Answers of Algorithms and Data Structures, Reexam 21 August 2018

The questions start on page 6.

It is strongly preferred that you fit your answers on these sheets. If you really must, you can use a separate sheet of paper instead. Please indicate that clearly.

Your name:

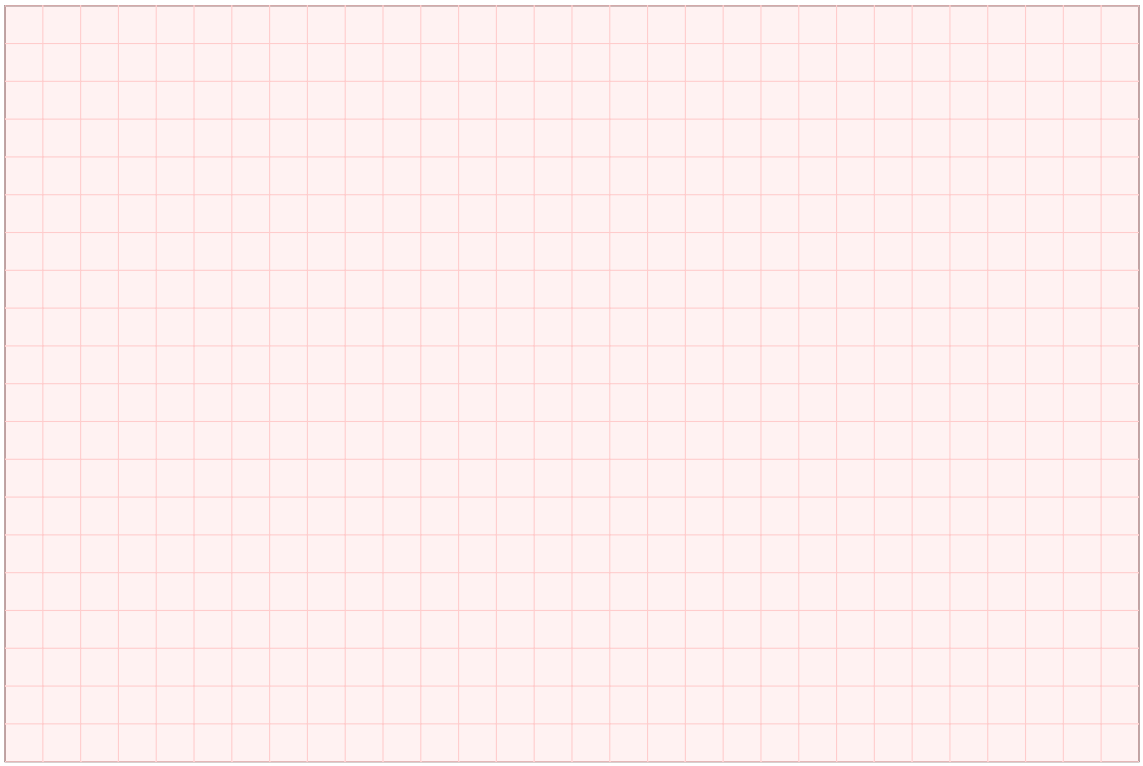
	1a	1b	1c	1d	1e	1f	1g	2b	2c	2d	3a	3b	3d	3f
A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
C	C	C	C	C	C	C	C	C	C	C	C	C	C	C
D	D	D	D	D	D	D	D	D	D	D	D	D	D	D

2a

2e

2f

2g

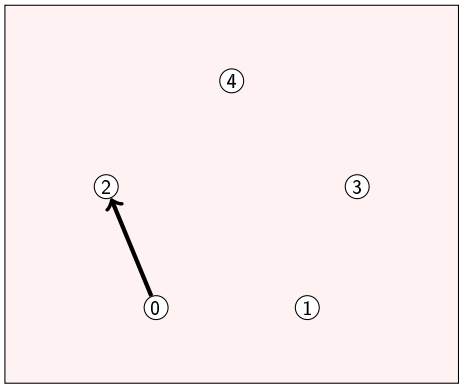


3c

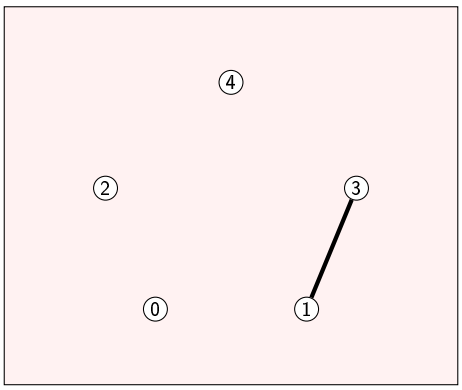
3e

3g

3h



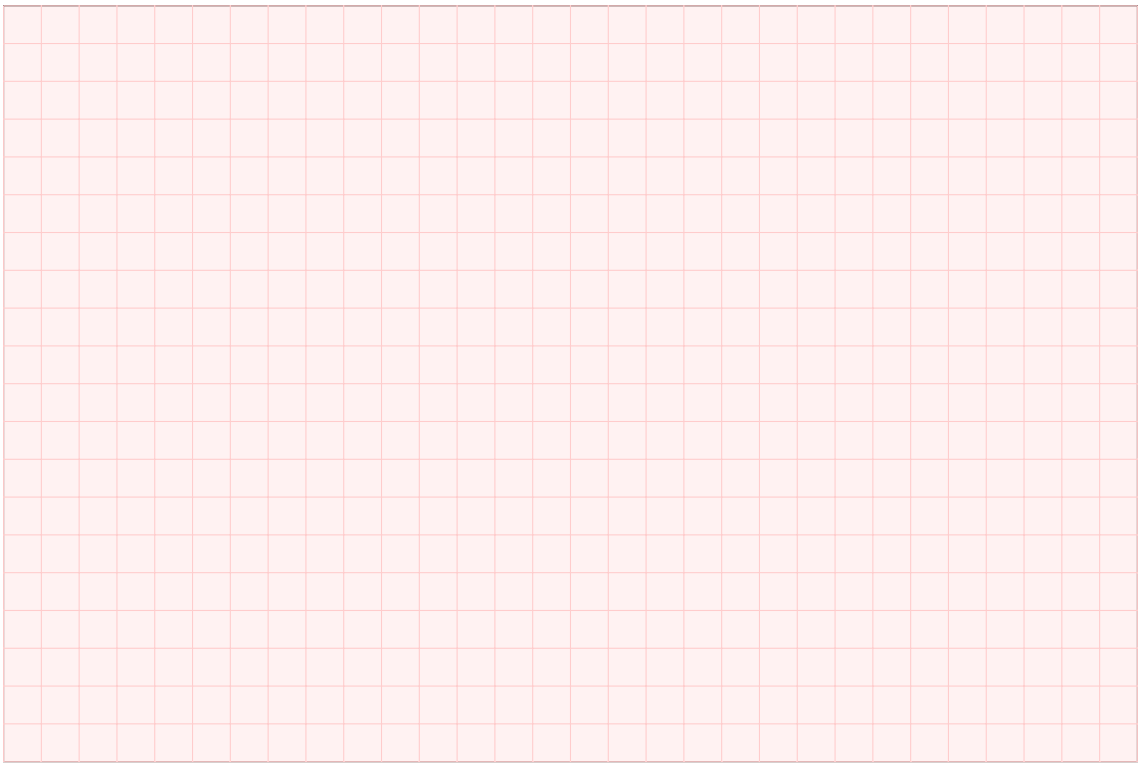
3i



4a

4b

4c



Algorithms and Data Structures

Reexam 21 August 2018

Thore Husfeldt and Riko Jacob, 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.)

You do not have access to electronic devices such as a mobile phone, a computer, or an e-book reader.

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 on pages 1–5. If you really have to, you may use separate sheets of paper for the free text questions instead, but please be clear about it (cross out everything and write "see separate paper, page 1" or something like that.) For the love of all that is Good and Holy, write legibly. Hand in pages 1–5, and any separate sheet(s) of paper. Do not hand in pages 6–12, 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 $f(N) = N$ and $g(N) = N + 2N^2$

B $f(N) = 2N$ and $g(N) = N + \sqrt{N}$

C $f(N) = N \log N$ and $g(N) = N \log N + 2N$

D $f(N) = 2\sqrt{N} + N$ and $g(N) = \sqrt{N} + 2N$

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

A $f(N) = 3N + \frac{1}{2}N \log N$ and $g(N) = 3N$

B $f(N) = (\log N)$ and $g(N) = \sqrt{N}$

C $f(N) = (N + 1) \cdot (N + 10)$ and $g(N) = 2N + 1$

D $f(N) = N^3$ and $g(N) = N^2 + 8N$

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

```
# python3
i = 1
while i < N:
    i = i+3
    stdio.write("*")
```

```
// java
for (int i = 1 ; i < N; i = i+3)
    StdOut.print("*");
```

A $\sim \log_3 N$ B $\sim N/3$ C $\sim N + 3$ D $\sim \frac{1}{3}N^3$

(d) (*1 pt.*) Assume f and g satisy $f(N) \sim aN$ and $g(N) = b \log_2 N$ for some positive integers a and b. Define h by $h(N) = f(N) + g(N)$. What is true for every choice of a and b?

A $h(N) \sim aN$ B $h(N) \sim b \log_2 N$ C $h(N) = O(\log N)$ D $h(N) \leq a + b$

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

```
# python3
i = 0
while i < N:
    i = i+1
    j = 0
    while j < N:
        A[i] = A[i] + A[j]
        j = j + 1
```

```
// java
for (int i = 0; i < N; i = i+1) {
    for (int j = 0; j < N; j = j+1)
        A[i] = A[i] + A[j];
}
```

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 arithmetic operations (additions and subtractions) performed by the following recursive function . (Choose the smallest correct estimate. The base case is $T(0) = 0$ in all cases.)

```
# python3
def r( N ):
    if N > 1:
       return r(N-1) + r(N-2)
    else:
       return 1
```

```
// java
static int r(int N)
{
  if (N > 1) return r(N-1) + r(N-2);
    else       return 1;
}
```

A $T(N) = T(N-1) + T(N-2) + 3$
B $T(N) = T(N-1) + T(N-2)$
C $T(N) = T(N-1) + T(N)$
D $T(N) = 2T(N-2) + 3$

(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

```
# python3
for i in range(N):
    f(i)
```

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

(Choose the smallest correct estimate.)

A Linearithmic in N.

B Linear in N.

C Quadratic in N.

D Impossible to say from the information given.

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

public class C
{
    public static void c(String line)
    {
        int n = line.length();
        for (int i = 0; i<n; i++)
        {
            boolean found = false;
            for (int j = i+1; j<n; j++)
                if (line.charAt(i) == line.charAt(j))
                    found = true;
            if (found) StdOut.print(line.charAt(i));
        }
    }

    public static void main(String[] args)
    {
        c("THEQUICKBROWNFOXJUMPSOVERTHELAZYDOG");
    }
}
```

Figure 1: Function c, java version.

```python
# python3

def c(line):
    n = len(line)
    for i in range(n):
        found = False
        for j in range(i+1,n):
            if line[j] == line[i]:
                found = True
        if found:
            print (line[i])

c("THEQUICKBROWNFOXJUMPSOVERTHELAZYDOG")
```

Figure 2: Function c, python version.

2. Function c. The next few questions all concern the code in fig. 1 and 2.

(a) (*1 pt.*) Write the output of running the code. (Just write the letters in order, ignoring spaces and newlines.)

(b) (*1 pt.*) Professor Precipitat makes the following claim about function `c`:
"When given a string of letters, a letter is printed only if it appears exactly twice in the string."
Fix the good professor's explanation:

A both occurrences of 'string' should be 'set'

B 'only if' should be 'unless'.

C 'exactly' should be 'at least'.

D 'a string of letters' should be 'two strings of letters'.

(c) (*1 pt.*) What happens if I run `c` on the empty string, i.e., if I call `c("")`?

A A runtime error.

B Nothing is printed.

C 'false' (java) or 'False' (python) is printed.

D 'null' is printed.

(d) (*1 pt.*) What is the running time on input of length N?

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

(e) (*1 pt.*) Rewrite the body of function `c` (completely, producing correct code) so that it prints the letters that appear exactly once in the string. (The ordering is not important.)

(f) (*2 pt.*) Describe a faster implementation of `c` with the same functionality as in fig. 1 or 2. (You may assume that the input is a string of upper case letters of the English alphabet.) You don't have to write correct code (but you may if you want.) Be brief. State the running time. Faster is better.

(g) (*2 pt.*) Describe how to implement a function that returns exactly the letters in the given string, each letter once, ordered by their number of occurrences. For instance, given the string

```
THEQUICKBROWNFOXJUMPSOVERTHELAZYDOG
```

the output could be

```
OETHURQICKBWNFXJMPSVLAZYDG
```

Ties can be broken arbitrarily. In the example, both `Y` and `Z` appear exactly once, so their ordering in the output is unspecified. You may assume that the input is a string of upper case letters of the English alphabet. You don't have to write correct code (but you may if you want.) Be brief. State the running time. Faster is better.

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.

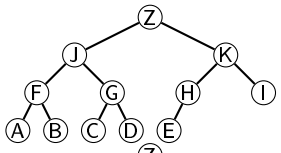
5 3 9 $*$

What is the data structure if the removed elements is 3 ?

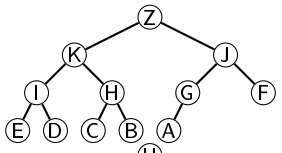
A Priority queue B Stack C Queue D Trie

(b) (*1 pt.*) Which of the following trees is a search tree?

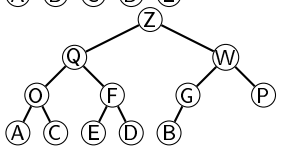
A



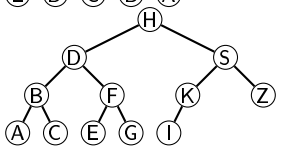
B



C



D



(c) (*1 pt.*) Insert the keys `5 4 3 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 `C B D A` into alphabetically increasing order using an unknown sorting algorithm. At some time during the process, the letters are arranged like this: `A B D C`. Which algorithm could I be using?

A Mergesort. B Quicksort. C Insertion sort. D Selection sort.

(e) (*1 pt.*) In the style of the book, draw the trie for the following key–value pairs:

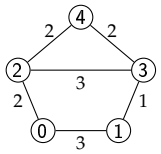
key	value
ALBA	4
ALGAE	3
ALGO	2
AL	1
A	0

(f) (*1 pt.*) Consider a connected, unweighted, undirected graph. Let V be the number of vertices and E be the number of edges. Which claim is always true?

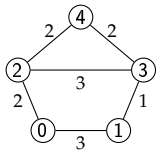
A There is a path between any pair of vertices. B $E = V - 1$.
C The graph contains a simple cycle. D Any BFS and DFS trees are identical.

(g) (*1 pt.*) Insert the letters `D E B A C L` into a 2–3 search tree in that order. Draw the resulting structure in the style of the book.

(h) (*1 pt.*) Draw a shortest path tree from vertex 0 for



(i) (*1 pt.*) Draw a minimum spanning tree for

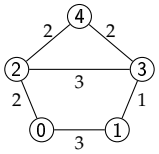


4. Design of algorithms

In graph theory, the *diameter* is a number associated with a connected, undirected graph with positive integer edge-weights and is defined as follows. For vertices u and v in G, let $d(u, v)$ denote the distance between u and v. (Recall that the distance between u and v is the total length of a shortest path connecting u and v.) Then the *diameter* of G is the largest distance between any pair of vertices in the graph, in other words,

$$\operatorname{diam}(G) = \max_{u,v} d(u, v)\,,$$

where u and v range over all vertices in G.

For example, the graph 

has diameter 4.

(a) (*3 pt.*) Describe an algorithm that computes the diameter of a given graph. You may assume the graph is given as an edge-weighted graph, as described in section 4.3 of the text book.

(b) (*1 pt.*) Assume the edge weights are unity (i.e., all equal to the integer 1). Describe an asymptotically faster algorithm.

(c) (*2 pt.*) (Harder.) Assume the graph is a tree. Describe a linear-time algorithm. (You don't need to prove correctness.)

For these questions, state the running time of your resulting algorithms in terms of the number V or vertices and E of edges. 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, as well as clarity of explanation.