Durham University

Examination Paper

Examination Session:	Year:	Exam Code:
May/June	2022	COMP1081-WE01

Title: ALGORITHMS AND DATA STRUCTURES

Release Date/Time	10/05/2022 09:30
Latest Submission Date/Time	11/05/2022 09:30
Format of Exam	Take home exam
2 hours	2 hours
Word/Page Limit:	None
Additional Material provided:	None
Expected form of Submission	A SINGLE file: a pdf file (typed or readably handwritten), submitted to Gradescope.
Submission method	Gradescope

Instructions to Candidates:	Answer ALL questions.

Section A Part I
(Prof Matthew Johnson)

Question 1

(a) Consider the Hastings numbers H_n (where n is a non-negative integer) defined by

$$H_n = \begin{cases} n-2 & \text{for } 0 \leq n \leq 3, \\ H_{n-1} + 2H_{n-2} + 2H_{n-3} & \text{for } n \geq 4. \end{cases}$$

i. Calculate H_{10}. **[1 Mark]**

ii. Write pseudocode for a **recursive** function that returns H_n for an integer $n \geq 0$. **[3 Marks]**

(b) Let S and T be two stacks. The following two algorithms, add(x) and extract(), have access to S and T. The input to add(x) is a positive integer.

```
add(x)
  S.push(x)
```

```
extract()
  if S.isEmpty() = True then
    return error
  end if
  while S.isEmpty() = False do
    T.push(S.pop())
  end while
  temp = T.pop()
  while T.isEmpty() = False do
    S.push(T.pop())
  end while
  return temp
```

i. Suppose that S and T are initially empty. What is the output of the final extract if the following sequence is applied:
add(5), add(9), extract(), add(7), add(3), extract(), add(6), extract()
[2 Marks]

ii. Describe how, in general, the two stacks and two algorithms manage data. Justify your answer. **[5 Marks]**

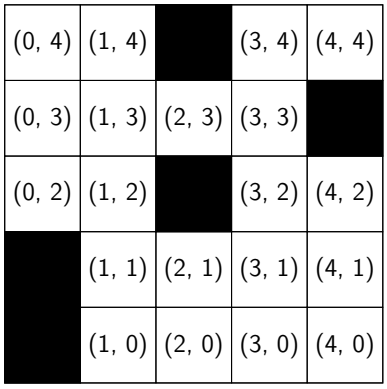
this question is continued on the next page

(c) Here are two floodfill algorithms floodfill_A and floodfill_B. Note that floodfill_B makes two calls to floodfill_A.

```
floodfill_A(x,y)
  if (x,y) is out of range or filled then
    return
  end if
  fill(x,y)
  floodfill_A(x-1,y)
  floodfill_A(x+1,y)
  floodfill_A(x,y-1)
  floodfill_A(x,y+1)
```

```
floodfill_B(x,y)
  if (x,y) is out of range or filled then
    return
  end if
  fill(x,y)
  floodfill_B(x+1,y)
  floodfill_B(x-1,y)
  floodfill_A(x,y+1)
  floodfill_A(x,y-1)
```

The algorithms will be applied to the array of pixels illustrated below. Each unfilled pixel is labelled with its coordinates. The black areas are filled pixels. Pixels not in the illustration are out of range.



i. List the pixels in the order they are filled by floodfill_A(2,1). **[3 Marks]**

ii. List the pixels in the order they are filled by floodfill_B(1,1). **[3 Marks]**

(d) Let L be a linked list that contains an even number of nodes. To **switch** L means to consider L as a sequence of pairs of nodes and to exchange the data in the two adjacent nodes of each pair. For example, if, considering the nodes in order from the head, the data in L is initially 1,2,3,4,5,6, then after a switch the data would be 2,1,4,3,6,5. If initially it was 5,7,9,1,3,5 then after a switch it would be 7,5,1,9,5,3. Write pseudocode for an algorithm that performs a switch of L. **[8 Marks]**

End of Section A **continued**

Section B Part II
(Prof Thomas Erlebach)

Question 2

(a) Consider the MergeSort algorithm. If the recursive MergeSort function is called on the array $[9, 12, 8, 4, 19, 25, 4, 7]$, write down all the resulting recursive calls (including those made within another recursive call) of the MergeSort function. Give each recursive call in the format: "MergeSort(⟨array that is passed to this recursive call⟩)". **[5 Marks]**

(b) An algorithm executes in three phases. In the first phase, it makes $O(\sqrt{n})$ calls to a subroutine, and each of these calls takes $O(n^2)$ time. The second phase takes $o(n^2)$ time. The third phase takes $O(n^2 \log n)$ time. What is the overall time complexity of the algorithm (in big-O notation)? Justify your answer. **[4 Marks]**

(c) Consider functions $f, g : \mathbb{N} \to \mathbb{R}^+$ with $g(n) \geq 2$ for all $n \geq 1$. Is it true that $f(n) = O(g(n))$ implies that $\log_2(f(n)) = O(\log_2(g(n))$? Justify your answer. **[5 Marks]**

(d) Consider the following algorithm, which we call FunSort. If the input array contains at most 3 elements, it sorts those three elements using Insertion-Sort. If the input array contains $n \geq 4$ elements, FunSort computes the value $K = \lceil \frac{3}{4}n \rceil$ (that is, $\frac{3}{4}n$ rounded up to the nearest integer) and then makes three recursive calls:

- Call FunSort on the first K elements of the array.
- Call FunSort on the last K elements of the array.
- Call FunSort (again) on the first K elements of the array.

This completes the description of FunSort.

i. Does FunSort work correctly? In other words, are the elements of the input array guaranteed to be in sorted order when the algorithm terminates? Justify your answer. **[6 Marks]**

this question is continued on the next page

ii. Give a recurrence that captures the time complexity $T(n)$ of FunSort on inputs of size n. It suffices to state the recurrence for the case $n \geq 4$. **[2 Marks]**

iii. Solve the recurrence from part ii. to determine a tight upper bound on the worst-case time complexity of FunSort, using big-Theta (Θ) notation. You are allowed to ignore any ceilings that may appear in your recurrence. In your solution you can use the Master Theorem if you wish (stated below for ease of reference). **[3 Marks]**

Here is a reminder of the statement of the Master Theorem: Suppose a recurrence is of the form $T(n) = aT(n/b) + f(n)$ for $a \geq 1$ and $b > 1$.

- **If** $f(n) = O(n^{\log_b(a)-\epsilon})$ for some constant $\epsilon > 0$ **then** $T(n) = \Theta(n^{\log_b(a)})$.
- **If** $f(n) = \Theta(n^{\log_b(a)} \cdot (\log n)^k)$ with $k \geq 0$ **then** $T(n) = \Theta(n^{\log_b(a)} \cdot (\log n)^{k+1})$.
- **If** $f(n) = \Omega(n^{\log_b(a)+\epsilon})$ for some constant $\epsilon > 0$ **and** if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all n large enough **then** $T(n) = \Theta(f(n))$.

End of Section B **continued**

Section C Part III
(Prof Andrei Krokhin)

Question 3

(a) Consider the Median-of-Medians algorithm for selecting the i-th smallest element in an array, as described in the lectures. Assume that the median of an array with an even number of elements is the right median - in other words, the median of the array with $2n$ elements is the $(n+1)$-th smallest element. Assume that the Median-of-Medians algorithm is called on the following input

$$[6, 25, 1, 15, 9, 13, 8, 2, 16, 11, 4, 3, 30, 7], i = 10.$$

Write down all the resulting recursive calls (including those made within another recursive call) of the Select function. Give each recursive call in the format "Select($\langle$array that is passed to this recursive call$\rangle$, i)". **[5 Marks]**

(b) Consider an arbitrary AVL tree. For each of the following two cases, does one always return to the original AVL tree after performing the operations? Justify your answers.

i. Insert an element which is not in the AVL tree and then delete this element. **[2 Marks]**

ii. Delete an element from the AVL tree and then insert it. **[2 Marks]**

(c) Explain why there is no algorithm running in $o(n)$ time that can find the smallest element in any max-heap with n elements. **[4 Marks]**

(d) Consider all AVL trees with 1000 vertices that contain the first 1000 positive integers $1, 2, 3, \ldots, 1000$.

i. What is the smallest number that can be stored at the root of such an AVL tree?

ii. What is the largest number that can be stored at the root of such an AVL tree?

[12 Marks]

End of Section C **continued**

Section D Part IV
(Dr George Mertzios)

Question 4

(a) Given a graph G, the intermeasure of G is the length of the longest possible shortest path between any two vertices of G. Assuming that G is a tree with n vertices and m edges, design an algorithm that computes the intermeasure of G in $O(n+m)$ time, without using BFS or DFS. Prove its correctness and its running time. **[10 Marks]**

(b) Consider the following variant of the minimum spanning tree problem. The input consists of a connected undirected graph $G = (V, E)$ with positive weights on the edges, together with a subset $S \subseteq V$ of its vertices. The aim is to find the least-weight spanning tree T of G, with the additional property that every vertex of S is a leaf of T. Recall that a leaf of a tree is a vertex of degree 1. We call this the MST-S problem. Note that it is possible that G has no spanning tree with this property.

i. Let H be the subgraph of G that is induced by the vertices of $V \setminus S$, that is, H is obtained by deleting from G the vertices of S and all edges incident with them. Show that the MST-S problem has a solution if and only if H is a connected graph. **[5 Marks]**

ii. Show that, if a solution T to the MST-S problem exists, then for each vertex $v \in S$, the edge incident with v in the tree T has the smallest weight among all edges between v and the vertices of $T \setminus S$. **[2 Marks]**

iii. Describe a polynomial-time algorithm for the MST-S problem (without giving any pseudocode) and prove its correctness. What is its running time on a graph G with n vertices and m edges? Justify your answer. You should aim to make your algorithm as efficient as possible. **[8 Marks]**

END OF PAPER