UNIVERSITY OF
CAMBRIDGE

Computer Laboratory

Algorithms I — Exercises for students

Academic year 2012–2013

Lent–Easter term 2013

http://www.cl.cam.ac.uk/Teaching/1213/AlgorithI/

frank.stajano--algs1@cl.cam.ac.uk

(course-related emails sent to *this* address will be given higher priority)

Revised 2013 edition

Revision 13 of 2013-02-03 20:27:30 +0000 (Sun, 03 Feb 2013).

Exercises from the course handout

Your supervisor may also point you at suitable sections of past exam questions. Past exam questions are available from the course web site.

Exercise 1
Assume that each `swap(x, y)` means three assignments (namely `tmp = x; x = y; y = tmp`). Improve the insertsort algorithm pseudocode shown in the handout to reduce the number of assignments performed in the inner loop.

Exercise 2
Provide a useful invariant for the inner loop of insertion sort, in the form of an assertion to be inserted between the "while" line and the "swap" line.

Exercise 3

$$
\begin{aligned}
|sin(n)| &= O(1) \\
|sin(n)| &\neq \Theta(1) \\
200 + sin(n) &= \Theta(1) \\
123456n + 654321 &= \Theta(n) \\
2n - 7 &= O(17n^2) \\
\lg(n) &= O(n) \\
\lg(n) &\neq \Theta(n) \\
n^{100} &= O(2^n) \\
1 + 100/n &= \Theta(1)
\end{aligned}
$$

For each of the above "=" lines, identify the constants k, k_1, k_2, N as appropriate. For each of the "$\neq$" lines, show they can't possibly exist.

Exercise 4
What is the asymptotic complexity of the variant of insertsort that does fewer swaps?

Exercise 5
The proof of Assertion 1 (lower bound on exchanges) convinces us that $\Theta(n)$ exchanges are always *sufficient*. But why isn't that argument good enough to prove that they are also *needed*?

Exercise 6
When looking for the minimum of m items, every time one of the $m-1$ comparisons fails the best-so-far minimum must be updated. Give a permutation of the numbers from 1 to 7 that, if fed to the Selection sort algorithm, maximizes the number of times that the above-mentioned comparison fails.

Exercise 7
Code up the details of the binary partitioning portion of the binary insertion sort algorithm.

Exercise 8
Prove that Bubble sort will never have to perform more than n passes of the outer loop.

Exercise 9
Can you spot any problems with the suggestion of replacing the line that assigns to `a3[i3]` with the more explicit and obvious `a3[i3] = min(a1[i1], a2[i2])`? What would be your preferred way of solving such problems? If you prefer to leave that line as it is, how would you implement the procedure `smallest` it calls? What are the trade-offs between your chosen method and any alternatives?

Exercise 10
In one line we return the same array we received from the caller, while in another we return a new array created within the mergesort subroutine. This asymmetry is suspicious. Discuss potential problems.

Exercise 11
Never mind the theoretical computer scientists, but how do you mergesort in $n/2$ space?

Exercise 12
Justify that the merging procedure just described will not overwrite any of the elements in the second half.

Exercise 13
Write pseudocode for the bottom-up mergesort.

Exercise 14
Can picking the pivot at random *really* make any difference to the expected performance? How will it affect the average case? The worst case? Discuss.

Exercise 15
Justify why running Insertion sort over the messy array produced by the truncated Quicksort might not be as stupid as it may sound at first. How should the threshold be chosen?

Exercise 16
What is the smallest number of pairwise comparisons you need to perform to find the smallest of n items?

Exercise 17
(More challenging.) And to find the *second* smallest?

Exercise 18
What are the minimum and maximum number of elements in a heap of height h?

Exercise 19
For each of the sorting algorithms seen in this course, establish whether it is stable or not.

Exercise 20
Give detailed pseudocode for the counting sort algorithm (particularly the second phase), ensuring that the overall cost stays linear. Do you need to perform any kind of precomputation of auxiliary values?

Exercise 21
Why couldn't we simply use counting sort in the first place, since the keys are integers in a known range?

Exercise 22
Leaving aside for brevity Fibonacci's original 1202 problem on the sexual activities of a pair of rabbits, the Fibonacci sequence may be more abstractly defined as follows:

$$\begin{cases} F_0 = 1 \\ F_1 = 1 \\ F_n = F_{n-2} + F_{n-1} \quad \text{for } n \geq 2 \end{cases}$$

(This yields $1, 1, 2, 3, 5, 8, 13, 21, \ldots$)
In a couple of lines in your favourite programming language, write a *recursive* program to compute F_n given n, using the definition above. And now, finally, the question: how many function calls will your recursive program perform to compute F_{10}, F_{20} and F_{30}? First, guess; then instrument your program to tell you the actual answer.

Exercise 23
Prove (an example is sufficient) that the order in which the multiplications are performed may dramatically affect the total number of scalar multiplications—despite the fact that, since matrix multiplication is associative, the final matrix stays the same.

Exercise 24
Provide a small counterexample that proves that the greedy strategy of choosing the item with the highest £/kg ratio is not guaranteed to yield the optimal solution.

Exercise 25
Draw the memory layout of these two representations for a 3×5 matrix, pointing out where element (1,2) would be in each case.

Exercise 26
Show how to declare a variable of type list in the C case and then in the Java case. Show how to represent the empty list in the Java case. Check that this value (empty list) can be assigned to the variable you declared earlier.

Exercise 27
As a programmer, do you notice any uncomfortable issues with your Java definition of a list? *(Requires some thought and O-O flair.)*

Exercise 28
Draw a picture of the compact representation of a list described in the notes.

Exercise 29
Invent (or should I say "rediscover"?) a linear-time algorithm to convert an infix expression such as
`(3+12)*4 - 2`
into a postfix one without parentheses such as
`3 12 + 4 * 2 -`.
By the way, would the reverse exercise have been easier or harder?

Exercise 30
How would you deal efficiently with the case in which the keys are English words? *(There are several possible schemes of various complexity that would all make acceptable answers provided you justified your solution.)*

Exercise 31
Should the new key-value pair added by `set()` be added at the start or the end of the list? Or elsewhere?

Exercise 32
Solve the recurrence, again with the trick of setting $n = 2^m$.

Exercise 33
(Clever challenge, straight from CLRS3—exercise 12.2-4.) Professor Bunyan thinks he has discovered a remarkable property of binary search trees. Suppose that the search for key k in a binary search tree ends up in a leaf. Consider three sets: A, the keys to the left of the search path; B, the keys on the search path; and C, the keys to the right of the search path. Professor Bunyan claims that any three keys $a \in A$, $b \in B$, and $c \in C$ must satisfy $a \leq b \leq c$. Give a smallest possible counterexample to the professor's claim.

Exercise 34
Why, in BSTs, does this up-and-right business find the successor? Can you sketch a proof?

Exercise 35
(Important.) Prove that, in a binary search tree, if node n has two children, then its successor has no left child.

Exercise 36
Prove that this deletion procedure, when applied to a valid binary search tree, always returns a valid binary search tree.

Exercise 37
What are the smallest and largest possible number of nodes of a red-black tree of height h?

Exercise 38
For each of the three possible types of 2-3-4 nodes, draw an isomorphic "node cluster" made of 1, 2 or 3 red-black nodes. The node clusters you produce must:

- Have the same number of keys, incoming links and outgoing links as the corresponding 2-3-4 nodes. as the corresponding 2-3-4 nodes.
- Respect all the red-black rules when composed with other node clusters.

Exercise 39
(The following is not hard but it will take somewhat more than five minutes.) Using a soft pencil, a large piece of paper and an eraser, draw a B-tree with $t = 2$, initially empty, and insert into it the following values in order:

$$63, 16, 51, 77, 61, 43, 57, 12, 44, 72, 45, 34, 20, 7, 93, 29.$$

How many times did you insert into a node that still had room? How many node splits did you perform? What is the depth of the final tree? What is the ratio of free space to total space in the final tree?

Exercise 40
Prove that, if a key is not in a bottom node, its successor, if it exists, must be.

Exercise 41
(Trivial) Make a hash table with 8 slots and insert into it the following values:

$$15, 23, 12, 20, 19, 8, 7, 17, 10, 11.$$

Use the hash function

$$h(k) = (k \bmod 10) \bmod 8$$

and, of course, resolve collisions by chaining.

Exercise 42
Non-trivial Imagine redoing the exercise above but resolving collisions by open addressing. When you go back to the table to retrieve a certain element, if you land on a non-empty location, how can you tell whether you arrived at the location for the desired key or on one occupied by the overspill from another one? *(Hint: describe precisely the low level structure of each entry in the table.)*

Exercise 43
How can you handle deletions from an open addressing table? What are the problems of the obvious naïve approach?

Exercise 44
If we are using bubblesort, why did I indicate the costs as linear rather than quadratic?

Exercise 45
Prove that the sequence of trees in a binomial heap exactly matches the bits of the binary representation of the number of elements in the heap.