Algorithms and Data Structures (BADS)

Exam 29 September 2011

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 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)].

(Just to make sure: a question that is not multiple-choice cannot give you negative points.)

Typographic remark. I 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 04e2f12. . ., Tue Jan 24 16:54:59 2017 +0100, Thore Husfeldt.

Analysis of algorithms

1.

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

[A] $(N^2+1)(N^2+N)$ and N^4

[B] $(N \log_2 N)(N+7)$ and N^2

[C] $8N^2+3N$ and $3N^2+8N$

[D] 2^N and 2^{N+1}

(b) (*1 pt.*) Find a recurrence relation for the number of arithmetic operations (subtractions, additions, multiplications, divisions) performed by the following recursive method:

```
static int f(int N)
{
    if (N > 1) return f(N - 1) + N - 2;
    else return 3;
}
```

(Choose the smallest correct estimate.)

[A] $T(N) = T(N-1) + T(N-2)$

[B] $T(N) = T(N-1) + N - 2$

[C] $T(N) = T(N-1) + 2$

[D] $T(N) = T(N-1) + 3$

Class Y

The next few questions all concern the class defined in fig. 1.

2.

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

[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 A.length;`

[C] `return A[N];`

[D] `return hi - lo;`

(c) (*1 pt.*) Which invariant does the data structure maintain after every public operation?

[A] `N < A.length`

[B] `lo < hi`

[C] `hi < N`

[D] `hi == N`

(d) (*1 pt.*) Draw the data structure (including the contents of `A` and the values `hi`, `lo`, and `N`) after the following operations:

```
Y y = new Y();
y.insert(1);
y.remove();
y.insert(2);
y.remove();
y.insert(3);
```

your drawing goes in the shaded region

(e) (*1 pt.*) How many array accesses does a single call to `Y.remove` take in the worst case? (To make this well-defined we assume that the compiler performs no clever optimisations. That is, every array access we've written in the code will actually be performed.)

```
public class Y<Key extends Comparable<Key>>
{
    private Key[] A = (Key[]) new Comparable[1];
    private int lo, hi, N;

    public void insert(Key in)
    {
        A[hi] = in;
        hi = hi + 1;
        if (hi == A.length) hi = 0;
        N = N + 1;
        if (N == A.length) rebuild();
    }

    public Key remove() // assumes Y is not empty
    {
      Key out = A[lo];
      A[lo] = null;
      lo = lo + 1;
      if (lo == A.length) lo = 0;
      N = N - 1;
      return out;
    }

    private void rebuild()
    {
       Key[] tmp = (Key[]) new Comparable[2*A.length];
      for (int i = 0; i < N; i++ )
        tmp[i] = A[(i + lo)  % A.length];
      A = tmp;
      lo = 0;
      hi = N;
     }
}
```

Figure 1: Class Y.

- [A] $\sim 4N$. [B] 2.
- [C] $\sim 2N$. [D] 7.

(f) (*1 pt.*) How many array accesses does a single call to the most expensive public method of `Y` take in the worst case?

- [A] $\sim 4N$. [B] 2.
- [C] $\sim 2N$. [D] 7.

(g) (*1 pt.*) What is the amortized number of array accesses per operation in a sequence of k `Y.insert` operations beginning in an empty data structure?

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

(h) (*1 pt.*) What is the number of array accesses per operation in the following sequence of $2k$ operations, starting from an empty data structure: `y.insert(1); y.remove(); y.insert(2); y.remove(); y.insert(3); y.remove();` $\cdots$ `y.insert(k); y.remove();`

- [A] linear in k in the worst case and in the amortized case.
- [B] constant in the worst case.
- [C] constant in the amortized case, but linear in k in the worst case.
- [D] quadratic in k in the worst case.

(i) (*3 pt.*) True of false: The data structure `Y` uses space linear in `N`. Explain your answer on a separate piece of paper. (Be as formal *and short* as you can, but not shorter. If you use more than half a page of text you're on the wrong level of abstraction.)

Operation of common data structures

3.

(a) (*1 pt.*) Consider the following sequence of operations on symbol table:

```
put(2,5)  put(3,1)  get(2)  get(2)  get(3)  put(2,10)  get(2)
```

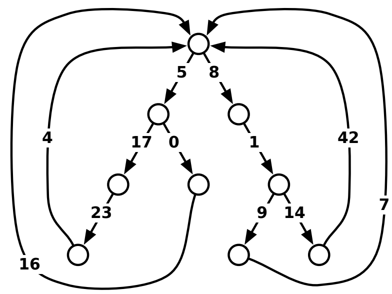
What sequence of values is returned by the `get`-operations? (Your answer goes into the shaded box:)

(b) (*1 pt.*) I have sorted the word "how much wood would a woodchuck chuck if a woodchuck would chuck wood?" using MSD string sort ([SW, section 5.1]). In the figure below, mark all characters that were examined by the MSD string sort with a circle.

```
a
a
chuck
chuck
how
if
much
wood
wood
woodchuck
woodchuck
would
would
```

Design of algorithms

4. A *looped tree* is a weighted, directed graph built from a binary tree by adding an edge from every leaf back to the root. Every edge has a non-negative weight.



A looped tree.

(a) (*0 pt.*) Convince yourself that the shortest path from the leftmost leaf to the rightmost leaf in the above example graph has length 27.

(b) (*1 pt.*) Given such a graph and two vertices, u and v, we want to find the shortest path from u to v. Clearly Dijkstra's algorithm ([SW, Algorithm 4.9]) solves this problem. What is the running time in terms of V?

(c) (*5 pt.*) Explain how you would solve this problem faster. If you want, you can make use of existing algorithms, models, or data structures from the book, please be precise in your references (for example). Estimate the running time of your construction. Be short and precise. This question can be perfectly answered on half a page of text, maybe less. If you find yourself writing more than one page, you're using the wrong level of detail.