Algorithms and Data Structures (BADS/SGDS)

Exam 30 May 2016

Thore Husfeldt, ITU

Instructions

What to bring. You can bring any written aid you want. This includes the course book and a dictionary. In fact, these two things are the only aids that make sense, so I recommend you bring them and only them. But if you want to bring other books, notes, print-out of code, old exams, or today's newspaper you can do so. (It won't help.)

Answering multiple-choice questions. In the multiple-choice questions, there is one and only one correct answer. However, to demonstrate partial knowledge, you are allowed to check 2 or more boxes, but this earns you less than full points for that question.

number of checked boxes	0	1	2	3	4
points if correct answer checked		1	0.5	0.21	0
points if correct answer not checked	0	−0.33	−0.5	−0.62	

In particular, the best thing is to only check the correct answer, and the worst thing is to check all answers but the correct one. If you don't check anything (or check *all* boxes) your score is 0. Also, if you check boxes at random, your expected score is 0. (Just to make sure: a question that is not multiple-choice cannot give you negative points.)

Where to write. Mark you answers to questions 1–3 on pages 7 and 8. If you really have to, you may use separate sheets of paper for these questions instead, but please be clear about it (cross out everything and write "see separate paper, page 1" or something like that.) Question 4 is answered on separate sheet(s) of paper anyway. For the love of all that is Good and Holy, write legibly. Hand in pages 7 and 8, and any separate sheet(s) of paper. Do not hand in pages 1–6, they will not be read.

Exam questions

1. Analysis of algorithms

(a) (*1 pt.*) Which pair of functions satisfy $f(N) \sim g(N)$?

A $f(N) = 2N$ and $g(N) = N + 2$
B $f(N) = 2N$ and $g(N) = N^2$
C $f(N) = 2N^2 + N$ and $g(N) = 2N^2 + 2N$
D $f(N) = 2N^2 + N$ and $g(N) = N^2 + 2N$

(b) (*1 pt.*) Which pair of functions satisfy $f(N) = O(g(N))$?

A $f(N) = N + N + N$ and $g(N) = N \log N$
B $f(N) = (N+1) \cdot (N+1)$ and $g(N) = N + 1$
C $f(N) = (\log N) \cdot (\log N)$ and $g(N) = \log N$
D $f(N) = N^5$ and $g(N) = 4N^4 + 5N$

(c) (*1 pt.*) How many stars are printed?

```
for (int i = 1 ; i < N; i = i*2) StdOut.print("*");
```

Base revision 1d61dee. . ., Mon Aug 10 12:00:17 2020 +0200, Riko Jacob @X260.

A $\sim \log_2 N$ B $\sim N/2$ C $\sim N$ D $\sim \frac{1}{2}N^2$

(d) (*1 pt.*) How many stars are printed? (Choose the smallest correct estimate.)

```
for (int i = 0; i < N; i = i+1)
     for (int j = i; j > 0; j = j-1)
          StdOut.print("*");
```

A $O(\log N)$ B $O(N)$ C $O(N \log N)$ D $O(N^2)$

(e) (*1 pt.*) What is the asymptotic running time of the following piece of code? (Choose the smallest correct estimate.)

```
if (N < 1000) for (int i = 0; i < N*N*N; i = i+1) A[i] = i;
else          for (int i = 0; i <   N*N; i = i+1) A[i] = 2*i*i;
```

A linear in N B linearithmic in N C quadratic in N D cubic N

(f) (*1 pt.*) Find a recurrence relation for the number of arithmitic operations (additions and subtractions) performed by the following recursive method. (Choose the smallest correct estimate. The base case is $T(0) = 0$ in all cases.)

```
static int r(int N)
{
    if (N > 0) return r(N-1) + N;
    else       return 1;
}
```

A $T(N) = T(N-1) + N$ B $T(N) = T(N) + T(N-1)$
C $T(N) = T(N-1) + 2$ D $T(N) = T(N) + N - 1$

(g) (*1 pt.*) Assume I have a function `f(int K)` that runs in amortised logarithmic time in K, but linear worst case time. What is the running time of

```
for (int i = 0; i<N; i=i+1) f(N);
```

(Choose the smallest correct estimate.)

A Linearithmic in N.

B Amortised linear in N.

C Quadratic in N.

D Impossible to say from the information given.

2. Class M. The next few questions all concern the class defined in fig. 1. Assume the stacks S and T are implemented as a Linked List.

(a) (*1 pt.*) Class M behaves like which well-known data structure?

A (LIFO) Stack. B (FIFO) Queue. C Priority queue. D Union–Find.

(b) (*1 pt.*) Draw the data structure after the following operations: (This means "*at the end* of the operations," not "*after each* operation," so you need to draw only a single picture. Make sure you draw the entire data structure. Make sure to include all instance variables.)

```
M<Integer> m = new M<Integer>();
m.insert(1); m.insert(2); m.remove();
```

```java
import edu.princeton.cs.algs4.*;

public class M<Item>
{
        Stack<Item> S = new Stack<Item>();
        Stack<Item> T = new Stack<Item>();

        public void insert(Item i)
        {
                S.push(i);
        }

        public Item remove()
        {
                if (T.isEmpty()) move(S,T);
                return T.pop();
        }

        private void move(Stack<Item> from, Stack<Item> to)
        {
                while(!from.isEmpty()) to.push(from.pop());
        }
}
```

Figure 1: Class `M` (for Mystery). The method names `insert` and `remove` are purposefully vague.

(c) (*1 pt.*) What is the worst-case running time of `insert`? (Choose the smallest correct estimate.)

A $O(\log N)$. B $O(N)$. C $O(N \log N)$. D $O(1)$.

(d) (*1 pt.*) What is the worst-case running time of `remove`? (Choose the smallest correct estimate.)

A $O(\log N)$. B $O(N)$. C $O(N \log N)$. D $O(1)$.

(e) (*1 pt.*) What is the *total* running time of the following code. (Choose the smallest correct estimate.)

```
M<Integer> m = new M<Integer>();
for (int i = 0; i < N; i++) m.insert(i);
for (int i = 0; i < N; i++) m.remove();
```

A $O(\log N)$. B $O(N)$. C $O(N \log N)$. D $O(N^2)$.

(f) (*1 pt.*) Assume I used a resizing array instead of a linked list to implement the two stacks S and T. Which statement is true:

A Now all operations take constant worst-case time.

B Now `insert` takes linear worst-case time.

C Now `remove` takes constant worst-case time.

D The time for `insert` and `remove` does not change.

(g) (*1 pt.*) Add a method

```
public int size()
```

that returns the total number of elements in the data structure. *Don't change any other methods in class* `M` *and don't introduce new instance variables to* `M`.

(h) (*2 pt.*) Extend class `M` with a method `public void prepare()` with the following functionality. The method cleans up the data structure so that each of the next N calls to `remove` are guaranteed to run in constant worst-case time, where N is the current number of elements in the data structure. *Don't change any other methods in class* `M` *and don't introduce new instance variables to* `M`.

(i) (*2 pt.*)
Add a method

```
public Iterator<Item> iterator()
```

that iterates over the elements of the data structure in the order in which they were inserted. (Most recently inserted element last.) You are free to use the Stack iterator `Stack.iterator()`, but recall that that iterator reports the most recently inserted element *first.*

3. Operation of common algorithms and data structures.

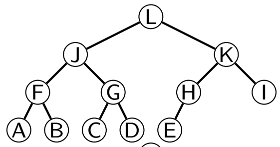
(a) (*1 pt.*) Consider the following sequence of operations on a data structure, where a number i means `insert(`i`)` and "$*$" means `remove()`. The data structure is initially empty.

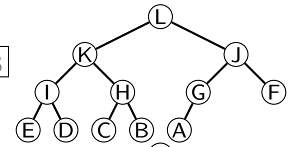
$$5 \quad * \quad 3 \quad 9 \quad 7 \quad *$$

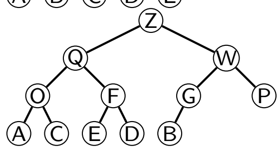
What is the data structure if the removed elements are 5, 7 in that order?

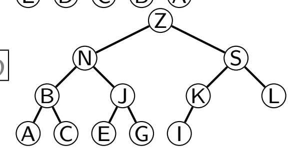
A Symbol table B Stack C Queue D Heap

(b) (*1 pt.*) Which of the following trees is *not* in heap order?

A 

B 

C 

D 

(c) (*1 pt.*) Insert the keys `4 6 8 1 2` in that order into a binary search tree (without any rebalancing, using algorithm 3.3 in [SW]). Draw the result.

(d) (*1 pt.*) Assume I sort the letters `S E Q U E N C E` using selection sort. Which one of the following situations *does* arise at some time during the algorithm?

A `C E Q U E N S E` B `C S E Q U E N E` C `S E Q U E C N E` D `E S Q U E N C E`

(e) (*1 pt.*) Consider the key–value pairs

key	P	A	N	A	M	A	P	A	P	E	R	S
value	0	1	2	3	4	5	6	7	8	9	10	11

We use the hash function `(key.hashCode() & 0x7fffffff) % 7`. To spare you the calculations, the hash values are here:

key	F, M, T	G, N, U	A, H, O, V	B, I, P, W	C, J, Q, X	D, K, R, Y	E, L, S, Z
hash	0	1	2	3	4	5	6

The keys are inserted from left to right into an initally empty hash table of size 7 using linear probing. Draw the result.

(f) (*1 pt.*) In which sense is mergesort better than insertion sort?

A It has better asympotic worst-case performance

B It is faster on sorted inputs.

C It is in-place.

D It uses less space.

(g) (*1 pt.*) Insert the letters `M A X I` in that order into a red–black BST as defined in the textbook (algorithm 3.4). Draw the resulting structure in the style of the book, use a fat edge to represent red links. (Your answer will be photocopied in black and white, so don't use fancy colours.)

(h) (*1 pt.*) Sort 4 strings using LSD string sort (algorithm 5.1 in [SW], with $N = 4$, $W = 3$). Give a trace in the style of the book by completing this table:

input	$d = 2$	$d = 1$	$d = 0$
BOB	IDA	...	
SUE	...		
IKE			
IDA			

4. Design of algorithms

In order to increase production, the government has devided to introduce minimum speed requirements on all roads. If a road has minimum speed requirement 120 km/h and the maximum speed of your old Citroën 2CV is only 108 km/h then you can't use that road.

The roadmap is given as a list of connections between cities like this:

```
Århus Odder 23 120
Odder Skanderborg 20 95
Skanderborg Hørning 10 100
Hørning Århus 20 107
Hørning Odder 19 120
```

The total number of different cities is C. The number of roads (which equals the number of lines in the description) is R. The format of the ith line is

$$f_i \quad t_i \quad d_i \quad m_i\,,$$

describing the two cities f_i and t_i joined by the ith road, their distance d_i (in kilometers), and their minimum speed limit m_i (in km/h). For instance, you can get from Odder to Hørning (via Skanderborg) with a car that travels at 100 km/h. Take a moment to convince yourself that you can get from Århus to Odder with the abovementioned Citroën.
To make your life easy, let's agree that there are no one-way streets: if a road joins a to b then it also joins b to a with the same distance and minimum speed. Also, city names contain no spaces, and each d_i and m_i is an integer.

(a) (*3 pt.*) The input is a roadmap description as above, the name of two cities f and t, and an integer s that gives the speed of your car. Design an algorithm that decides if you can get from f to t with your car.
In our example, if f is Odder and t is Århus and $s = 108$ then the answer is "true," but if $s = 106$ then the answer is "false."

(b) (*2 pt.*) Extend your algorithm from the previous question so that if the answer is "true," it also reports the minimum distance between f and t that your car needs to go. In our example, with $s = 108$, the answer is 50. With $s = 150$, the answer is 23.

(c) (*3 pt.*) The input is a board description as above. Design an algorithm that returns the speed of the slowest car that still gets you everywhere. (In the sense that, after you bought the car, then no matter which f and t you choose, you can get from f to t in that car. Possibly that trip requires a huge detour, but that's not important.) In our example, the answer is 107.

For all three questions, state the running time of your resulting algorithms in terms of the given parameters. You are strongly encouraged to make use of existing algorithms, models, or data structures from the book, but please be precise in your references (for example, use page numbers or full class names of constructions in the book). Be short and precise. Each question can be perfectly answered on half a page of text. (Even less, in fact.) If you find yourself writing much more than one page, you're using the wrong level of detail. However, it is a very good idea to include a drawing of a concrete (small) example. You don't need to write code. (However, some people have an easier time expressing themselves clearly by writing code. In that case, go ahead.) You are evaluated on correctness and efficiency of your solutions, and clarity of explanation.

Answers

This is where you mark your answers for questions 1–3. It is strongly preferred that you fit your answers on these sheets, except for question 4. If you really must, you can use a separate sheet of paper instead. Please indicate that clearly.

Your name:

	1a	1b	1c	1d	1e	1f	1g	2a	2c	2d	2e	2f	3a	3b	3d	3f
A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C
D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D

2b

2g

```
public int size() {

}
```

2h

```
public void prepare() {

}
```

2i

```
public Iterator<Item> iterator() {

}
```

3c

3e

	0	1	2	3	4	5	6
keys							
values							

3g

3h

input	$d = 2$	$d = 1$	$d = 0$
BOB	IDA		
SUE			
IKE			
IDA			

4a,b,c *On a separate piece of paper.*