Delft University of Technology
EEMCS Faculty

CSE1305 Algorithms & Data Structures

Final Written Exam

25 January 2022, 09:00–11:00

Examiners:

Examiner responsible: Joana Gonçalves and Ivo van Kreveld
Examination reviewer: Stefan Hugtenburg

Parts of the examination and determination of the grade:

Exam part	Number of questions	Question specifics	Grade (%)	Grade (points)
Multiple-choice	18 questions (equal weights)	One correct answer per question	50%	$5 \cdot \frac{\text{score}}{18}$
Open questions	3 questions (different weights)	Multiple parts	50%	$5 \cdot \frac{\text{score}}{25}$

Use of information sources and aids:

- A hand-written double-sided A4 cheat sheet can be used during the exam.
- No other materials may be used, including but not limited to books, lecture slides in any form, or devices such as laptops and phones.
- Scrap paper sheets are provided at the beginning of the exam. Additional scrap paper can be requested.

General instructions:

- Solve the exam on your own. Any form of collaboration is prohibited.
- You cannot leave the examination room during the first 30 minutes.
- If you are eligible for extra time, place the "Verklaring Tentamentijd Verlenging" on your desk, together with your student card.

Instructions for writing down your answers:

- You should answer the questions using the provided answer sheets.
- Write your name and student number on every sheet of paper.
- **For multiple-choice questions**, mark the answers on this exam paper first, and 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.

Multiple-choice questions (50%, 18 points)

1. (1 point) Which of the following statements about asymptotic algorithmic complexities is **true**?
   A. There exists a function $f(n)$ which is $\mathcal{O}(n)$ and $\Omega(n)$, but not $\Theta(n)$.
   B. There exists a function $f(n)$ which is $\mathcal{O}(n)$ and $\Theta(n)$, but not $\Omega(n)$.
   C. There exists a function $f(n)$ which is $\Omega(n)$ and $\Theta(n)$, but not $\mathcal{O}(n)$.
   D. None of the above is true.

2. (1 point) Consider a recursive method that makes two recursive calls whose input size is 1 lower than the input size with which the method was called. Which of the following statements about this recursive method is **true**?
   A. Both the time and space complexity can be $\mathcal{O}(n)$.
   B. The time complexity can be $\mathcal{O}(n)$, but the space complexity cannot.
   C. The space complexity can be $\mathcal{O}(n)$, but the time complexity cannot.
   D. Both the time and space complexity cannot be $\mathcal{O}(n)$.

3. (1 point) The Java method `rearrange` below rearranges the elements of a linked list with `head` node of type `Node`, also defined below.

```
1  public class ADSList {
2    private static class Node {
3      private int value;
4      private Node next;
5      /* ... */
6    }
7
8    public static void rearrange(Node head) {
9      Node p, q;
10     int temp;
11     if (head == null || head.getNext() == null) {
12       return;
13     }
14     p = head;
15     q = head.next;
16     while (q != null) {
17       temp = p.value;
18       p.value = q.value;
19       q.value = temp;
20       p = q.next;
21       if (p != null)
22         q = p.next;
23       else
24         q = null;
25     }
26   }
27 }
```

   Method `rearrange` is called with a list containing the following integer values [0, 2, 4, 6, 8, 10, 12] in the given order (left/head to right/tail). What will be the contents of the list after the call to method `rearrange` completes its execution?
   A. head [2, 0, 6, 4, 10, 8, 12] tail
   B. head [4, 2, 0, 10, 8, 6, 12] tail
   C. head [0, 4, 2, 8, 6, 12, 10] tail
   D. head [4, 0, 2, 10, 6, 8, 12] tail

4. (1 point) Consider the most time-efficient algorithm to append all m elements of a sequence $S2$ one by one to the end of another sequence $S1$ already containing n elements, without removing the elements from sequence $S2$. What is the tightest worst-case time complexity of such an algorithm when sequences $S1$ and $S2$ are implemented using a dynamic array that increments the size by one with every resizing? You can assume that $n > m$.

   A. $\mathcal{O}(nm)$
   B. $\mathcal{O}(n)$
   C. $\mathcal{O}(m^2)$
   D. $\mathcal{O}(m)$

5. (1 point) Consider the following Java method `recursiveMethod` below.

```
1 private static void recursiveMethod(int[] array, int low, int high) {
2   if (high < low)
3     return;
4
5   int temp = array[low];
6   array[low] = array[high];
7   array[high ] = temp;
8   recursiveMethod(array, low + 1, high - 1);
9 }
```

   What is the tightest worst-case **space** complexity of calling the method for a given array of integer elements `arr`, that is, executing the call `recursiveMethod(arr, 0, arr.length-1)`? Use variable n to denote the length of the input array.

   A. $\mathcal{O}(\log n)$
   B. $\mathcal{O}(n)$
   C. $\mathcal{O}(n \log n)$
   D. $\mathcal{O}(n^2)$

6. (1 point) Consider that a queue is implemented using a singly-linked list with both a reference to the head and a reference to the tail of the list (without dummy header and trailer nodes). Both enqueueing and dequeueing operations are implemented such that they take $\mathcal{O}(1)$ time. Which of these references will change when enqueueing an element into either an empty or a non-empty queue?

   A. **empty:** head **non-empty:** head
   B. **empty:** tail **non-empty:** tail
   C. **empty:** both **non-empty:** head
   D. **empty:** both **non-empty:** tail

7. (1 point) Consider a complete binary tree whose inorder traversal is 6, 5, 1, 4, 3, 2, 10, 7. What is the parent of node 1?

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

8. (1 point) Consider that the **heapify** algorithm is applied to build a heap of height h from the elements of a given array. What is the tightest upper bound on the total number of swaps in the calls made by heapify for the nodes at the third last level (or antepenultimate) of the complete binary tree represented by the array? You can assume that the antepenultimate level exists.

   A. $2^{h-1} - 1$
   B. $2^{h-1} - 2$
   C. $2^h - 1$
   D. $2^h - 2$

9. (1 point) Consider the Java method `specialTraversal` below, which performs a special traversal of a binary tree rooted at the given `node`.

```
1  public class BinaryTree {
2    private static class Node {
3      int value;
4      Node left, right;
5      public Node(int v) { value = v; }
6    }
7
8    public void specialTraversal(Node node) {
9      DS1<Node> ds1 = new DS1<>();
10     DS2<Node> ds2 = new DS2<>();
11     ds1.insert(node);
12
13     while (!ds1.isEmpty()) {
14       node = ds1.peek();
15       ds1.remove();
16       ds2.insert(node);
17
18       if (node.right != null)
19         ds1.insert(node.right);
20       if (node.left != null)
21         ds1.insert(node.left);
22     }
23
24     while (!ds2.isEmpty()) {
25       node = ds2.peek();
26       System.out.print(node.data + " ");
27       ds2.remove();
28     }
29   }
30 }
```

For a given complete binary tree t, we know the following: the breadth-first traversal of t visits the nodes of t in the following order [1, 2, 3, 4, 5, 6, 7], and the `specialTraversal` method prints out the nodes of t in the following order [4, 5, 6, 7, 2, 3, 1]. Knowing this, what kind of data structures are `DS1` and `DS2` ? You can assume that `specialTraversal` is called using the root of the binary tree as argument.

A. **DS1:** stack **DS2:** queue

B. **DS1:** stack **DS2:** stack

C. **DS1:** queue **DS2:** queue

D. **DS1:** queue **DS2:** stack

10. (1 point) Consider sorting the elements of the following array [1, 5, 7, 6, 4, 3, 2] in increasing order using the in-place insertion sort algorithm. How many elements are shifted and how many comparisons are made during the complete execution of the inner loop for **only the one** iteration of the outer loop that processes the element at index 4?

A. 3 shifts and 3 comparisons

B. 3 shifts and 4 comparisons

C. 4 shifts and 3 comparisons

D. 4 shifts and 4 comparisons

11. (1 point) Consider sorting the elements of the following array [0, 4, 3, 6, 5, 2, 1] in increasing order using the in-place quick sort algorithm that always chooses the pivot as the last element in the subarray to be sorted. How many swaps are performed in total by the algorithm when sorting this specific array?
    A. 3
    B. 4
    C. 5
    D. 6

12. (1 point) Which of the following statements about sorting and selection algorithms is **true**?
    A. The tightest worst-case big-Oh **space** complexity of the merge sort, in-place quick sort, and in-place quick select algorithms is $\mathcal{O}(n)$.
    B. Radix sorts, merge sort, in-place heap sort, and in-place insertion sort are all stable sorting algorithms.
    C. For finding the median of an unsorted array, sorting the array using in-place heap sort followed by accessing the middle element has lower worst-case big-Oh time complexity than applying quick select.
    D. Key-based sorting is always faster than comparison-based sorting.

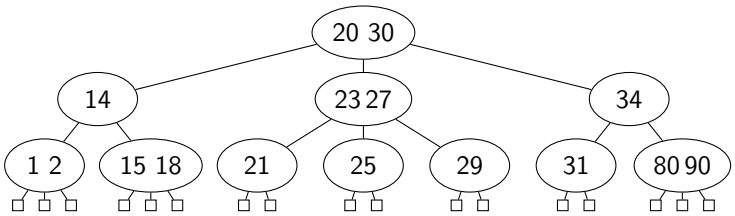
13. (1 point) Consider the following fixed size hash table using linear probing (associated values are omitted):

0	1	2	3	4	5	6	7	8	9
9		11					36	16	18

    The hash function is $h(k) = (k + 1) \mod 10$. Assume we first delete the entry with key 11, then insert an entry with key 19, and then delete an entry with key 17. How many buckets do we need to search before concluding that 17 is not in the hash table?
    A. 1
    B. 4
    C. 5
    D. 6

14. (1 point) Maps can be implemented in multiple ways. Consider a map implemented as a sorted list, a map implemented as a hash table, and a map implemented as a balanced search tree. For which of the following operations does the map implemented as a balanced search tree have the lowest expected time complexity?
    A. Removing the element with a given key.
    B. Removing the element with the smallest key.
    C. Removing the element with the greatest key which is smaller than a given value.
    D. Removing all elements with a given set of k keys.

15. (1 point) Consider the following (2,4) tree:



    When deleting **30** from the tree, what happens with the middle child of the root currently containing (23,27)? If the node of the entry to be deleted has non-null children, the node will be replaced with the in-order predecessor (i.e. the maximal entry in the left child of the entry).
    A. Node (23 27) will be fused with its right sibling.
    B. Entry 23 will be moved to a child for a fusion operation.
    C. Entry 27 will be moved to a child for a fusion operation.
    D. Nothing happens, node (23 27) will stay as it is.

16. (1 point) Given a red-black tree of height 10, what is the minimum depth that each leaf needs to have? Note that height and depth do **not** mean black height and black depth here.

    A. 1

    B. 5

    C. 9

    D. 10

17. (1 point) Consider that we want to find a shortest path in a graph were all edges have a weight of 1. Which of the following statements is **true**?

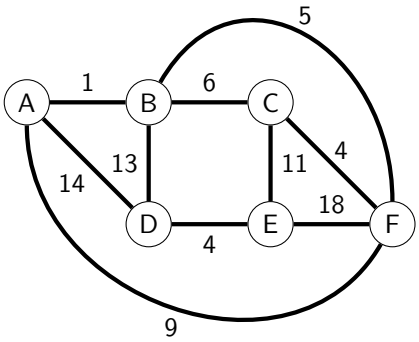
    A. Dijkstra's algorithm might not return the correct result.

    B. There exists an algorithm faster than Dijkstra's to find a shortest path in such a graph.

    C. Although we could use another algorithm to find a shortest path in such a graph, Dijkstra's algorithm has the same time complexity as that alternative algorithm.

    D. Dijkstra's algorithm is the fastest algorithm to find a shortest path in all types of graphs.

18. (1 point) Consider the following weighted graph:



    If we apply Kruskal's algorithm to find the minimum spanning tree in the above graph, what is the last edge that will be added to the minimum spanning tree?

    A. (A, F)

    B. (B, C)

    C. (C, E)

    D. (E, F)

Open questions (50%, 25 points)

19. Prove that $4 \cdot n^2 - 20$ is $\mathcal{O}(n^2)$.
    (a) (2 points) State in detail the mathematical conditions that should be proven.
    (b) (2 points) Prove that these conditions hold. Explain all the steps in your proof.
    (c) (4 points) Prove that all functions which are $\mathcal{O}(4 \cdot n^2 - 20)$ are also $\mathcal{O}(n^2)$ i.e. prove that for an arbitrary function $f(n)$, if $f(n)$ is $\mathcal{O}(4 \cdot n^2 - 20)$, then $f(n)$ is $\mathcal{O}(n^2)$.

20. Consider the following Java implementation of method `methodX`.

```
1  public static <V,E> PositionalList<Vertex<V>> methodX(Graph<V,E> g) {
2      PositionalList<Vertex<V>> ret = new LinkedPositionalList<>( );
3      Stack<Vertex<V>> ready = new LinkedStack<>( );
4      Map<Vertex<V>, Integer> inCount = new ProbeHashMap<>( );
5      for (Vertex<V> u : g.vertices()) {
6          inCount.put(u, g.inDegree(u));
7          if (inCount.get(u) == 0)
8              ready.push(u);
9      }
10     while (!ready.isEmpty( )) {
11         Vertex<V> u = ready.pop( );
12         ret.addLast(u);
13         for (Edge<E> e : g.outgoingEdges(u)) {
14             Vertex<V> v = g.opposite(u, e);
15             inCount.put(v, inCount.get(v) - 1);
16             if (inCount.get(v) == 0)
17                 ready.push(v);
18         }
19     }
20     return ret;
21 }
```

    (a) (2 points) Explain what the algorithm `methodX` does. In other words, what can you tell about the output of the method given the input?
    (b) (6 points) Assume that graph `g` does not have parallel edges or self-loops. Graph `g` could be implement in multiple ways. Consider the use of these 3 possible data structures:
        - An edge list
        - An adjacency map
        - An adjacency matrix

        Which data structure is optimal for this method concerning worst-case time complexity? Give the polynomial expressing the tightest worst-case **time** complexity of `methodX` as a function of the number of vertices n and the number of edges m of graph `g` when the optimal data structure is used. Define all variables and constants, and explain which lines of code contribute to each term of the polynomial.
    (c) (4 points) Describe what would change in this analysis for each of the data structures that are not optimal. Note that your answer should contain the following:
        - The polynomial expression of the first data structure that is not optimal.
        - The operation (and line of code) that causes the difference in the polynomial obtained for the first non-optimal data structure compared to the optimal one in your answer to question 20b. Explain why this difference occurs.
        - The polynomial expression of the second data structure that is not optimal.
        - The operation (and line of code) that causes the difference in the polynomial obtained for the second non-optimal data structure compared to the optimal one in your answer to question 20b. Explain why this difference occurs.

21. (5 points) Derive the closed form of the following recurrence equation:

$$T(1) = c_0$$
$$T(n) = c_1 + 2 \cdot T\left(\frac{n}{4}\right) \qquad \text{if } n > 1$$

You may assume that the function is only called with integers that result in all recursive calls being made with a whole number.

To answer this question, you should either:

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

End of the exam