Data Structures and Algorithms 2020/2021

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, 31-03-2021 afternoon

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 upper bound for the worst case running time 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 particular order), each of which could be re-used (could be the answer for more than one of a. – d.):
   $O(n^2)$, $O(n \log n)$, $O(n)$, $O(n^2 \log n)$, $O(n^5)$, $O(2n)$, $O(n^3)$, $O(\log n)$, $O(1)$, $O(n^4)$, $O(nn)$
   ****For any credit, you must explain your answer.**

a.

```
void silly(int n) {
     for (int i = 0; i < n; ++i) {
          j = n;
          while (j > 0) {
               System.out.println("j = " + j);
               j = j - 2;
          }
     }
}
```

Answer: $O(n^2)$

b.

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

Answer: $O(n^3)$

c.

```
void silly(int n) {
     for (int i = 0; i < n; ++i) {
          for (int j = 0; j < n; ++j)
               System.out.println(”j = ” + 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: $O(n^2)$

d.

```
int silly(int n, int m) {
     if (m < 2) return m;
     if (n < 1) return n;
     if (n < 10)
          return silly(n/m, m);
     else
          return silly(n - 1, m);
}
```

Answer: O(n)

2. Sorting (5 points)

a) Illustrate the operation of insertion sort by completing the table below. In successive rows of the table, show the array contents after each pass of the algorithm.

9	**8**	**6**	**7**	**5**	**0**
8	9	6	7	5	0
6	8	9	7	5	0
6	7	8	9	5	0
5	6	7	8	9	0
0	5	6	7	8	9

b) Illustrate the operation of bubble sort by completing the table below. In successive rows of the table, show the array contents after each pass of the algorithm.

9	**8**	**6**	**7**	**5**	**0**
8	6	7	5	0	9
6	7	5	0	8	9
6	5	0	7	8	9
5	0	6	7	8	9
0	5	6	7	8	9

3. Boyer-Moore algorithm (5 points)

   Suppose that you run the Boyer-Moore algorithm (the basic version considered in the textbook and lecture) to search for the pattern

   I D O F T H E

   in the text

   M E N D E R O F R O A D S W I T H T H E A I D O F T H E

   Give the trace of the algorithm in the grid below, circling the characters in the pattern that get compared with the text.

M	E	N	D	E	R	O	F	R	O	A	D	S	W	I	T	H	T	H	E	A	I	D	O	F	T	H	E
I	D	O	F	T	H	(E)																					

Question A2. (20 Points)

1. (10 points) Give the tightest possible upper bound for the worst case running time 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 particular order), each of which could be re-used (could be the answer for more than one of a. – d.):
   $O(n^2)$, $O(n \log n)$, $O(n)$, $O(n^2 \log n)$, $O(n^5)$, $O(2n)$, $O(n^3)$, $O(\log n)$, $O(1)$, $O(n^4)$, $O(nn)$
   ****For any credit, you must explain your answer.** *Assume that the most time-efficient implementation is used.*

   a) Pop a value off a stack containing n elements implemented as an array.

Explanation: Answer: O(1)

   b) Printing out all the odd values stored in a binary search tree containing n positive integers in ascending order.

Explanation: Answer O(n)

   c) Finding the minimum value in a binary search tree of size n.

Explanation: Answer O(n)

   d) Moving the values from a binary min heap, into an initially empty array of the same size. The final contents of the array should be sorted from low to high.

Explanation: Answer O(n log n)

e) Finding the maximum value in a binary min heap of size n.

Explanation: Answer O(n)

f) Printing out the values in an AVL tree in post-order.

Explanation: Answer O(n)

2. Insertion sort (5 points)

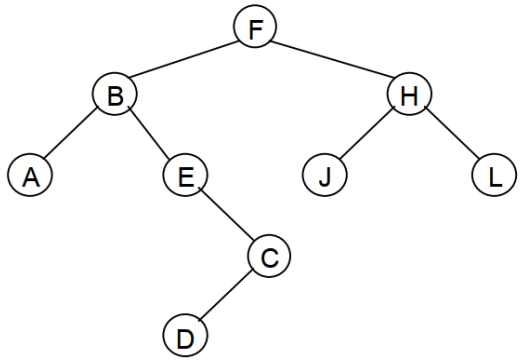
Complete this Java method so that it properly implements insertion sort. Make sure the result is in-place (do not use another array).

```
void insertionSort(int [] array) {
        for(int i=1; i < array.length; i++) {
        // YOUR CODE HERE
        int tmp = array[i];
        int j;
        for(j=i; j > 0 && tmp < array[j-1]; j--) {
        array[j] = array[j-1];}
        array[j] = tmp;


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

3. Binary Tree (5 points)

a) Consider the binary tree shown below. For each of the traversals listed, give the order in which the nodes are visited.



preorder									
inorder									
postorder									
breadth-first									

Pre-order F B A E C D H J L

In-order A B E D C F J H L

Post-order A D C E B J L H F

Breath-first F B H A E J L C D

b) What is the main advantage of an AVL Tree over a Binary Search Tree (BST)?

Deepth is limited always O(log(n))

c) How many internal nodes are there in a balanced binary tree of height $h \geq 0$?

N-1/2

2^h – 1 or 2^(h-1) - 1

d) Describe an optimally efficient algorithm for deleting a node *d* from a BST when neither of *d*'s subtrees is empty. Explain why it works and show that what remains is still a BST.

right and then always left or left and then always right. Put leave at the current postion

property still valide since all elements richt are larger since it is the smallest of the right elements

Question A3. (20 Points)

1. (10 points) Give the tightest possible bounds for each of the following questions concerning runtime in Big-Oh notation in terms of the variable n. You MUST choose your answer from the following (not given in any particular order), each of which could be re-used (could be the answer for more than one of a. – d.):
   $O(n^2)$, $O(n \log n)$, $O(n)$, $O(n^2 \log n)$, $O(n^5)$, $O(2n)$, $O(n^3)$, $O(\log n)$, $O(1)$, $O(n^4)$, $O(nn)$
   ****For any credit, you must explain your answer.** *Assume that the most time-efficient implementation is used.*

   (a) What is the worst-case asymptotic running time of heap-sort?

Explanation: Answer: $O(n \log n)$

   (b) What is the worst-case asymptotic running time of merge-sort?

Explanation: Answer: O(n long n)

   (c) What is the worst-case asymptotic running time of quick-sort?

Explanation: Answer: $O(n^2)$

(d) Can heap-sort be done in-place?

Explanation:

Answer: Yes

(e) Consider one item in an array that is sorted with mergesort. In asymptotic (big-Oh) terms, how many times can that one item move to a different location?

Explanation:

Answer: O(log n)

(f) What is the asymptotic running time of quick-sort if the array is already sorted (or almost sorted) and the pivot-selection strategy picks the leftmost element in the range-to-be-sorted?

Explanation:

Answer: $O(n^2)$

(g) What is the asymptotic running time of quick-sort if the array is already sorted (or almost sorted) and the pivot-selection strategy picks the middle element in the range-to-be-sorted?

Explanation:

Answer: O(n log n)

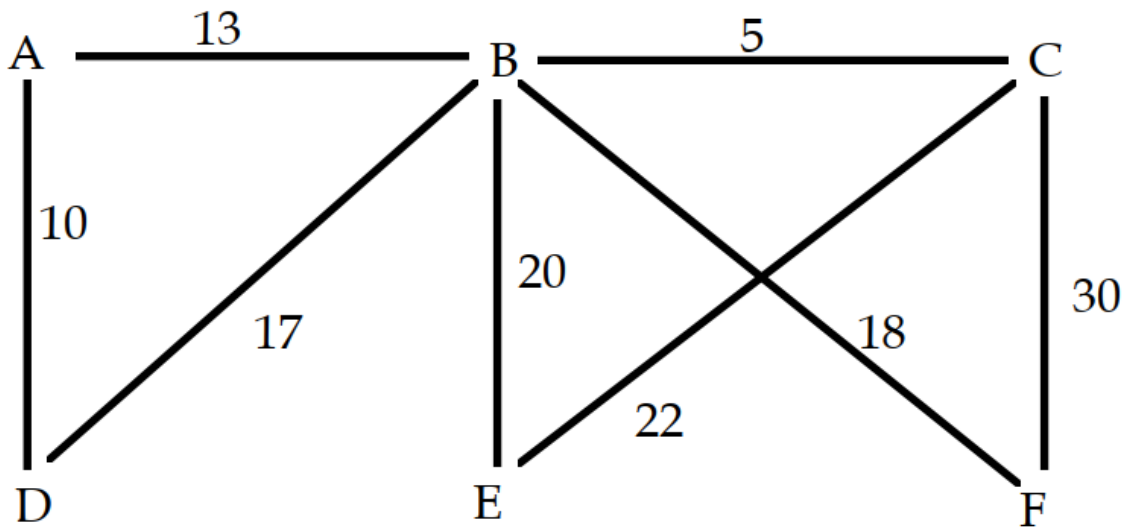
2. Hashing (5 points)

   Consider a hash table with separate chaining with ten hash locations. Using the hash function h(x) = x mod 10, insert the keys {33, 54, 69, 74, 18, 19} (in the order given) into the hash table. Draw the resulting hash table.

0	
1	
2	
3	33
4	54 74
5	
6	
7	
8	18
9	69 - 19

3. Graphs (5 points)

   Given this graph



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

C B A D F E

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

C B E F A D

c) Suppose you are using Dijkstra's algorithm to find the shortest path from vertex D to vertex E. 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

D
D - A 10
D – B 17
D –C 22
D –F 35
D – E 37

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)

Given a large set of words (which may include duplicate words) find the maximum occurring word in it. If two words have the same count, return either one of the two words. Your algorithm may use O(n) extra space and should run in O(n*m) time where n is the number of words, and m the length of the longest word.

Correct solutions to this exercise should use a Trie to store the words and keep a counter at each leaf for the number counted.

Partial grades are given to algorithms that solve the problem with a different time or space complexity.

Question B2. (20 Points)

Given an infinite stream of numbers, describe and algorithm that continuously returns the k^{th} largest element in the stream, where k is a positive integer. The algorithm may return null when $k < n_{streamed}$ i.e. the number of streamed numbers is smaller than the selected k. Explain the complexity of your algorithm.
You receives full points if your algorithm runs in O(log k) time and uses O(k) space. Other solutions will receive partial grades.

```
Input:  Infinite stream of integers

4
5
12
8
9
10
20
42
…

Output:

The k’th largest element is NA.
The k’th largest element is NA.
The k’th largest element is 4
The k’th largest element is 5
The k’th largest element is 8
The k’th largest element is 9
The k’th largest element is 10
The k’th largest element is 12
```

(as some students pointed out, there was a mistake in the question: $k < n_{streamed}$ **should have been** $k > n_{streamed}$

Correct solutions to this question use a balanced binary tree to store the current elements in a balanced binary search tree such as an AVL tree or a Heap.

Partial grades are given to solutions that solve the problem correctly but with different space or time complexity.

Extra answer sheet.

Extra answer sheet.