Data Structures and Algorithms 2020/2021

Resit Exam Questions

– Do not turn this page before the official start of the exam! –

First name, Surname: _____

Student ID: _____

Program: Bachelor Data Science and Artificial Intelligence

Course code: KEN1420

Examiner: Jan Niehues and Tom Pepels

Date/time: Wednesday, 30-06-2021 morning

Format: Closed book exam

Allowed aides: Pens, simple (non-programmable) calculator from the DKE-list of allowed calculators.

Instructions to students:

- The exam consists of 6 questions on 15 pages (including cover page).
- **Solve two problems out of A1,A2, and A3, and one problem out of B1 and B2**
- **If you do not follow the directions and solve all A problems, you will get credit only for A1 and A2. Similarly, if you solve all B problems, you will get credit only for B1.**
- Fill in your name and student ID number on each page, including the cover page.
- Answer every question at the reserved space below the questions. If you run out of space, continue on the back side, and if needed, use the extra blank page.
- Ensure that you properly motivate your answers.
- Do not use red pens, and write in a readable way. Answers that cannot be read easily cannot be graded and may therefore lower your grade.
- You are not allowed to have a communication device within your reach, nor to wear or use a watch.
- You have to return all pages of the exam. You are not allowed to take any sheets, even blank, home.
- If you think a question is ambiguous, or even erroneous, and you cannot ask during the exam to clarify this, explain this in detail in the space reserved for the answer to the question.
- If you have not registered for the exam, your answers will not be graded, and thus handled as invalid.
- **Success!**

The following table will be filled by the examiner:

Question:	A1	A2	A3	B1	B2	Total
Maximum points:	20	20	20	20	20	60
Achieved points:						

Question A1. (20 Points)

1. (10 points) Give the tightest possible bounds for each of the following questions concerning space cost in Big-Oh notation in terms of the variable |V| (number of vertices). You MUST choose your answer from the following (not given in any order), each of which could be re-used (could be the answer for more than one of a. – d.):

   $O(|V|^2)$, $O(|V| \log |V|)$, $O(|V|)$, $O(|V|^2 \log |V|)$, $O(|V|^5)$, $O(2|V|)$, $O(|V|^3)$, $O(\log |V|)$, $O(1)$, $O(|V|^4)$, $O(|V|^5)$

   (a) Suppose a graph has no edges. What is the asymptotic space cost of storing the graph as an adjacency list?

   Answer: $O(|V|)$

   (b) Suppose a graph has no edges. What is the asymptotic space cost of storing the graph as an adjacency matrix?

   Answer: $O(|V|^2)$

   (c) Suppose a graph has every possible edge. What is the asymptotic space cost of storing the graph as an adjacency list?

   Answer: $O(|V|^2)$

   (d) Suppose an undirected graph has one node A that is connected to every other node and the graph has no other edges. What is the asymptotic space cost of storing the graph as an adjacency list?

   Answer: $O(|V|)$

(e) Suppose an undirected graph has one node A that is connected to every other node and the graph has no other edges. What is the asymptotic space cost of storing the graph as an adjacency matrix?

Answer: $O(|V|^2)$

(f) Is an adjacency list faster or slower than an adjacency matrix for answering queries of the form, is edge (u; v) in the graph"?

Answer: slower

(g) Is an adjacency list faster or slower than an adjacency matrix for answering queries of the form: “are there any directed edges with u as the source node?”

Answer: faster

2. Sorting (5 points)
   a) Illustrate the operation of mergesort by drawing the sorting trees. Sort the provided input
      Input: 9 8 6 7 5 0

b) Illustrate the operation of quicksort by drawing the sorting trees. Always choose the first element as a pivot. Sort the provided input
   Input: 9 8 6 7 5 0

3. Trie (5 points)
   You are given the following set of strings { aeef, ad bbfe, c, bbfg }. First, construct a trie. In the second step, compress the trie.

Question A2. (20 Points)

1. (10 points) Give the tightest possible upper bound for the worst-case runtime for each of the following questions in Big-Oh notation in terms of the variable n. You MUST choose your answer from the following (not given in any order), each of which could be re-used (could be the answer for more than one of a. – d.):

$O(n^2)$, $O(n \log n)$, $O(n)$, $O(n^2 \log n)$, $O(n^5)$, $O(2n)$, $O(n^3)$, $O(\log n)$, $O(1)$, $O(n^4)$, $O(n^5)$, $O(n^n)$

a) Pushing a value onto a stack containing n values, implemented as a linked list.

Answer: O(1)

b) Enqueue a value onto a queue containing n values implemented as a circular array (as described in class). (Assume the array is size n+5.)

Answer: O(1)

c) Deleting the minimum value in a binary min heap of size n.

Answer: O (log n)

d) Given a binary search tree containing n integers, create an AVL tree containing the same values. You should not destroy the original BST in the process.

Answer: O(n log n)

e) Printing out the values stored in all of the leaves of a perfect BST containing n values in ascending order.

Answer: O(n)

f) Given an AVL tree containing n positive integers, print out all the even values contained in the tree in descending order (e.g. 12, 8, 6, 2). Be sure to explain how you will get descending order

Answer: O(n)

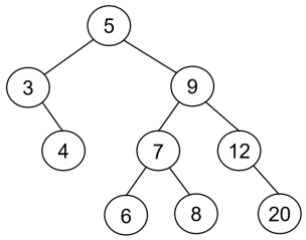
2. Insertion sort (5 points)

Complete this Java method so that it properly implements insertion sort. Make sure the result is in-place (do not use another array).

```
void selectionSort(int [] array) {
    for(int i=0; i < array.length; i++) {
       int index_of_least = i;
       int j;
      for(j=i+1; j < array.length; j++) {
// YOUR CODE HERE #1




} // end of the inner for-loop
// YOUR CODE HERE #2



} // end of the outer for-loop
}
```

3. Binary Tree (5 points)

a) The binary search tree shown below was constructed by inserting a sequence of items into an empty tree.



Which of the following input sequences will produce or not produce this binary search tree? Mark the corresponding cells in the table below for each sequence:

Sequence	Will produce above BST	Will not produce above BST
5 3 4 9 12 7 8 6 20	X	
5 9 3 7 6 8 4 12 20	X	
5 9 7 8 6 12 20 3 4	X	
5 9 7 3 8 12 6 4 20	X	
5 9 3 6 7 8 4 12 20		X

b) Briefly explain what a binary search tree (BST) is, listing its properties.

Binary tree

Internal nods sorted

b) What is the height of a balanced binary tree with n internal nodes?

c) Describe an optimally efficient algorithm to find the largest element that is smaller than a given node n in a BST and explain why it works.

Question A3. (20 Points)

1. (10 Points) Give the tightest possible upper bound for the worst-case runtime for each of the following pseudo code functions in Big-Oh notation in terms of the variable n. You MUST choose your answer from the following (not given in any order), each of which could be re-used (could be the answer for more than one of a. – d.):

   $O(n^2)$, $O(n \log n)$, $O(n)$, $O(n^2 \log n)$, $O(n^5)$, $O(2n)$, $O(n^3)$, $O(\log n)$, $O(1)$, $O(n^4)$, $O(n^5)$,$O(n^n)$

   a.

```
void silly(int n, int x, int y) {
    if (x < y) {
        for (int i = 0; i < n; ++i)
            for (int j = 0; j < n * i; ++j)
                System.out.println(”y = ” + y);
   } else {
        System.out.println(”x = ” + x);
   }
}
```

Answer: O(n^3)

   b.

```
int silly(int n, int m) {
    if (n < 1) return n;
    else if (n < 100)
        return silly(n - m, m);
    else
        return silly(n - 1, m);
}
```

Answer: O(n)

   c.

```
void silly(int n) {
    j = 0;
```

```
    while (j < n) {
        for (int i = 0; i < n; ++i) {
            System.out.println(”j = ” + j);
        }
        j = j + 5;
    }
}
```

Answer: O(n^2)

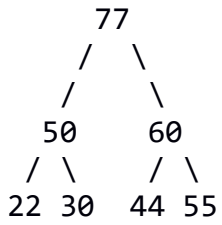
d.

```
void silly(int n) {
    for (int i = 0; i < n * n; ++i) {
        for (int j = 0; j < n; ++j) {
            for (int k = 0; k < i; ++k)
                System.out.println(”k = ” + k);
            for (int m = 0; m < 100; ++m)
                System.out.println(”m = ” + m);
        }
    }
}
```

Answer: O(n^5)

2. Max-Heap (5 points)

   You have the following max-heap:



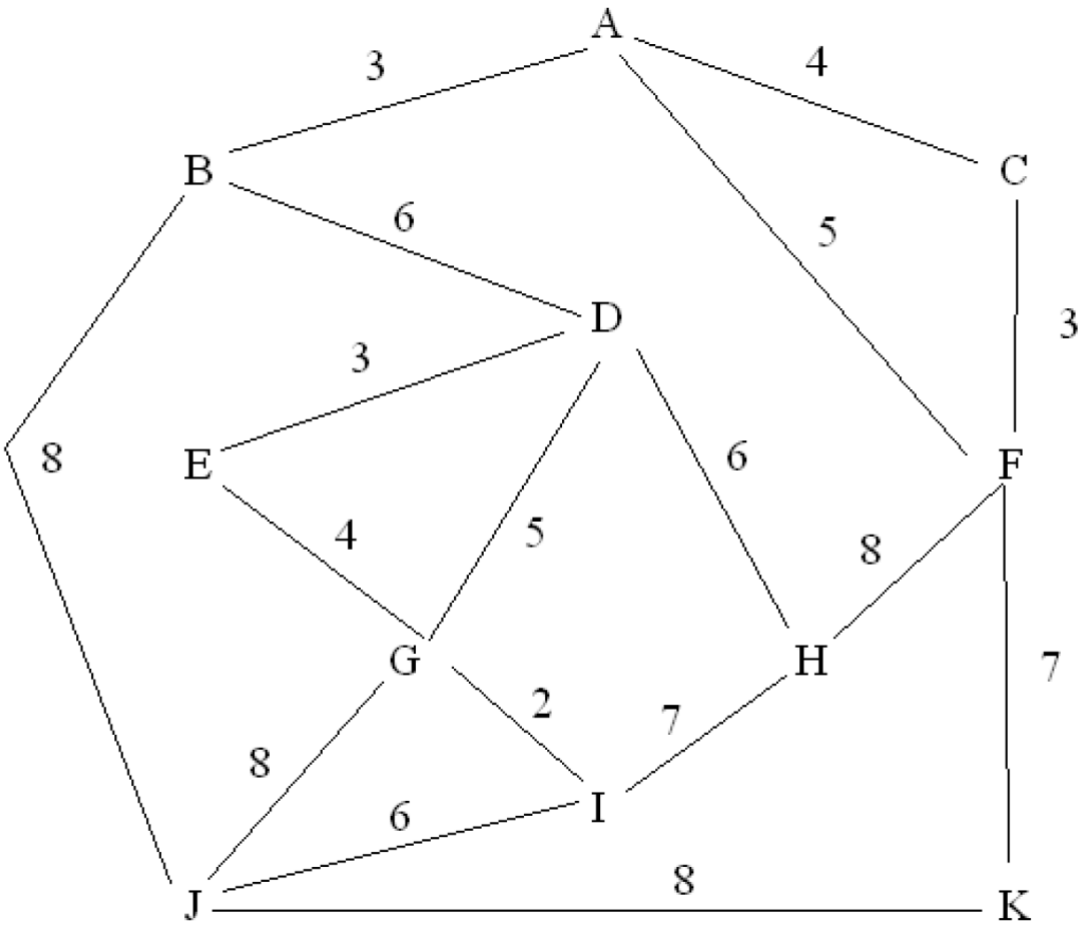
   Insert the following sequence of elements into it and show the heap after each insertion

   55, 51,29

   Then remove the root node from the heap and show the heap after the removal

3. Graphs (5 points)

   Given this graph



a) Perform a depth-first traversal of the graph shown above, starting with vertex A. When expanding the search, choose vertices alphabetically. In the space below, list the vertices in the order in which they are visited.

A B D E G I J K F C H
A B D E G I H F CK J

b) Perform a breadth-first traversal of the graph shown above, starting with vertex A. Select the smallest edge first when appropriate. In the space below, list the vertices in the order in which they are visited.

ABCDJFEGHKI
ABC F D J KH EG I

c) Suppose you are using Dijkstra's algorithm which we discussed in the lecture to find the shortest path from vertex A to vertex I. List, in the order in which they become known, all vertices to which a shortest path is determined in the process of solving this problem, and the length of the shortest path to each of these vertices

For all B questions give your answers in the form of pseudo-code or (simplified) java code. You may additionally explain your answer in English text if you think it makes your solution clear or the question asks for it. Handing in only an English description of your algorithm will result in no points, i.e. you must use pseudo-code or (simplified) java when describing your algorithm. We are not strict in terms of syntax of your code, but the idea must be clearly explained. You may use any data-structure discussed in the course without describing it.

For instance:

```
LinkedList myList <- LinkedList()
myList.addAll(X)
```

is an acceptable description of an algorithm that adds all elements in X to a LinkedList.

```
BST bst  = new BST()
bst.add(1)
bst.add(5)
if bst.search(4)
      print “found”
else
      print “not found”
```

Is an acceptable description of using a binary search tree, add two elements and search for an element even though it is not valid Java or Pseudocode, the idea is sufficiently clearly explained in a formal format.

Question B1. (20 Points)

The longest common prefix of a set of strings S is the longest prefix that all strings in S share. For example:

Example 1:
```
Input: ["flower","flow","flight"]
Output: "fl"
```

Example 2:
```
Input: ["dog","racecar","car"]
Output: ""
```
Explanation: There is no common prefix among the input strings.

Example 3:
```
Input: ["dog","deer","dance"]
Output: "d"
```

A (15 points).
Write, in pseudocode, an efficient algorithm to find the longest common prefix between a given set of strings. The input of the algorithm is a List *L* of length *n* of strings of varying length. The output should be a single string that is the longest prefix that all strings in *L* share. Return an empty string "" if no common prefix exists.

Use a Trie to implement your algorithm. Show how you would modify the insertion algorithm of a Trie and how you would use the modified Trie to determine the longest common prefix.

```
void insert(String s)
{
  for(every char in string s)
  {
    if(child node belonging to current char is null)
    {
      child node=new Node();
    }
    current_node=child_node;
  }
}
```

Your algorithm's runtime should be O(*nm*) or faster, (where n is the length of *L* and *m* is the length of the longest string in the list) assuming a constant alphabet of size 26 (only considering lower-case letters).

B (5 points).
What is the runtime complexity of your algorithm expressed in big-O notation? Give a short explanation (max 3 sentences).

Question B2. (20 Points)

Write an algorithm in pseudocode that, given a stack ***S*** of length ***n***, determines which value in the stack is the smallest and moves that value to the top of the stack while otherwise leaving the remainder of ***S*** in its ***original order***.
See below for example input/outputs for the algorithm.
Note that in the examples, the most right element of the stack is the top element.

Example 1
```
Input:  Stack [4, 2, 8, 3, 6, 5]
Output: Stack [4, 8, 3, 6, 5, 2]
```

Example 2
```
Input:  Stack [10, 23, 54, 23, 12, 58]
Output: Stack [23, 54, 23, 12, 58, 10]
```

The stack class that you have to extend implements only the methods in the interface below. You only have access to the stacks data through these methods.
(Hint: you will probably need more than 1 stack to achieve this task)

```
public interface Stack {
      public boolean isEmpty();

      public void push(double d);

      // throws EmptyStackException if stack is empty
      public double pop();

      // throws EmptyStackException if stack is empty
      public double peek();
}
```

B (5 points):
What is the runtime complexity of your algorithm expressed in big-O notation? Give a short explanation (max 3 sentences).

Extra answer sheet.

Extra answer sheet.