Exam: TDDD86

Data Structures, Algorithms and Programming Paradigms

2020-08-25 kl: 08-12

On-call (jour):

Ahmed Rezine, ahmed.rezine@liu.se, 013 - 28 1938

Admitted material:

- You can access your individual notes, books, and even search the internet.
- No contacts, whether physical or virtual, are allowed during the duration of the exam with any person, whether the person is related to the course or not, except for contacting the examiner via email for questions if any.
- Any **suspected breach** will be **systematically reported to the disciplinary board**.

General instructions:

- **The questions will refer to your "D1 D2 D3 D4" digits.** These are the last four digits of your "anonymous-ID". You should duplicate the last digit if your "anonymous-ID" has less than 4 digits. This "anonymous-ID" can be found on the Lisam submission page next to this document and to the deadline. For instance, if your "anonymous-ID" is A-2709 then your D1 is 2, your D2 is 7, your D3 is 0 and your D4 is 9. If your "anonymous-ID" is A-60013 then your D1 is 0, your D2 is 0, your D3 is 1 and your D4 is 3. If your "anonymous-ID" is A-123 then your D1 is 1, your D2 is 2, your D3 is 3 and your D4 is 3. Ask the examiner if this is unclear.
- You may answer in either English or Swedish.
- The questions are formulated in a way that you can answer with a text editor (e.g., vim, notepad, gedit, emacs) or an office program (e.g., Microsoft Word, Open/Libre Office)
- You can also work on paper and take pictures/scan your solutions.
- Be **precise and clearly motivate** all statements and reasoning.
- If in doubt about the question, write down your interpretation and assumptions.
- The exam is divided into two parts:
  - Part A with a maximum of 30 pts.
  - Part B with a maximum of 12 pts.
- Grading:
  - Grade 3 requires at least 25pts exclusively from Part A.
  - Grade 4 requires grade 3 is secured and at least 36 pts in total.
  - Grade 5 requires grade 3 is secured and at least 39 pts in total.

Part A

Preliminaries (1pt)

The questions may refer to a sequence (S= S1, S2, …, S16) of 16 positive integers. This sequence is determined by your four digits D1 D2 D3 D4 and is given by the following table:

15	D2+D3+1	D1+3	19	D3+D4+5	D2+1	8	D1+D4+1	D3+5	10	D1+D2+3	D4+2	2	D1+D3+6	D2+D4+1	14

You need to derive your own sequence "S". See first page on how to derive your 4 digits. Follow some examples for different digit sequences:

- D1D2D3D4 = 2709 then S = 15, 8, 5, 19, 14, 8, 8, 12, 5, 10, 12, 11, 2, 8, 17, 14
- D1D2D3D4 = 0013 then S = 15, 2, 3, 19, 9, 1, 8, 4, 6, 10, 3, 5, 2, 7, 4, 14
- D1D2D3D4 = 1233 then S = 15, 6, 4, 19, 11, 3, 8, 5, 8, 10, 6, 5, 2, 10, 6, 14

The questions will use S1 to mean the first element of the sequence and S16 to mean the last. The other elements are referred to in a similar manner. For example, S13 is the 13th element.

1. Derive your own sequence according to the above and fill in the following table or give the sequence directly (1pt).

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16
=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=
15	.	.	19	.	.	8	.	.	10	.	.	2	.	.	14

Traversal and representation of binary trees (5 pts)

Consider the binary tree in Figure 1. (This tree will likely not be "sorted"). Replace each node with the corresponding integer value from your sequence S and:

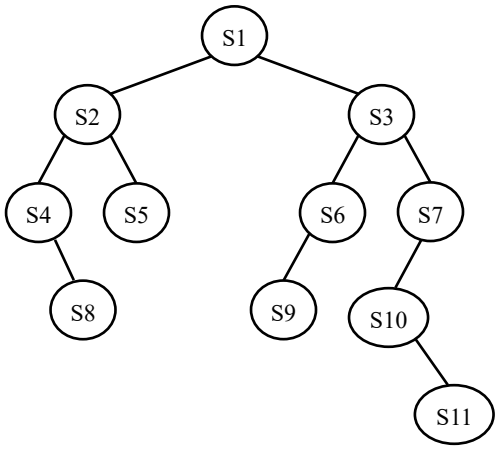

Figure 1

2. List the integer values encountered in a **pre-order** traversal of the tree. (1pt).

3. List the integer values encountered in an **in-order** traversal of the tree. (1pt).

4. List the integer values encountered in a **post-order** traversal of the tree. (1pt).

5. Recall that general binary trees can be

represented sequentially. We adopt the approach described in 8.3.1 in this OpenDSA Book. For instance, the binary tree in Figure 2 can be represented using the sequence: "AB/D//CEG///FH//I//", where the symbol "/" is used to represent a "null" child. Using the same approach, give a sequential representation of the integer values appearing in the tree of Figure 1. (2pts).

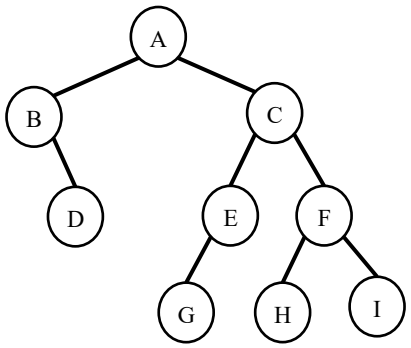

Figure 2

Binary search trees (8pts):

Assume Binary Search Trees satisfy the property that each right child has an integer value that is **strictly** larger than the one of its respective parent.

6. Give a sequential representation (see question 5 above) of the final binary search tree obtained by starting from an empty tree and inserting the integers in your sequence S one after the other (i.e., first insert S1, then S2, then S3 … then S16). Do not try to balance the tree. Call this tree T1. (2pts).

7. Explain the sequence of steps needed to check whether 13 is an element of T1. A step is either a comparison of two integers or a move to the right or left child or a parent. (2pts).

8. Give a sequential representation of the tree obtained by removing the value 14 from tree T1. Call the resulting tree T2. (2pts).

9. Give a sequential representation of the tree obtained by removing the value 8 from tree T2. (2pts).

Heaps and Heapsort (10pts)

10. Place the elements of your sequence S in a Min-Heap and give a sequential representation of the resulting binary tree. There can be many possible Min-Heaps for your sequence, just choose one. (2pts).

11. Explain two main differences between a Min-Heap and a Binary Search Tree. (hint: order and structure). (2pts).

12. Explain (convincingly but without getting in the algorithm details) why reading the value of a minimum of the values stored in a Min-Heap is in O(1). (2pts).

13. Explain (convincingly but without getting in the algorithm details) why adding an element to a Min-Heap is in O(log2(n)), where "n" is the number of elements in the heap and log2 is the binary logarithm. (2pts).

14. Heapsort starts by building a Heap of the values to be sorted and then repeatedly removes the root value and restoring the heap property. Give the sequential representations of the two Min-Heaps obtained after removing the two first roots (i.e., the Min-Heaps of sizes 15 then 14). (2pts).

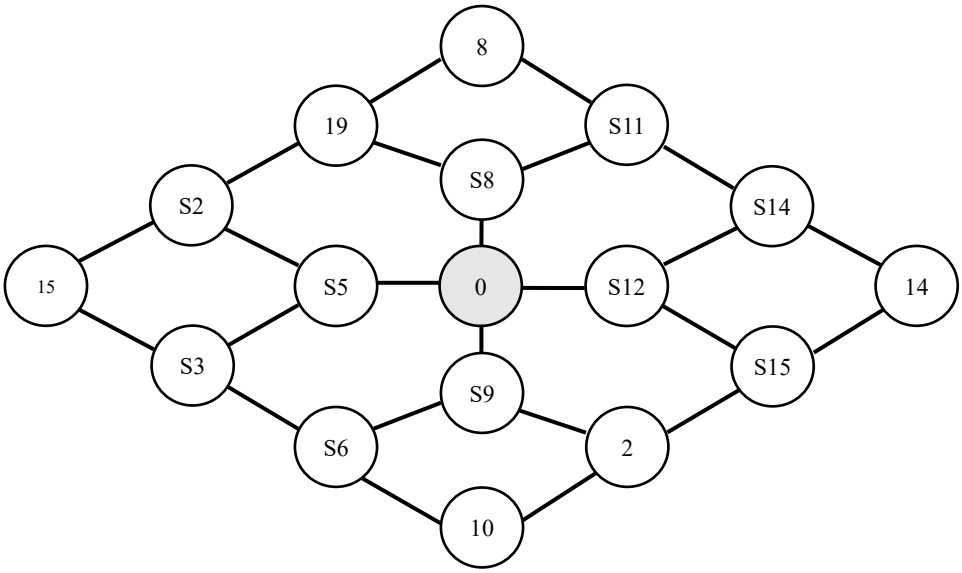

Figure 3.

Graphs and shortest paths (6pts)

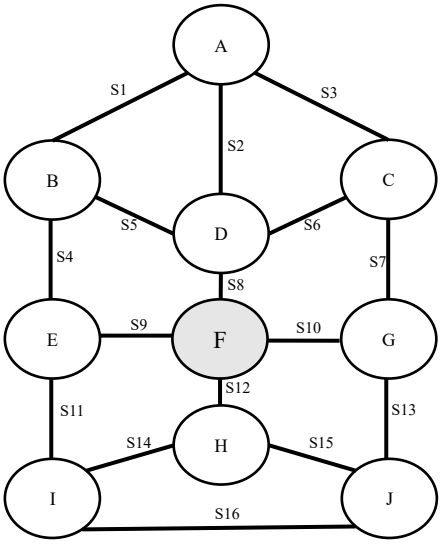

Figure 4

15. List the integer values in the order they are encountered in a **depth first traversal** of the graph in Figure 3. Start with the grey node with value 0. Chose a node with a smallest value to break ties if any. (2pts).

16. List the integer values in the order they are encountered in a **breadth first traversal** of the graph in Figure 3. Start with the grey node with value 0. Chose a node with a smallest value to break ties if any. (2pts).

17. Consider the graph in Figure 4. Replace the weights of the edges with the corresponding integer values from your sequence S. You can enumerate an edge by stating its two end-points. E.g., DC is the edge from node D to node C. Starting from node F, enumerate the edges in the order they are traversed in Dijkstra's algorithm. In case of similar costs, add nodes alphabetically. (2pts).

Part B:

Problem 1 (6pts):

Motivate your answers to the following questions:

18. Assume the worst-case time complexity of an algorithm is in $O(n^2)$. Does this contradict the existence of an input data that takes T(n) in $O(n)$ steps? (1pts).

19. Assume the worst-case time complexity of an algorithm is in $O(n^2)$. Does this contradict the existence of T(n) in O(n) describing the number of steps for all input data? (1pts).

20. Assume the worst-case time complexity of an algorithm is in $\Theta(n^2)$. Does this contradict the existence of an input data that takes T(n) in $\Omega(n)$ steps? (1pts).

21. Assume the worst-case time complexity of an algorithm is in $\Theta(n^2)$. Does this contradict the existence of T(n) in $O(n)$ describing the number of steps for all input data? (1pts).

22. Assume the worst-case time complexity of an algorithm is in $\Omega(n^2)$. Does this contradict the existence of an input data that takes T(n) in $O(n)$ steps? (1pts).

23. Assume the worst-case time complexity of an algorithm is in $\Omega(n^2)$. Does this contradict the existence of T(n) in $O(n)$ describing the number of steps for all input data? (1pts).

Problem 2 (6 pts):

We analyze the sorting algorithm "mystery" described in the following page. The algorithm takes a sequence $a_1, a_2, \dots, a_n$ of integers to be sorted. The length n of the sequence is assumed to be a power of 2. The algorithm uses the recursive procedure "merge", also described in the following page.

1. Let $\_merge(n)$ be the number of steps performed by "merge" for an input of length n. Show $\_merge(n) \leq \text{c. n. } log_2(n)$. (3pts).

2. Let $\_mystery(n)$ be the number of steps performed by "mystery" on an input of length n. Find, and justify, a tight upper-bound for $\_mystery(n)$. (3pts).

mystery($a_1, a_2, \ldots, a_n$)

Input:
$a_1, a_2, \ldots, a_n$ are n integers where n is a power of 2.

Output:
a sorted permutation of $a_1, a_2, \ldots, a_n$

if $(n > 1)$ **then**
1. apply mystery($a_1, a_2, \ldots, a_{n/2}$) first half
2. apply mystery($a_{n/2+1}, a_{n/2+2}, \ldots, a_n$) to second half
3. apply merge to the concatenation of the two sorted halves obtained by steps 1,2 above

merge($a_1, a_2, \ldots, a_n$)

Input:
$a_1, a_2, \ldots, a_n$ are n integers where both halves $a_1, a_2, \ldots, a_{n/2-1}$ and $a_{n/2}, a_{n/2+1}, \ldots, a_n$ are sorted.

Output:
a sorted permutation of $a_1, a_2, \ldots, a_n$

if $(n > 2)$ **then**
1. apply merge($a_1, a_3, \ldots, a_{n-1}$) to odd positions
2. apply merge($a_2, a_4, \ldots, a_n$) to even positions
3. swap, if needed, a_i with a_{i+1} for each $i \in \{2,4,6, \ldots, n-2\}$

else
1. swap, if needed, a_1 and a_2