Durham University

Examination Paper

Examination Session:	Year:	Exam Code:
May/June	2023	COMP1081-WE01

Algorithms and Data Structures

Release Date/Time	17/05/2023 09:30
Latest Submission Date/Time	17/05/2023 12:30
Format of Exam	Restricted window exam
Duration:	2 hours
Word/Page Limit:	
Additional Material provided:	
Expected form of Submission	A SINGLE PDF file submitted to Gradescope
Submission method	Gradescope

Instructions to Candidates:	Answer ALL questions

Section A Fundamental data structures (Dr Eamonn Bell)

Question 1

(a) List 1 is a list of applications. List 2 is a list of data structures that could be used to implement a computer program to support these applications.

List 1

- FRESH: A process for adding fresh goods to a shelf, such that the freshest can be reached first
- AIRLINE: A process to match a unique, two-character airline code to the full name of the airline, which may or may not be unique
- FILES: A process for storing an ordered sequence of fixed-size references to files on disk, which itself may grow or shrink over time
- WAIT: A process for ensuring that access to a finite, shared compute resource is first given to the users who have been waiting for it the longest

List 2

- array
- singly linked list
- stack
- queue
- hash table (with a `dict`-like interface)

Choose three applications from List 1, and for each one of them:

i. Choose one appropriate data structure from List 2 that you could use to efficiently implement the application. Justify your choice of data structure with reference to one advantage of the chosen data structure. **[6 Marks]**

ii. Write up to five lines of pseudocode to show how one operation supported by your chosen data structure can be used to implement a key aspect of the chosen application. **[6 Marks]**

this question is continued on the next page

(b) Consider the Florette numbers F_n (where n is a non-negative integer) defined by

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

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

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

iii. Briefly describe how you can change the implementation of item ii to make it more time-efficient. State one data structure that can be used to implement your change. **[2 Marks]**

(c) The following algorithm takes a string of characters (T) as input.

```
PROCESS(T)
if T.length == 0 then
  return error
else
  stack S
  for c in T do
    if c = '/' then
      S.pop()
    else
      S.push(c)
    end if
  end for
  return S
end if
```

i. What is returned by PROCESS('frei//iendlyskii/es')? **[2 Marks]**

ii. Describe what this algorithm does, in general, and how the algorithm uses sequential data structures to manage data for this purpose. **[2 Marks]**

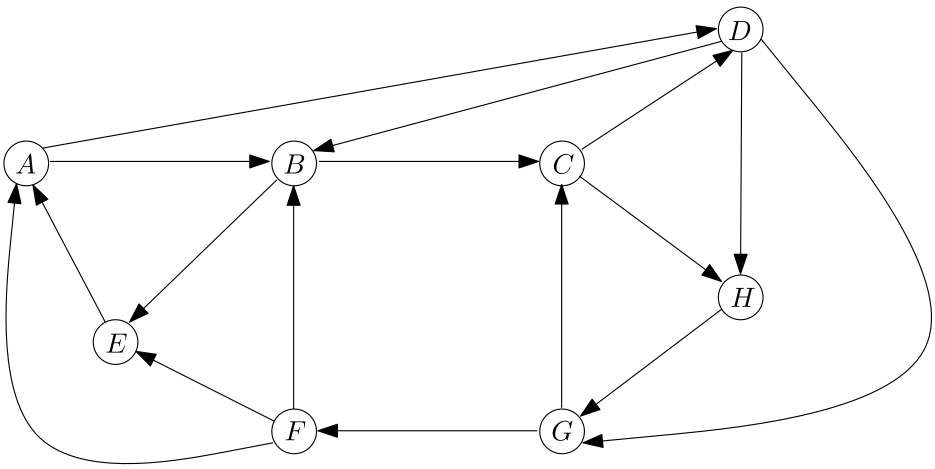
iii. Write pseudocode for a function PRINTOUT(S), where S is a reference to the stack returned by PROCESS(). The function PRINTOUT(S) should print the contents of the stack, character by character, in the reverse order. You should use a stack in the body of your function. **[3 Marks]**

End of Section A

Section B Graph algorithms (Dr George Mertzios)

Question 2

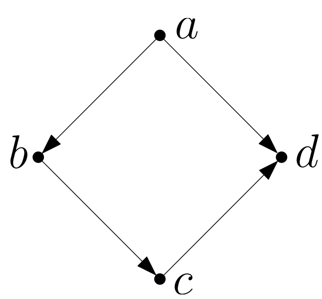
(a) Suppose that a breadth-first search is run on the directed graph below, with vertex A as the source vertex. Classify each edge of the graph as a tree, forward, back, or cross edge.



[6 Marks]

(b) In a connected undirected graph G, we want to find the smallest cycle that contains two specific vertices u and v. Consider the following algorithm: first we find a shortest path P from u to v in G. Then we remove from G the internal vertices of P, and in the remaining graph we search for the shortest path Q from u to v. The algorithm then returns the cycle that consists of the union of the two paths P and Q.
Is this algorithm correct or not? Justify your answer by providing either a proof of correctness, or a counterexample. **[5 Marks]**

(c) i. Consider the following graph with four vertices. Does there exist an execution of the depth-first search algorithm, starting at vertex a, which only contains tree and cross edges? Justify your answer.



[3 Marks]

this question is continued on the next page

ii. Suppose that a directed graph G contains at least one directed cycle. Is it possible that an execution of the depth-first search algorithm classifies every edge either as a tree edge or as a cross edge? Justify your answer. **[5 Marks]**

(d) Let $G = (V, E)$ be an undirected graph such that every edge $e \in E$ has a positive weight w_e. Provide an efficient algorithm that computes a spanning tree of G which has the largest total weight among all spanning trees of G. Justify your answer. **[6 Marks]**

End of Section B

Section C Selection and data structures (Dr Anish Jindal)

Question 3

(a) Consider the problem of selecting the i-th smallest element of an array A using the algorithm QUICKSELECT which uses the partition function that

- always chooses the leftmost element of the subarray $A[\mathit{left} \ldots \mathit{right}]$ as the pivot,
- keeps the elements less than the pivot in the lower partition and reverses their order,
- keeps the elements greater than the pivot in the higher partition in the same order with respect to each other.

The algorithm QUICKSELECT is represented as $QS(A, \mathit{left}, \mathit{right}, i)$ where A is the input array, *left* and *right* are the leftmost and rightmost index-values in the array A, respectively, and i is the index to be found. Show the recursive function and the contents of A at each step after manually running QUICKSELECT with the above rule on the following input:

$$i = 4 \text{ and } A = [13, 9, 7, 5, 2, 4, 12, 8, 10, 15].$$

[10 Marks]

(b) Suppose that the post-order traversal and the in-order traversal of a binary tree give the following output:

In-order:	3	5	12	7	9	11	14	15	17
Post-order:	3	12	5	9	7	14	17	15	11

i. Draw the binary tree. **[2 Marks]**

ii. Replace the root node with a value from the tree to make it a binary search tree. **[1 Mark]**

this question is continued on the next page

(c) Construct an AVL tree T, initially of height 3, with the following properties:

- On inserting a new number a_1 into the initial tree T, a left/left rotation is performed during the fix-up procedure.
- On inserting a new number a_2 into the initial tree T, a right/right rotation is performed during the fix-up procedure.

Show the initial tree T, the values of a_1 and a_2, and any one of the new trees (say T_1 or T_2) after performing the fix-up procedure for either a_1 or a_2. **[6 Marks]**

(d) Consider the following heap represented by the array

$$A = [90, 40, 89, 10, 23, 12, 15, 4, 3, 22, 21, 8]$$

i. Draw the tree for A and state the type of this heap. **[3 Marks]**

ii. Extract the maximum value from this heap and rearrange it with the HEAPIFY procedure. Show the tree at every iteration of HEAPIFY. **[3 Marks]**

Section D Asymptotic notation and sorting (Prof Thomas Erlebach)

Question 4

(a) For each of the following statements for functions f and g from the natural numbers to the natural numbers, state whether it is true or false, and justify your answer.

i. If $f(n)$ is in $\Theta(g(n))$, then $(f(n))^2$ is in $\Theta((g(n))^2)$. **[4 Marks]**

ii. If $f(n)$ is in $\Theta(g(n))$, then $2^{f(n)}$ is in $\Theta(2^{g(n)})$. **[4 Marks]**

(b) Consider the following algorithm RECURSIVEMIN for computing the minimum number in an array. It receives as input an array $A = [a_1, a_2, \ldots, a_n]$ with n elements and two indices *left* and *right* that specify the start and end position of the part of the array in which we want to determine the minimum number. The initial call to find the minimum in the whole array will be RECURSIVEMIN(A,1,n).

```
RecursiveMin(A,left,right)
if right = left then
  return A[left]
end if
mid = ⌊(left + right)/2⌋
m1 = RecursiveMin(A,left,mid)
m2 = RecursiveMin(A,mid+1,right)
if m1 < m2 then
  return m1
else
  return m2
end if
```

Note that *mid* is computed as $(left+right)/2$, rounded down to the nearest integer.

i. Explain briefly why the algorithm RECURSIVEMIN is correct. **[2 Marks]**

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

this question is continued on the next page

iii. Solve the recurrence from part ii to determine a tight upper bound on the worst-case time complexity of RECURSIVEMIN, using the big-Theta (Θ) notation. You are allowed to ignore any floors or 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 constants $a \geq 1$ 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** $T(n) = \Theta(f(n))$.

(c) Consider the MERGESORT algorithm.

i. If we call MERGESORT on an array of length 32, how many calls of the MERGESORT function get executed in total (including the initial call on an array of length 32 and all calls on arrays of length 1)? **[2 Marks]**

ii. We change the MERGESORT function so that it uses SELECTION-SORT (instead of making recursive calls and merging the resulting arrays) whenever it is called with an array of size at most 4. What is the worst-case time complexity of this modified MERGESORT implementation, and how many calls of this modified MERGESORT function get executed if we call it on an array of length 32? **[4 Marks]**

iii. We change the MERGESORT function so that it uses SELECTION-SORT whenever it is called with an array of size at most $\sqrt{n}$, where n denotes the length of the array that was passed to the initial call of MERGESORT (not the length of the array that is passed to the current recursive call of MERGESORT, which may be much smaller than n). What is the worst-case time complexity of this modified version of MERGESORT? Justify your answer. **[4 Marks]**

END OF PAPER