amongst comparison-based sorting algorithms.
C. Heap sort cannot be implemented in-place.
D. Quick sort and merge sort follow the divide-and-conquer paradigm.

10. (1 point) What would be the fastest algorithm to sort a sequence S of 50 nearly sorted integers, with only 2 elements out of place?

    A. Selection sort

    B. Insertion sort

    C. Merge sort

    D. Quick sort

11. (1 point) If we want to use a radix sort algorithm to sort 32-bit integers taking contiguous sets of 8 bits as elementary keys, how many times do we need to apply the underlying bucket sort algorithm and what property does this algorithm need to have? Select the correct statement.

    A. 8 times, in-place

    B. 8 times, stable

    C. 4 times, in-place

    D. 4 times, stable

12. (1 point) Consider that you are grading the final exam of Algorithms and Data Structures, and would like to use a data structure to store pairs (s, g), where s is the student number and g is the corresponding grade. While grading you take exams from the pile in any order, and you would like to access and eventually update or replace the grade of each student multiple times. The **number** of students is known, but the **range** of student numbers is not fixed. What data structure would provide the **best expected time complexity** for a large number of accesses and updates?

    A. AVL tree

    B. Hash table

    C. Array-based list

    D. Linked list

13. (1 point) Which of the following properties is **not** important for a hash function?

    A. The hash function should be easy to compute.

    B. Objects with different values should have a different hash code with high probability.

    C. If `a.hashCode() == b.hashCode()` is true, then `b.equals(a)` should be true.

    D. If `b.equals(a)` is true, then `a.hashCode() == b.hashCode()` should be true.

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

0	1	2	3	4	5	6	7	8	9	10	11
		26	15	40	5		19				23

    The hash function is $h(k) = k \mod 12$. Assume we first remove the entry with key 40 and then insert an entry with key 14. In which bucket will the inserted entry end up?

    A. 1

    B. 2

    C. 4

    D. 6

15. (1 point) Consider Kruskal's algorithm operating on a graph $G = (V, E)$ where $n = |V|$ is the number of vertices and $m = |E|$ is the number of edges. Which of the following statements is **true**?

    A. The largest edge in E is never added to the tree by the algorithm.

    B. Kruskal's algorithm can be most efficiently implemented using an unsorted list to store edges.

    C. Kruskal's algorithm starts with n clusters, and terminating it after k iterations results in a set of $n - k$ clusters.

    D. The tighest bound on n is $\mathcal{O}(m^2)$, and we can denote the tighest complexity bound of the algorithm as $\mathcal{O}(m \log m)$ when using an efficient version of the union-find data structure.

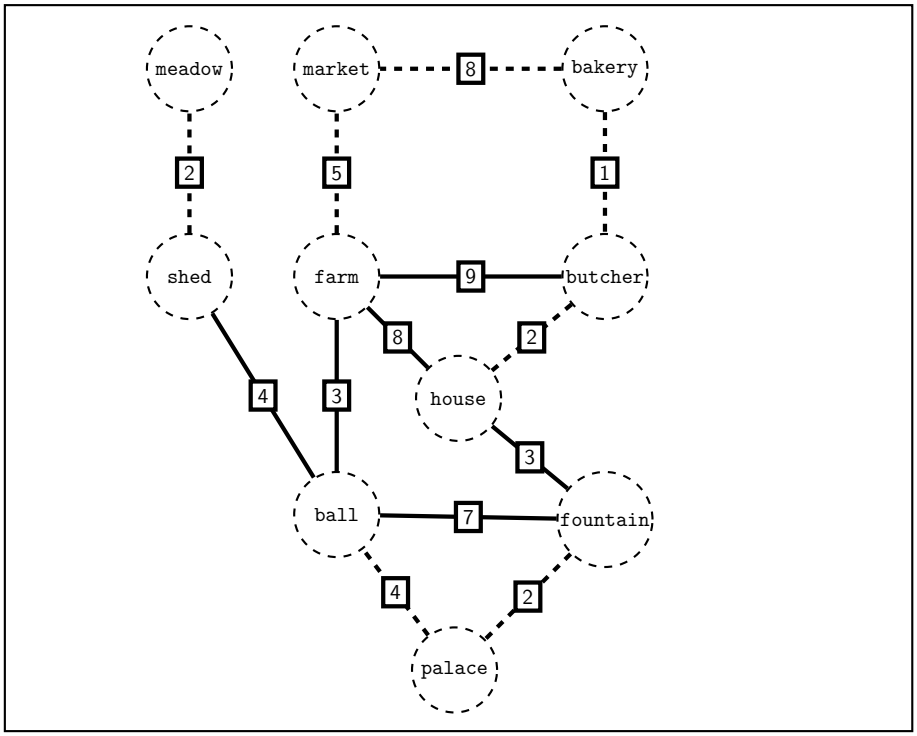


Figure 1: Graph for multiple choice questions 16 and 17.

16. (1 point) To get to the ball, Cinderella studies a map which takes the form of the graph depicted in Figure 1. Which of the following claims about this graph $G = (V, E)$ is **true**?
Note that for this question you can ignore the weights of the edges.

    A. $\sum_{v \in V} \deg(v) = 2|E| - 1$

    B. Removing the dashed edges (only the edges, we keep the vertices), results in a graph G' consisting of 5 connected components.

    C. If we add a single edge to the set of dashed edges and vertices, the resulting set (of dashed edges and vertices) forms a spanning tree of G.

    D. It is possible to make a simple cycle through all vertices of G.

17. (1 point) Consider again the graph in Figure 1. Cinderella uses Dijkstra's algorithm to compute the shortest route from her house (vertex `house`) to the ball (vertex `ball`). Which of the following claims about the execution of Dijkstra's algorithm is **true**?

    A. The shortest distance to vertex `ball` is 9 directly after adding `fountain` to explored vertices.

    B. We can terminate the algorithm as soon as `ball` is added to the set of explored vertices.

    C. Vertex `farm` is added to the set of explored vertices after both the `market` and `bakery` vertices.

    D. The first vertex added by the algorithm to the set of explored vertices is `fountain`.

18. (1 point) What would be the most efficient algorithm to find all vertices at a fixed distance d from a specific vertex v in a graph, where $d \geq 1$?

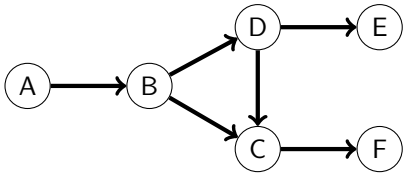
    A. Topological sorting

    B. Depth-first traversal

    C. Breadth-first traversal

    D. Kruskal's algorithm

19. (1 point) Consider the following directed acyclic graph (DAG):



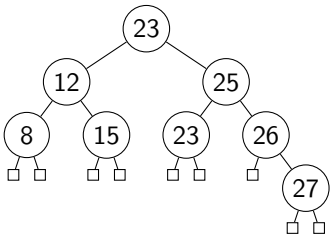
In general, DAGs can have multiple topological orders. How many different topological orders does the above graph have?

A. 2
B. 3
C. 4
D. 5

20. (1 point) Assume we have a graph with 1000 vertices and 2000 edges. Which graph data structure is most suitable for storing this graph if we want to use the least amount of space possible in the absolute (non-asymptotic) sense?

A. Edge list
B. Adjacency list
C. Adjacency map
D. Adjacency matrix

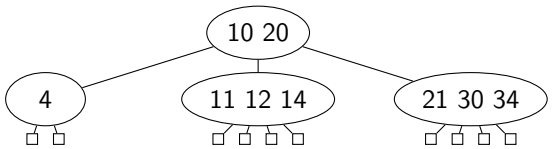
21. (1 point) Consider the following tree:

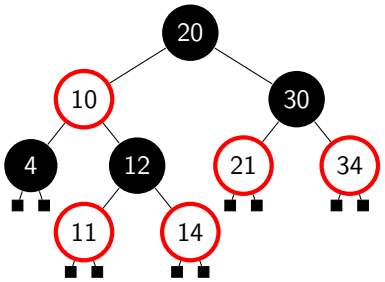


Which of the following is **true**?

A. The tree is an AVL tree.
B. The tree is a minimum-oriented heap.
C. The tree is a maximum-oriented heap.
D. The tree is a binary tree, but neither a heap nor a binary search tree.

22. (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.

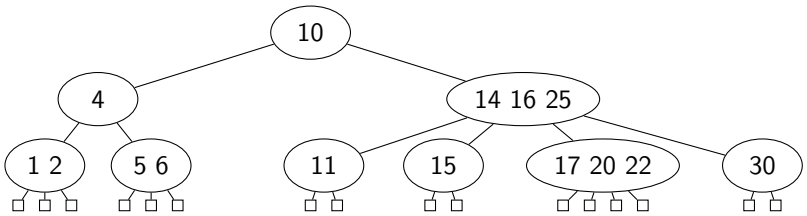




Which of the following statements is **true**?

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

23. (1 point) Consider the following (2,4) tree:



When inserting **21** into the tree, how many overflows are caused in the tree?

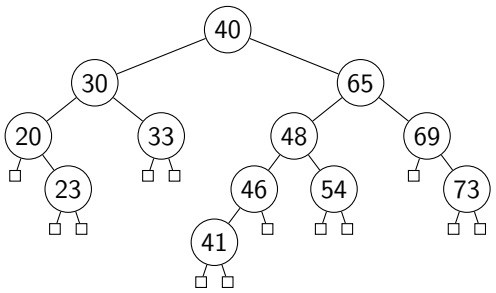
A. No overflow.

B. One overflow.

C. Two overflows.

D. Three overflows.

24. (1 point) Consider the following AVL tree:



When deleting **30** 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. None.

B. One tri-node restructuring.

C. Two tri-node restructurings.

D. Three tri-node restructurings.

Open questions (35%, 29 points)

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

```
public static void sort(int[] data) {
  sortHelper(data, 0, data.length - 1);
}

public static void sortHelper(int[] data, int low, int high) {
  int i = low, j = high;
  int y = data[low];

  while (i <= j) {
    while (data[i] < y) i++;
    while (data[j] > y) j--;
    if (i <= j) {
      swap(data, i, j);
      i++;
      j--;
    }
  }

  if (low < j)  sortHelper(data, low, j);
  if (i < high) sortHelper(data, i, high);
}

public static void swap(int[] data, int i, int j) {
  int x = data[i];
  data[i] = data[j];
  data[j] = x;
}
```

The base and recurrence equation for the worst-case time complexity of method `sortHelper` are:

$$T(0) = c_1 \qquad T(n) = T(n-1) + nc_2 + c_3.$$

Here, we let $n = \texttt{high} - \texttt{low} + 1$.

(a) (4 points) Explain why these equations correctly describe the worst-case **time** complexity. Refer to the relevant parts of the code to justify your answer.

(b) (6 points) Derive the closed form of the given recurrence equation. You should either:

- guess the closed form and prove correctness of your solution by induction, or
- derive the closed form solution by repeatedly unfolding the recurrence equation.

(c) (4 points) State the base and recurrence equation for the **space** complexity of method `sortHelper` in terms of n. Refer to the relevant parts of the code to justify your answer.

26. Prove that $x^3 + 5 + x$ is $\Omega(x^2)$.

(a) (2 points) State in detail the mathematical conditions that should be proved.

(b) (2 points) Prove that these conditions hold. Explain all the steps in your proof.

27. The following Java code implements an algorithm to check whether two arrays a and b are permutations of each other.

```
public static boolean isPermutation(int[] a, int[] b) {                    1
  int j, temp;                                                             2
  if (a == null || b == null) return false;                                3
  if (a.length != b.length) return false;                                  4
                                                                           5
  for (int i = 0; i < a.length; i++) {                                     6
    for (j = i; j < b.length; j++) {                                       7
      if (a[i] == b[j]) {                                                  8
        temp = b[i];                                                       9
        b[i] = b[j];                                                       10
        b[j] = temp;                                                       11
        break;                                                             12
      }                                                                    13
    }                                                                      14
    if (j == b.length) {                                                   15
      return false;                                                        16
    }                                                                      17
  }                                                                        18
  return true;                                                             19
}                                                                          20
```

(a) (5 points) Write the polynomial expressing the worst-case **time** complexity of method `isPermutation` as a function of n, where n is the number of elements in each array (a and b have the same length). Define all variables and constants. Explain each term of the polynomial by referring to the lines of code.

(b) (4 points) Simplify your polynomial expression as much as possible. Derive and explain the tightest worst-case **time** complexity in Big-Oh notation.

(c) (2 points) Describe a faster solution to check whether an array is a permutation of another and state its tightest worst-case **time** complexity.

Implementation questions (30%)

There are three implementation questions on Weblab.

End of the exam