Exam: TDDD86

Data Structures, Algorithms and Programming Paradigms

2020-12-17 kl: 08-12

On-call (jour):

Ahmed Rezine, ahmed.rezine@liu.se

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" digits.** These are the last three digits of your "anonymous-ID". You should duplicate the last digit if your "anonymous-ID" has less than three 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 7, your D2 is 0 and your D3 is 9. If your "anonymous-ID" is A-60003 then your D1 is 0, your D2 is 0 and your D3 is 3. If your "anonymous-ID" is A-12 then your D1 is 1, your D2 is 2 and your D3 is 2. Ask the examiner if this is unclear.
- You may answer in either English or Swedish.
- **Submit a single PDF with your answers. The PDF should only contain text** (e.g., no pictures, drawings, handwritten answers).
- The questions are formulated so that you can answer with a text editor (e.g., vim, notepad, emacs) or an office program (e.g., Microsoft Word, Open/Libre Office).
- 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 17 pts.
- Grading:
  - Grade 3 requires at least 20 pts exclusively from Part A.
  - Grade 4 requires grade 3 is secured and at least 6 pts from Part B.
  - Grade 5 requires grade 3 is secured and at least 10 pts from Part B.

Preliminaries

- Recall that general binary trees can be represented sequentially. We adopt the approach described in 8.3.1 in this OpenDSA Book[1]. 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. Do not draw your trees! Use this approach instead.
- You can write Theta for Θ and Omega for Ω.

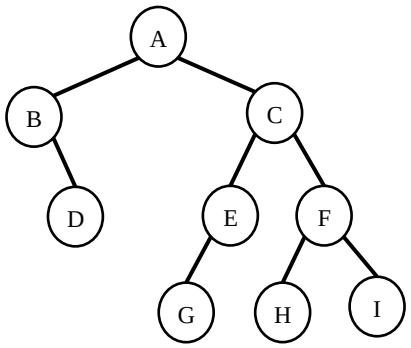

Figure 1

Part A

Sequence (1pt)

The questions refer to your sequence (S= S1, S2, …, S16) of 16 positive and pairwise different integers. Using your D1D2D3 digits, you should retrieve your sequence from the file "sequences.xlsx" (a PDF version "sequences.pdf" is also made available for convenience). The two files are accessible from the same place you downloaded this with the questions (i.e., from the LISAM submissions/inlämningar page for this exam).

Follow some examples for different digit sequences:

- D1D2D3 = 709 then S = 6, 2, 12, 11, 10, 1, 19, 4, 13, 18, 14, 9, 17, 7, 3, 20
- D1D2D3 = 003 then S = 8, 9, 10, 14, 16, 1, 3, 12, 15, 7, 17, 2, 4, 20, 19, 13
- D1D2D3 = 122 then S = 7, 1, 10, 4, 10, 15, 13, 12, 3, 16, 6, 2, 5, 8, 14, 20

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 = .

Binary search trees and their traversal (12 pts)

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 (i.e., when inserting: go to the left in case of equality).

1. Give a sequential representation (see the preliminaries in page 2) of the final

[1] https://www.ida.liu.se/opendsa/Books/TDDD86F20/html/SequentialRep.html

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).

2. List the integer values encountered in an **in-order** traversal of T1. (2pts).

3. List the integer values encountered in a **post-order** traversal of T1. (2pts).

4. Give a sequential representation of the tree obtained by removing the root node from tree T1. (2pts).

5. Give a sequence S' that is a reordering of the 15 first elements of S (i.e., S' is a permutation of S1, S2, …, S15) so that the binary search tree obtained by starting from an empty tree and inserting the integers in S' one after the other is of maximum height. (2pts).

6. Give a sequence S'' that is a reordering of the 15 first elements from S (i.e., S'' is a permutation of S1, S2, …, S15) so that the binary search tree obtained by starting from an empty tree and inserting the integers in S'' one after the other is of minimum height. (2pts).

Sorting (9pts)

Define H = S1, S2, S3, S4 to be the first four elements of your sequence S.

7. Consider the insertion sort algorithm described in 11.3.1 in this OpenDSA book[2]. Consider a call `inssort(A,4)` where the array `A` is of length 8 and contains the sequence H (i.e., `A[0]=S1, A[1]=S2, A[2]=S3, A[3]=S4`). Write down the sequence of eight values in array `A` after each swap (there will be as many sequences as there are swaps during the considered execution). (3pts).

8. Rearrange the sequence H so you get a minimum number of swaps. (Just give the permutation of H, not the sequence of arrays like in the previous question). (2pts).

9. What is the asymptotic worst-case complexity (in the form $\Theta(f(n))$) of the algorithm? Explain. (2pts).

10. Consider the Mergesort algorithm described in 11.9.1 in this OpenDSA book[3]. Suppose we executing a Mergesort procedure that took a list `L` containing the sequence S1,…,S8 of the first eight elements of your sequence S. Suppose we have just returned from the two recursive calls on the two halves of the sequence `L` with the list `L1` containing the result of the recursive call on the first half and `L2` containing the result of the recursive call on the second half. Give the sequence of values of `L1` and of `L2`. Explain the merge step and why it is linear in the sum of the lengths of `L1` and `L2`. (2pts).

---

[2] https://www.ida.liu.se/opendsa/Books/TDDD86F20/html/InsertionSort.html
[3] https://www.ida.liu.se/opendsa/Books/TDDD86F20/html/Mergesort.html

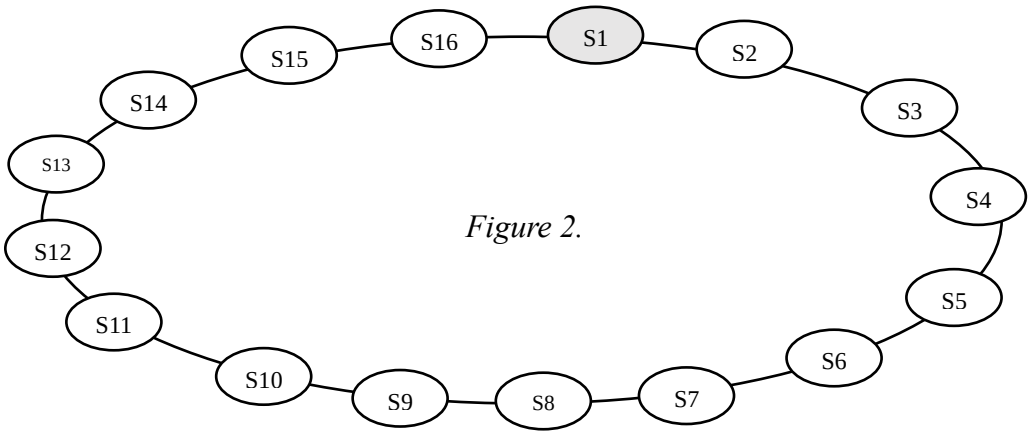

Figure 2.

Graphs and shortest paths (8pts)

11. List the integer values in the order they are processed in a **depth first traversal** of the graph in Figure 2. Start with the grey node with value S1. Chose a node with a smallest value to break ties if any. (2pts).

12. List the integer values in the order they are processed in a **breadth first traversal** of the graph in Figure 2. Start with the grey node with value S1. Chose a node with a smallest value to break ties if any. (2pts).

13. Consider the graph in Figure 3. 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 A, enumerate the edges in the order they are traversed in Dijkstra's algorithm for single-source shortest path. In case of similar costs, add nodes alphabetically. (4pts).

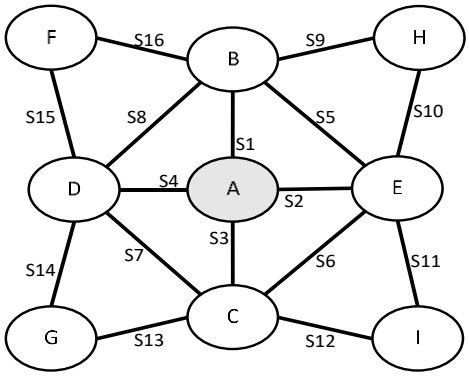

Figure 3

Part B:

Problem 1 (8pts):

AVL trees. Recall the height of the tree is the length of the longest paths from a root to a leaf. The height of a tree consisting of a single node is 0.

1. Can you have an AVL tree of height 2 using exactly the 4 first elements S1, S2, S3, S4 of your sequence S? if yes give a sequential representation of the tree, justify otherwise. (2pts).

2. Can you have an AVL tree of height 3 using exactly the 7 first elements S1, S2, S3, …, S7 of your sequence S? if yes give a sequential representation of the tree, justify otherwise. (2pts).

3. Can you obtain an AVL tree of height 4 using exactly the 12 first elements S1, S2, S3, …, S12 of your sequence S? if yes give a sequential representation of the tree, justify otherwise. (2pts).

4. Can you obtain an AVL tree of height 5 using exactly the elements S1, S2, …, S16 of your sequence S? if yes give a sequential representation of the tree, justify otherwise. (2pts)

Problem 2 (9 pts):

Consider the methods `main`, `foo` and `bar` listed below. Suppose the size of the array `a` is a power of 2. Assume array accesses are legitimate[4].

1. What is the result printed by `main` if the array was defined using your sequence S1, S2, … as follows: `a[8] = {S1-0,S2-S1,S3-S2,…,S8-S7};` (3pts).
2. Suppose we call `bar(a, low, mid, high, …)` with inputs `a, low, mid` and `high`. Let **bar_steps(n)** be the number of steps executed in the worst case by a call to `bar` when the number of elements from `low` to `high`, both inclusive, is n (i.e., n = `high-low+1`). Give a function **f(n)** for which **bar_steps(n)** is in Θ(**f(n)**). Justify. (2pts).
3. Suppose we call `foo(a, low, high, …)` with the inputs `a, low` and `high`. Let **foo_steps(n)** be the number of steps executed in the worst case by a call to `foo` when the number of elements from `low` to `high`, both inclusive, is n (i.e., n = `high-low+1`). Give a recurrence definition for **foo_steps(n)** in terms of **foo_steps(n/2)** and of **bar_steps(n)**. (2pts).
4. Give a function **g** resulting in a tight class of functions O(**g(n)**) that includes **foo_steps**. Justify. (2pts).

[4] each access `a[i]` occurs for $0 \leq i < |a|$ where |a| is the size of the array a.

```
int main(){
    int low, high, sum;
    int a[8] = {15,…}; //a[8]=S1-0, S2-S1, S3-S2,…, S8-S7
    foo(a, 0, 7, low, high, sum);
    cout << "(low,high,sum)=(" << low << "," << high << "," << sum << ")" << endl;
    return 0;
}
```

```
// input: array a with |a| elements
// input: indices 0 <= low <= high < |a|
// output: indices rslt_low, rslt_high and sum of values rslt_sum
void foo(int a[], int low, int high,int& rslt_low, int& rslt_high, int& rslt_sum){
    if (high == low){
        rslt_low=low; rslt_high=high; rslt_sum = a[low];
    }else{
       int mid = (low + high)/2;

       int left_low, left_high, left_sum;
        foo(a, low, mid, left_low, left_high, left_sum);

        int right_low, right_high, right_sum;
        foo(a, mid+1, high, right_low, right_high, right_sum);

        int mid_low, mid_high, mid_sum;
        bar(a, low, mid, high, mid_low, mid_high, mid_sum);

        if(left_sum >= right_sum && left_sum >= mid_sum){
            rslt_low=left_low; rslt_high=left_high; rslt_sum=left_sum;
        }else if(right_sum >= left_sum && right_sum >= mid_sum){
            rslt_low=right_low; rslt_high=right_high; rslt_sum=right_sum;
        }else {
            rslt_low=mid_low; rslt_high=mid_high; rslt_sum=mid_sum;
        }
    }
}
```

```
// input: array a[] with |a| elements
// input: indices 0 <= low <= mid <= high < |a|
// output: indices rslt_low, rslt_high and a sum of values rslt_sum
void bar(int a[],int low,int mid,int high,int& rslt_low,int& rslt_high,int& rslt_sum){
    int left_sum = INT_MIN; // left_sum initialized to smallest negative number
    int sum = 0;
    for(int i = mid; i >= low; i--){
        sum += a[i];
        if(sum > left_sum){
            left_sum = sum;
            rslt_low = i;
        }
    }
    int right_sum = INT_MIN; // right_sum initialized to smallest negative number
    sum = 0;
    for(int i = mid+1; i <= high; i++){
        sum += a[i];
        if(sum > right_sum){
            right_sum = sum;
            rslt_high = i;
        }
    }
    rslt_sum = left_sum + right_sum;
}
```