Algorithms and Data Structures (BADS/SGDS)

Exam 19 August 2015

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. Do not hand in pages 1–6. (You are welcome to mark pages 1–6 any way you want, including checking boxes. But you can not hand these pages in. They won't count.) Instead, mark your answers on pages 7 and 8, and separate sheets of paper, as instructed. For the love of all that is Good and Holy, write legibly.

Base revision 701dfb8..., Sun May 28 18:03:01 2017 +0200, Riko Jacob (@ITU Air 13).

Exam questions

1. Analysis of algorithms

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

A $f(N) = N + 6N$ and $g(N) = 6N$

B $f(N) = (N+1) + (N+2) + (N+3)$ and $g(N) = 3N$

C $f(N) = N + N^2 + N^3$ and $g(N) = 6N^3$

D $f(N) = \log N + \log 2N + \log 3N$ and $g(N) = \log 3N$

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

A $f(N) = N + N + N$ and $g(N) = 3 \log N$

B $f(N) = (N+1) \cdot (N+1) \cdot (N+1)$ and $g(N) = N^3$

C $f(N) = (\log N) \cdot (\log N)$ and $g(N) = \log N$

D $f(N) = N^3$ and $g(N) = 300N$

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

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

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

(d) (*1 pt.*) How many stars are printed when I call $f(N)$? (Choose the smallest correct estimate.)

```
static void f(int K)
{  for (int i = 0; i < K; i++) g(i); }

static void g(int R)
{  for (int j = 0; j < R; j++) StdOut.print("*"); }
```

Choose the smallest correct estimate.

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?

```
if (N < 1000)
  for (int i = 0; i < N; i++) A[i] = N*N*N;
else if (N < 10000)
  for (int i = 0; i < N*N; i++) A[i] = N*N;
else
  for (int i = 0; i < N*N*N; i++) A[i] = N*N;
```

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:

```
static int r(int N)
{
    if (N > 2) return r(N-2) + r(N-1);
    if (N == 0) return 1;
}
```

(Choose the smallest correct estimate.)

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

(g) (*1 pt.*) For the next 4 questions, consider the following sequence of operations

```
A a = new A(); for (int i = 0; i < N; i++) a.f();
```

on the class A defined in Figure 1.

```
class A
{
  int max = 1;
  int count = 0;

  void f()
  {
    count++;
    if (count == max)
    {
      for (int i = 0; i < max; i++) StdOut.print("*");
      max = 2 * max;
    }
}
```

Figure 1: Class `A`.

How many stars are printed by the last call to `a.f()` in the sequence in the worst case?

A Constant. B Logarithmic in N. C Linear in N. D Quadratic in N.

(h) (*1 pt.*) How many stars are printed by the last call to `a.f()` in the sequence in the best case?

A Constant. B Logarithmic in N. C Linear in N. D Quadratic in N.

(i) (*1 pt.*) How many stars are printed by the sequence *in total*?

A Constant. B Logarithmic in N. C Linear in N. D Quadratic in N.

(j) (*1 pt.*) How many stars are printed by a single call to the method `f` in the amortized sense?

A Constant. B Logarithmic in N. C Linear in N. D Quadratic in N.

2. Method f. The next few questions all concern the method `f` defined in Figure 2.

(a) (*1 pt.*) Let `int[] A = {2,4,4,5}`. For which `B` does `f(A,B)` return `true`?

A `int[] B = {5,4,2}`

B `int[] B = {2,4}`

C `int[] B = {2,4,4}`

D `int[] B = {}`

```
public static boolean f(int[] A, int[] B)
  {
    for (int i= 0; i < A.length; i++)
      {
        boolean found = false;
        for (int j = 0; j < B.length; j++)
            if (A[i] == B[j]) found = true;
        if (!found) return false;
      }
    return true;
  }
}
```

Figure 2: Method `f`.

(b) (*1 pt.*) Assume `int[] A = {1,2,3}`. Give an example of `int[] B` for which `f(A,B)` returns `false`.

(c) (*2 pt.*) What does `f` decide?

[A] Whether every value in `A` also appears in `B`.

[B] Whether `A` and `B` contain the same values.

[C] Whether `A` and `B` are identical.

[D] Whether `A` and `B` have the same length.

(d) (*1 pt.*) Let `N = A.length` and `M = B.length`. Give the asymptotic worst-case running time of `f` in terms of `N` and `M`.

(e) (*2 pt.*) Rewrite `f` to be asymptotically faster. Explain your solution in prose, pseudocode, drawing, or code, as you see fit. Be brief. You are welcome to use other code from the book. State the running time of your solution; faster is better.

(f) (*2 pt.*) Write a method `static boolean g(int[] A, int[] B)` that decides if each value in `A` also appears in `B`, *at least the same number of times*. So if `int[] A = {1,5,5}` and `int[] B = {1,5}` then `g(A,B)` should be `false`, but if we set `int[] C = {5,1,5,2,5}` then `g(A,C)` should be `true`. Explain your solution in prose, pseudocode, drawing, or code, as you see fit. Be brief. You are welcome to use other code from the book. State the running time; faster is better.

(g) (*2 pt.*) Write a method `static boolean g(int[] A, int[] B)` that decides if every value in `A` is strictly smaller than every value in `B`. So if `int[] A = {1,5,5}` and `int[] B = {6,5}` then `g(A,B)` should be `false`, but for `int[] C = {7,6}` then `g(A,C)` should be `true`. Explain your solution in prose, pseudocode, drawing, or code, as you see fit. Be brief. You are welcome to use other code from the book. State the running time; faster is better.

3. Operation of common algorithms and data structures.

(a) (*1 pt.*) Insert the keys `1 3 6 2 4 5` in that order into a heap. Draw the result.

(b) (*1 pt.*) Assume I sort the letters `A L G O R I S T` using merge sort. Which of the following situations *cannot* arise at any time during the algorithm?

[A] `A L G O R I S T` [B] `A G I L O R S T` [C] `A G L O R I S T` [D] `A I L G O R S T`

(c) (*1 pt.*) Run (i) insertion sort and (ii) selection sort on the 8-letter input `S E Q U E N C E`. Stop each algorithm after exactly 3 calls to `exch`. Write the resulting sequences.

(d) (*1 pt.*) In which sense is quicksort better than insertion sort?

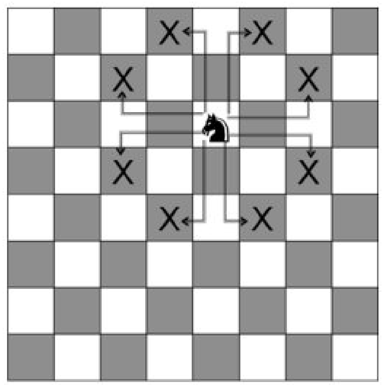
- [A] It has better asympotic worst-case performance
- [B] It is faster on random input.
- [C] It is faster on sorted input.
- [D] It uses less space.

(e) (*1 pt.*) Insert the letters `E X A M` in that order into a 2–3 search tree as defined in the textbook (section 3.3). Draw the resulting structure in the style of the book.

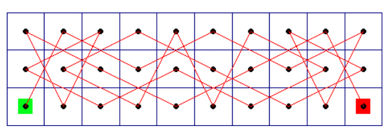
4. Design of algorithms. The *knight* in the board game *Chess* moves

- two squares horizontally and one square vertically, or
- two squares vertically and one square horizontally

On the 8×8 board below, the knight can reach each of the positions marked 'X' in one move:



Let's use integer coordinates for the knight, with $(0,0)$ at the bottom left. In the above image, the knight is at $(4,5)$. A *knight's tour* is a sequence of moves made by the knight. Here is a pretty impressive knight's tour on a 10×3 board from $(0,0)$ to $(9,0)$:



(a) (*2 pt.*) Write a method

```
static boolean valid(int N, int M, int x1, int y1, int x2, int y2)
```

that checks if the knight can go *in one move* from position (x_1, y_1) to position (x_2, y_2) on an $N \times M$ board. For instance `valid(8,8,4,5,5,7)` is `true` but `valid(8,8,4,5,5,6)` is `false`. Write complete and correct Java. State the running time.

(b) (*3 pt.*) Describe how to write a method

```
static boolean reachable(int N, int M, int x1, int y1, int x2, int y2)
```

that checks if there is a knight's tour from (x_1, y_1) to position (x_2, y_2) on an $N \times M$ board. For instance `valid(10,3,0,0,9,0)` is `true` but `valid(2,3,0,0,1,1)` is `false`.
State the running time; faster is better.
You are 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

Yes, in normal chess you use $1, 2, 3, \ldots$ for one direction and A,B,C,... for the other dimension. But that convention would make it slightly harder to write code.

Make sure to tell me what your variables mean. So don't just write $O(K)$ if you haven't defined K

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

Your name:

	1a	1b	1c	1d	1e	1f	1g	1h	1i	1j	2a	2c	3b	3d
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

2b

```
int[] B = {
```

2d

2e *On a separate piece of paper.*

2f *On a separate piece of paper.*

2g *On a separate piece of paper.*

3a

3c

Insertion sort:

Selection sort:

3e

4a *On a separate piece of paper.*

4b *On a separate piece of paper.*

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