Exam: TDDD86

Data Structures, Algorithms and Programming Paradigms

2025-12-16 kl: 14-18

On-call (jour): Ahmed Rezine (tel:1938)

Specific instructions for the computer exams:

- In summary: you log in with your LiU-ID and your private password. You can only save files in the desktop. We might leave files for you in the read-only "given_files" folder (e.g., lecture slides). **You will use the "student chat client", or "student client" to receive information during the exam, to ask questions and to submit your solution.** More details in the "EXAM_README.pdf" under "given_files".

- Your "student client" should start automatically, if you close it and need to start it again, double click on the "fish icon" on your desktop.

- Submit one file with all your answers. The document should only contain text with a .txt suffix (e.g., answers.txt). The document should not contain drawings or pictures. We will only look at the last submitted file.

- The questions are formulated so that you can answer with any text editor (e.g., vi, emacs, gedit, etc).

- You can access OpenDSA using chromium. The start page will list available links.

General instructions:

- You may answer in either English or Swedish.

- If in doubt about a question, write down your interpretation and assumptions.

- The exam is divided into two parts:
  - Part A with a maximum of 35 pts.
  - Part B with a maximum of 20 pts.

- Grading:
  - Grade 3 requires at least 20 pts exclusively from Part A.
  - Grade 4 requires grade 3 is secured and at least 8 pts from Part B.
  - Grade 5 requires grade 3 is secured and at least 12 pts from Part B.

Part A

Problem A.1: Asymptotic execution time (min 0 pts, max 10 pts)

Consider the five methods `f1`, `f2`, `f3`, `f4`, `f5` and the nine complexity classes (A)-(I) depicted below. Assume the manipulated arrays are large enough. The asymptotic analysis is to be carried out with respect to the number of elements between the indices lo and hi (inclusive), i.e., $n = hi - lo + 1$. If it simplifies your reasoning, you can restrict the analysis to sizes of the form $n = 2^p$ or $n = 2^p - 1$ for some natural number p.

```
int f1(int a[], int lo, int hi){
   int count = 0;
   int i = j = lo;
   while(i != hi + 1){
      if(a[i] == a[j] + j - i){
         count++;
      }
      if(j < hi){
         j++;
      }else{
         i++;
         j = lo;
         if(a[j] == 0) {
            return -1;
         }
      }
   }
   return count;
}
```

```
int f2(int a[], int lo, int hi){
   for(int i = lo; i <= hi; i++){
      for(int j = i + 1; j <= hi; j++){
         if(a[i] == a[j] + j - i){
            return j - i;
         }
      }
   }
   return -1;
}
```

```
int f3(int x, int a[], int lo, int hi){
  int count = 0;
  for(int j = lo; j <= hi; j++){
     count = count + a[j];
  }
  if((lo == hi) || (count < 0)){
     return count;
  }
  int m = lo + (hi - lo)/2;
  int count1 = f3(x, a, lo, m);
  int count2 = f3(x, a, m + 1, hi);
  return count1 + count2;
}
```

```
int f4(int count, int a[], int lo, int hi){
   for(int j = lo; j <= hi; j++){
     if(!f4(count + a[lo], lo + 1, hi)){
        return f4(count, lo + 1, hi);
     }
  }
     return count == 0;
}
```

```
int f5(int a[], int lo, int hi){
  for(int i= lo + 1; i < hi && i < 1000; i++){
    if(a[i-1] == a[i]){
      return i;
    }
  }
  return 1;
}
```

Complexity classes:

(A) $\Theta(1)$ (D) $\Theta(n \log n)$ (G) $\Theta(2^n)$
(B) $\Theta(\log n)$ (E) $\Theta(n^2)$ (H) $\Theta(3^n)$
(C) $\Theta(n)$ (F) $\Theta(n^3)$ (I) $\Theta(n!)$

1. For each one of the 5 methods above, give (without justification!) the complexity class among the classes (A-I) that best matches its asymptotic **worst-case** execution time. (For each method, 1pts if correct, 0 if not answered, -1pts if incorrect.)

2. For each one of the 5 methods above, give (without justification!) the complexity class among the classes (A-I) that best matches its asymptotic **best-case** execution time. (For each method, 1pts if correct, 0 if not answered, -1pts if incorrect.)

Problem A.2: Hashing and conflict resolution (min 0 pts, max 8 pts)

Assume linear probing is used (i.e., the probe function is $p(k, i) = i$). In addition, assume we use an array of size 9 **with indices 0 to 8**. The array is used to implement a hash table where the hash function hashes the keys A-I as given by the following table:

Key	A	B	C	D	E	F	G	H	I
Hash Value	0	3	4	3	2	2	8	8	3

In other words, the "home position" of key A is 0 and keys E and F hash both to 2.

3. Give the content of each cell of the table after inserting, starting from an empty table, the sequence A, B, C, D, E, F, G, H, I (i.e., inserting first A, then B, then C … and finally I). (2pts if correct, 0 if not answered, -2pts if incorrect).

4. Answer with yes or no (no need for justification). Is there a sequence that results, starting from an empty table, in this table? (2pts if correct, 0 if not answered, -2pts if incorrect).

Index	0	1	2	3	4	5	6	7	8
Content	E	F	D	B	A	C	G	H	I

5. Answer with yes or no (no need for justification). Is there a sequence that results, starting from an empty table, in this table? (2pts if correct, 0 if not answered, -2pts if incorrect).

Index	0	1	2	3	4	5	6	7	8
Content	A	G	F	B	D	C	E	I	H

6. Answer with yes or no (no need for justification). Recall a hash function computes "home positions" for all keys. Assume a hash function that extends the first table depicted in this problem and suppose we use linear probing like above. Does this hashing approach suffer from secondary clustering? (2pts if correct, 0 if not answered, -2pts if incorrect).

Problem A3. Sorting (min 0 pts, max 5 pts)

7. Answer with yes or no (no need for justification). Is it possible to have a "`heapify`" procedure with a worst-case asymptotic time complexity in $O(n)$ that, given any integer array of size n, reorganizes the integers to obtain a min-heap? (2pts if correct, 0 if not answered, -2pts if incorrect).

8. Consider the following min-heap:

Index	0	1	2	3	4	5	6
Content	10	12	15	13	14	16	17

Given an array of integers to sort in increasing order, the heapsort algorithm first generates a min-heap (like the one above) and then repeatedly pops elements from the min-heap. You can describe a min-heap by a comma separated enumeration of its elements. For instance, the min-heap above can be described with the enumeration: 10, 12, 15, 13, 14, 16, 17.

Give the sequence of six intermediary min-heaps obtained by heapsort when starting from the min-heap above:

a. Second min-heap (0.5pts if correct, 0 if not answered, -0.5pts if incorrect):

Index	0	1	2	3	4	5
Content						

b. Third min-heap (0.5pts if correct, 0 if not answered, -0.5pts if incorrect):

Index	0	1	2	3	4
Content					

c. Fourth min-heap (0.5pts if correct, 0 if not answered, -0.5pts if incorrect):

Index	0	1	2	3
Content				

d. Fifth min-heap (0.5pts if correct, 0 if not answered, -0.5pts if incorrect):

Index	0	1	2
Content			

e. Sixth min-heap (0.5pts if correct, 0 if not answered, -0.5pts if incorrect):

Index	0	1
Content		

f. Seventh min-heap (0.5pts if correct, 0 if not answered, -0.5pts if incorrect):

Index	0
Content	

Problem A4. Binary search trees (min 0 pts, max 8 pts)

Assume the binary search tree T_1 depicted in Figure 1.

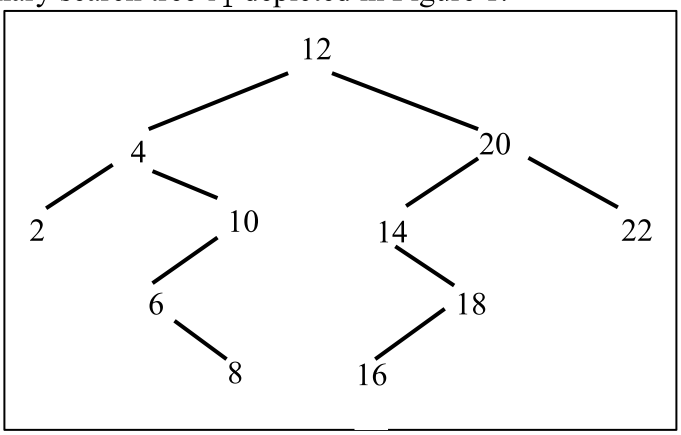

Figure 1. The binary search tree T_1 used in Problem A.4

9. Give a sequence of integers that results, if inserted from the first to the last element of the sequence, in the tree T_1 depicted in Figure 1. (2pts if correct, 0 if not answered, -2pts if incorrect).

Recall that binary trees can be represented sequentially. We adopt the approach described in 8.3.1 in OpenDSA. For instance, the binary tree in Figure 2 can be sequentially represented using the sequence: "A B / D / / C E G / / / F H / / I / /". The symbol "/" is used to represent a "null" child.

Do not draw trees in your answers! Use this approach instead.

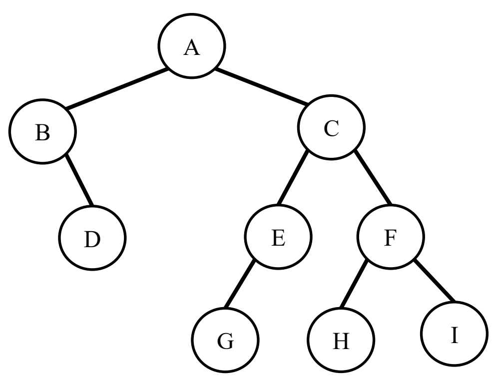

Figure 2. A B / D // C E G /// F H // I //

10. Give a sequential representation of the binary search tree obtained by removing from tree T_1 the node with key value 12. Call the obtained tree T_2. (2pts if correct, 0 if not answered, -2pts if incorrect).

11. Give a sequential representation of the binary search tree T_3 obtained by inserting the key 7 to the tree T_2 you obtained in the previous question. (2pts if correct, 0 if not answered, -2pts if incorrect).

12. Give a sequential representation of the binary search tree T_4 you obtain after performing a **splay(10)** operation on the tree T_1 depicted in Figure 1 (observe this is tree T_1 described at the beginning of the problem, **not** those obtained from questions 10 or 11 above). (2pts if correct, 0 if not answered, -2pts if incorrect).

Problem A.5: Graphs (min 0 pts, max 4 pts)

13. Give a topological sort of the directed graph depicted in Figure 3, or state that the graph does not admit a topological sort if you think it does not admit a topological sort. (2pts if two correct and different sorts, 0 if not answered, -2pts otherwise)

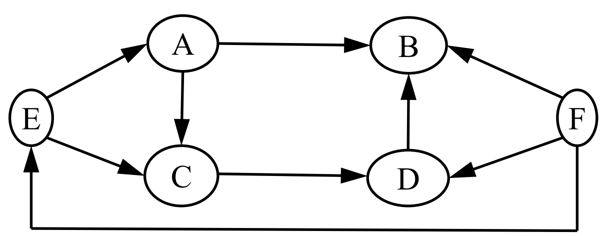

Figure 3. Directed graph for the topological sorting question A5.13

14. Give the nodes of one maximal strongly connected component in the graph depicted in Figure 4 if it has strongly connected components, otherwise state there are no strongly connected components in the graph. Observe single nodes without self-loops are not considered strongly connected components on their own. (2pts if correct, 0 if not answered, -2pts if incorrect).

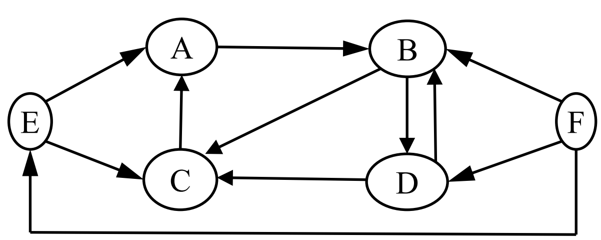

Figure 4. Directed graph for the topological sorting question A5.14

Part B:

Problem B.1 (min 0pt, max 5pts):

Recall that the height of a binary search tree is the length of a longest path from the root to a leaf. The height of a tree consisting of a single node is therefore 0.

AVL trees. Let $minNodes(h)$ be the minimum number of nodes in any AVL tree of height h. In other words, given a value h, it is (1) possible to build an AVL tree of height h and containing $minNodes(h)$ elements, and (2) there is not AVL tree of height h and containing $minNodes(h) - 1$ or less elements.

15. Give, without justification, $minNodes(3)$. (1pt if correct, 0 if not answered, -1pts if incorrect)

16. For any $h > 3$, give a recurrence relation defining $minNodes(h)$ in terms of $minNodes$ for a finite number of shorter trees. (i.e., AVL trees with strictly smaller heights). **Explain** why your recurrence relation is correct for all $h > 3$**.** (2pts).

17. Given an AVL tree t, write $height(t)$ and nodes(t) for the height and the number of nodes of t, respectively. Using the recurrence relation obtained in the question above, **show (**without using the Master Theorem!) that there are three constants c_1 c_2 and c_3 such that, for any AVL tree t with $height(t) > 3$, it is the case that:

$$height(t) \leq c_1 \log_2(nodes(t) + c_2) + c_3.$$

The constants c_1, c_2 and c_3 should be independent of the tree t. (2pts).

Problem B.2 (min 0pts, max 3pts)

Answer with yes or no (no need for justification). For each answer, 1pts if correct, 0 if not answered, -1pts if incorrect:

18. Assume the worst-case time complexity of an algorithm is in $\Theta(\mathrm{n}^2)$. Does this contradict the existence of a family of inputs, one input for each size n, on which the algorithm takes $t(n)$ steps and where we know that $t(n)$ belongs to $\Omega(n)$?

19. Assume the worst-case time complexity of an algorithm is in $O(\mathrm{n}^2)$. Does this contradict the existence of a family of inputs, one input for each size n, on which the algorithm takes $t(n)$ steps and where we know t(n) belongs to $\Omega(n^2 \log(n))$?

20. Assume the best-case time complexity of an algorithm is in $\Omega(\mathrm{n}^2)$. Does this contradict that there are constants c_1, c_2 and c_3 such that the algorithm always takes $(\lfloor c_1 n \log(n) \rfloor + c_2 n + c_3)$ steps?

Problem B.3 (max 12 pts):

You will be asked to provide codes for C++ solutions. These codes will manipulate C++ std vectors of integers using instructions and methods like those used by the method `foo` listed below. You should not use vector methods other than those needed for getting the size of a vector and for reading or writing the vector content at some index (similar to `foo`). The listed method `foo` gets as input a reference to a vector of integers together with references to two integers. Operations such as reading or writing a vector at some position, or passing a reference to a vector as a parameter are considered constant time. Obtaining the min/max integer values with `numeric_limits<int>::min()` and `numeric_limits<int>::max()` are also constant time. Observe this yields a worst-case time complexity in $\Theta(n)$ for the method `foo` (where n is the number of elements in the input vector `p`).

```
int foo(const vector<int>& p, int& min, int& max){
  min = numeric_limits<int>::max();
  max = numeric_limits<int>::min();
  int sum = 0;
  for(int i = 0; i < p.size(); i++){
    if(min > p[i]){
      min = p[i];
    }
    if(max < p[i]){
      max = p[i];
    }
    sum = sum + p[i];
  }
  return sum;
}
```

The single buy – single sell maximum profit problem

Assume you are allowed to buy a single item and then to sell it (in that order, i.e., buy first then sell). You can buy and sell on the same day (you get a profit of 0 sek), or on any two different days as long as selling does not occur before buying. You want to make the largest possible profit. For this, you are given access to the future n prices of the item. For instance, if $n = 6$ and the prices are given by the vector `prices` below:

Time	0	1	2	3	4	5
Price	50	60	10	40	5	30
Change		10	-50	30	-35	25

Then a maximum profit can be made by buying the item on day 2 (for 10 sek) and by selling it on day 3 (for 40 sek), thus making a profit of 30 sek. We will consider different approaches to identify an "optimal interval" (i.e., to identify buying and selling points to maximize the profit).

21. A brute force solution. Give a C++ implementation of a method:

```
int brute(const vector<int>& prices, int& bp, int& sp)
```

that **computes** the maximum profit given a vector `prices` of size n together with the corresponding buying index `bp` and selling index `sp`. The return value is the largest profit and the parameters `bp` and `sp` are passed by reference and are only used to return buying and selling indices giving the largest profit. Their initial value, when calling `brute`, should not influence the execution of `brute`. The worst-case asymptotic time complexity of `brute` should be in $\Theta(n^2)$. Explain why your solution is correct (i.e., why is it that the computed values of bp and sp and the one returned by the method correspond to a maximal profit). (2pt).

22. A divide and conquer solution. Suppose you are given a range $[lo, hi]$ with $0 \leq lo \leq hi < n$ where n is the size of a vector of prices. Let mid be the midpoint (in terms of indices) between indices lo and hi. The optimal profit is obtained by buying the item at an index "bp" and selling it at an index "sp". Observe that sp might be smaller than mid (an interval with maximal profit is before the midpoint), that bp might be larger than mid (the interval is after the midpoint), or neither (the midpoint is in the interval). Use this observation and give a C++ implementation of:

```
int dAc (const vector<int>& prices, int& bp, int& sp)
```

The method `dAc` should use the divide and conquer paradigm (involving one or more recursive C++ methods that you should give) to return a maximal profit together with an interval $[bp, sp]$ that results in the largest profit. The worst-case asymptotic time complexity of your solution should be in $\Theta(n \log(n))$. Explain why your solution is correct (i.e., why is it that the computed values of bp and sp and the one returned by the method correspond to a maximal profit). (4pts)

23. Complexity of the divide and conquer solution. Clearly show (without using the Master theorem) why is it the case that the worst-case asymptotic time complexity of your solution is in $\Theta(n \log(n))$. (3pts).

24. Iterative solution. It is possible to obtain a linear solution to the single buy- single sell maximal profit problem above by observing that an optimal interval in $[0, j+1]$ is either already an optimal interval in$[0, j]$ or an interval of the form $[i, j+1]$. Give a C++ solution:

```
int linear(const std::vector<int>& p, int& bp, int& sp)
```

The solution should be in $\Theta(n)$. Explain why the solution is correct (i.e., why is it that the computed values of bp and sp and the one returned by the method correspond to a maximal profit). (3pts)