Algorithms and Data Structures (BADS)

Tentative, fake exam, never actually used

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.)

You can't bring electronic aids (such as a laptop) or communication devices (such as a mobile phone). If you really want, you can bring an old-fashioned pocket calculator (not one that solves recurrence relations), but I can't see how that would be of any use to you.

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	.5	0.21	0
points if correct answer not checked	0	-0.33	-.5	-0.74	

In particular, the best thing you can do 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. For more details, read [Gudmund Skovbjerg Frandsen, Michael I. Schwartzbach: A singular choice for multiple choice. SIGCSE Bulletin 38(4): 34–38 (2006)].

Typographic remark. We follow the typographic convention used impliclty in the course book that a one-letter Java variable, such as `N`, is typeset in italics like a mathematical variable in body text: N.

Base revision e60a3d8. . ., Wed Apr 12 11:31:24 2017 +0200, Riko Jacob (@ITU Air 13).

Analysis of algorithms

Comment: Questions in this part are about analysis of algorithms. I include a lot of them below, so that you can see which types to expect. I need to turn them all into Java, instead of pseudocode.

1.

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

A $(x^4+1)(x+x^3)$ and x^7

B $(x \log_2 x)(x+7)$ and x^2

C $8x^2$ and $x!$

D 2^x and 4^x

(b) (*1 pt.*) Which pair of values, C and k, are witnesses that show that $3x+26$ is $O(x)$?

A $C = 3, k = 26$

B $C = 3, k = 1000$

C $C = 4, k = 27$

D $C = 5, k = 3$

(c) (*1 pt.*) Find a recurrence relation for the number of comparisons performed by the following recursive method:

```
static int f(int N)
{
    if (N > 3) return f(N - 1) * f(N - 3);
    else return 3;
}
```

(Choose the smallest correct estimate.)

A $T(N) = T(N-1) + T(N-3) + 1$

B $T(N) = T(N-1) \cdot T(N-3) + 1$

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

D $T(N) = T(N-1) + N$

(d) (*1 pt.*) Assume you have a data structure that maintains a set `S` under insertion. Assume that the operation "`S.insert(key)`" takes linear time in the number of elements in `S` if `key` does not belong to `S`, and constant time if `key` does belong to `S`.
Determine the order of growth of the running time of the following piece of code, starting with an empty set `S`.

```
for (int i = 0; i < N ; i++)
    for (int j = 0; j < N ; j++)
        S.insert(j);
```

A Linearithmic in `N`

B Quadradtic in `N`

C $\sim aN^2 \log N$ for some constant a

D Cubic in `N`

(e) (*1 pt.*) Assume I want the code from the previous exercise to use $\sim N^2$ additions. What bounds on the running time of **insert** would guarantee this result?

A `insert` performs zero additions.

B `insert` performs 1 addition.

C `S.insert(key)` performs $\sim \log M$ additions, where M is the number of elements in S.

Larger problem

Comment: A question of this type considers a slightly larger (new) data structure. Typically, this is a (not very clever) implementation of a well-known data type.

Consider the following data structure.

```
class X
{
    int cap = 10;
    Object[] A = new Object[cap];
    int x = cap;

    public void insert(Object in)
    {
        x = x - 1;
        if ( x == -1 ) rebuild();
        A[x] = in;
    }

    public Object remove()
    {
        x = x + 1;
        return A[x-1];
    }


    private void rebuild()
    {
        cap = 2 * cap;
        Object[] tmp = new Object[cap];
        for (int i = 0; i < cap/2; i = i + 1)
            tmp[i + cap/2] = A[i];
        A = tmp;
    }
}
```

Let us agree that N denotes the number of elements in the data structure.

2.

(a) (*1 pt.*) Which well-known data type does `X` implement?

- A Stack.
- B Queue.
- C Priority queue.
- D Union–Find.

(b) (*1 pt.*) Write the body of a method `int size()` that returns the number of elements in the data structure.

- A `return N;`
- B `return cap;`
- C `return x;`
- D `return cap-x;`

(c) (*1 pt.*) How many array accesses does a single call to a method of `X` take in the worst case?

- A $\sim 6N$.
- B 2.
- C $\sim 4N$.
- D 1.

(d) (*2 pt.*) What is the amortized number of array accesses per operation in a worst case sequence of k operations beginning in an empty data structure?

- A linear in k.
- B constant.
- C linearithmic in k.
- D quadratic in k.

Operation of common data structures

Comment: Questions in this part check your understanding of how the standard data structures work, both "from the outside" (as an abstract data type) and "from the inside". There are many of these types of questions in the book, where they are called "Exercises". Here are some more, relating to a previous edition of the book, but it should be clear enough. With the current book, I wouldn't pose the question about inorder traversal (p. 318), because it doesn't make a big deal out of that concept. Also, the illustrations in the book are slightly different, and I'll try to be as consistent as possible.

3.

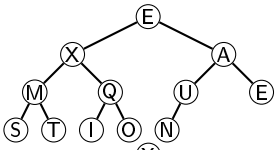
(a) (*1 pt.*) Consider a stack. A letter *i* means "insert *i*" and "$*$" means "pop".

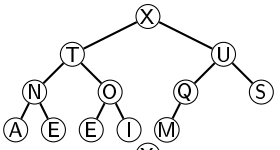
E * X A * * M Q * U E S * T * I O N

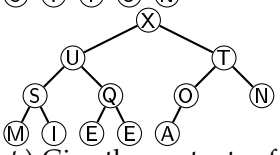
Give the sequence of values popped by these operations.

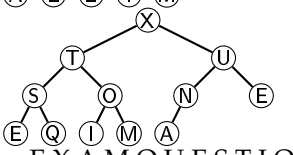
A E X A M Q U E S T I O N
B N O I T S E U Q M A X E
C E A X Q S T
D E A X Q M S E U T N O I

(b) (*1 pt.*) Assume keys E X A M Q U E S T I O N are inserted in that order into an initially empty heap-based priority queue ([SW] p. 218f), using the `insert` method. What is the resulting data structure?

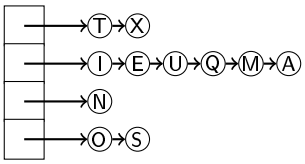
A 

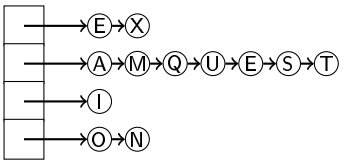
B 

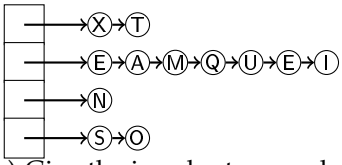
C 

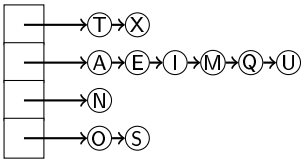
D 

(c) (*1 pt.*) Give the contents of a hash table where the keys E X A M Q U E S T I O N are inserted in that order into an initially empty table of size $M = 4$ using separate chaining (p. 369). Use the hash value $k \bmod M$ to transform the kth letter of the alphabet (A $= 1$) into a table index 0, 1, 2, 3.

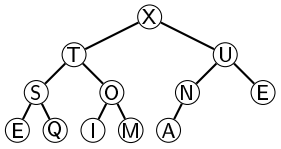
A 

B 

C 

D 

(d) (*1 pt.*) Give the inorder traversal of this tree:



A E X A M Q U E S T I O N
B E S Q T I O M X A N U E
C A E E I M N O Q S T U X
D E S Q T O I M X U N A E

Design of algorithms

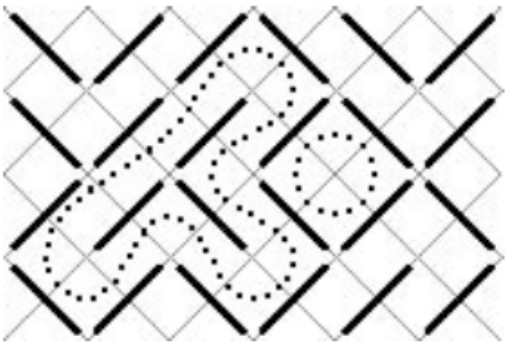
Comments: In this part, you have to design a solution to a given problem, typically be choosing among the various methods presented in the course. Questions of this type are free-form (so, no multiple choice). Most students consider them vastly more difficult. They correspond to some of the book's "Creative problems". For the love of all that is Good and Holy, please be as short and concrete as possible when answering a question of this type. I cannot stress how important a small example is – mainly to clarify to yourself, not me, what you actually mean.

4. This exercise asks you to suggest a solution to Slash Maze, which you can find on the next page. Don't write a full programme! Write your answer on a separate piece of paper; use English or Danish; be brief and precise (do not use more than one page, preferably less). One concrete, complete, well-chosen example is often the best way of explaining yourself. If you want, you can use some form of code or pseudo-code, but you don't have to.

(a) (*5 pt.*) Explain how you would solve this problem in an efficient way. (Polynomial time will be fine.) Do you use a data structure or an algorithm? Which ones? What is the resulting running time and space? Express yourself in terms of the parameters of the problem (e.g., w, h, k, l), instead of just saying things like "runs in $\sim aN$ for some a" without telling me what N is.

Slash Maze

By filling a rectangle with slashes (/) and backslashes (\), you can generate nice little mazes. An example is to the right. As you can see, paths in the maze cannot branch, so the whole maze contains only (1) cyclic paths and (2) paths entering somewhere and leaving somewhere else. We are only interested in the cycles. There are exactly two of them in our example. Your task is to write a program that counts the cycles and finds the length of the longest one. The length is defined as the number of small squares the cycle consists of (the ones bordered by gray lines in the picture). In this example, the long cycle has length 16 and the short one length 4.



Input

Each description begins with one line containing two integers w and h ($1 \le w, h \le 75$), representing the width and the height of the maze. The next h lines describe the maze itself and contain w characters each; all of these characters will be either "/" or "\".

Output

Output the line "k Cycles; longest has length l.", where k is the number of cycles in the maze and l the length of the longest of the cycles. If the maze is acyclic, output "There are no cycles." .

Example 1	Example 2

Sample Input
`6 4`
`\//\\/`
`\///\/`
`//\\/\`
`\/\///`
	Sample Input
	`3 3`
	`///`
	`\//`
	`\\\`

Sample output `2 Cycles; longest has length 16.`	Sample output `There are no cycles.`