Algorithms I Examples

Dr Robert Harle, 2010

The Algorithms I notes are an excellent source of information. They also contain a few exercises that are intended to quickly test understanding. They are not numbered and this can be difficult for supervisions. This examples sheet can be used to guide supervisions: some of the more substantial exercises from the notes are referenced here. It is, however, recommended that each student tackle all of the exercises in the notes as well as here.

Algorithm Specification

1. Explain the purpose of preconditions and postconditions when specifying an algorithm and assertions when implementing it.

2. Suggest pre- and post-conditions and possible assertions for the following examples:

i) `double sqrt(double)`: a function that returns the square root of any argument.

ii) `double divide (double a, double b)`: a function that computes a/b.

iii) `void normalise (int[] a)`: a function that takes a 3D vector specified in `a` and normalises it to have magnitude 1.0.

3. Discuss the following reasoning. *Major programs such as operating systems must run indefinitely. Assertions cause a program's execution to halt abruptly. Therefore assertions are not appropriate when designing such programs.*

Asymptotic Analysis

4. Notes exercise page 15

5. Show that:

i) $f(n) = 6n^2 + 7n$ is $O(n^2)$

ii) $f(n) = \frac{6n^2+7logn}{n}$ is $O(n)$

iii) $f(n) = 4n^2 + 2$ is $\theta(n^2)$

iv) $f(n) = 1^2 + 2^2 + 3^2 + ... + n^2$ is $\theta(n^3)$

Sorting Algorithms

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. (Notes exercise page 22).

7. Of all of the $O(n^2)$ sorting algorithms discussed in the lectures, which would you expect to perform best given a random input? Justify your answer.

8. Show how to mergesort an array in $\frac{n}{2}$ space (Notes exercise page 26).

9. Write pseudocode for the bottom-up mergesort (Notes exercise page 27).

10. For the merge step in mergesort we have two sorted subarrays that we need to merge into one sorted array. A programmer suggests this is just like a half-finished insertion sort and could therefore be achieved using an insertion sort algorithm.

i) Write out the pseudocode for an insertion sort that assumes the first half of the data is in sorted order.

ii) If the resultant array is of length n, show that your modified insertion sort is $O(\frac{3n^2}{8})$.

iii) In a merge step, we can additionally make use of the fact that the second half off the array is presorted. What is the Big-O complexity of this new 'insertion merge' step?

11. Show the steps involved in sorting the digits 4632739 using:

i) Mergesort;

ii) Quicksort;

iii) Heapsort.

12. Can picking the quicksort pivot at random really make any difference to the expected performance? How will it affect the average case? The worst case? (Notes exercise page 29).

13. What is the smallest number of pairwise comparisons you need to (Notes exercises page 30,31):

i) Find the smallest of n items.

ii) Find the *second* smallest of n items.

14. Implement two versions of quicksort in Java:

i) One where the pivot is always chosen as the first element; and

ii) One where the pivot is chosen at random.

How do you know your implementations work?

15. For each of the sorting algorithms covered in the notes, establish whether it is stable or not (Notes exercise page 36).

Algorithm Design

16. Briefly explain the basics of the following strategies:

i) Brute-force;

ii) Divide-and-Conquer;

iii) Dynamic Programming;

iv) Back-tracking.

17. i) Write pseudocode to solve the coin change problem introduced in lectures. Assume there are *k* denominations and that one of them is 1 (so there is always a solution). Your code should identify the optimal solution (i.e. *which* coins, not just how many of them there are).

ii) What is the computational complexity of your code?

18. Suggest a divide-and-conquer algorithm to find both the maximum and minimum of a set of n integers. What is the complexity of your algorithm?

19. In the lectures it was shown that a divide-and-conquer approach to multiplying two numbers was $O(n^{1.585})$ when dividing the numbers into two and using Gauss's trick to need only three multiplications with each merge. Repeat the complexity analysis, but for a split into *three* at each level. You may use the fact that the merge step can then be done in five multiplications. Comment on your answer.

20. Suggest how to use a backtracking algorithm to solve a sudoku puzzle.

21. * Two examiners are looking to fairly split the workload involved in marking a set of dissertations, $D_1, ..., D_n$. They know that simply dividing the pile in two does not give a fair split, since each dissertation is individual. They decide that the effort in marking a dissertation is proportional to its wordcount (which has been printed on the front of each dissertation). Therefore ghey seek to divide the dissertations into two sets, with each set having the same total wordcount, or the closest to it possible and use a dynamic programming algorithms to do so.

i) Consider the quantity $X(i, w)$ which is 1 if the total wordcount w can be made from *any* subset of the dissertations D_1 to D_i, and 0 otherwise. Give a recursive definition of $X(i, w)$, using the function $wordcount(D_i)$ to represent the wordcount of dissertation D_i.

ii) For the two final sets, A and B, we want the respective wordcounts to be as close as possible to half of the total wordcount of all the dissertations, $W = \frac{1}{2} \sum wordcount(D_i)$. Therefore for set A we are looking to minimise $(W - w)$, where w ranges over all achievable wordcounts. Write a minimisation expression involving X that embodies this.

Elementary Data Structures

22. Write pseudocode that tests whether a string is a palindrome (the same backwards as forwards) using only the standard operations of a stack.

23. Discuss the advantages and disadvantages of the linked-list and array based implementations of a queue.

24. Consider implementing a stack using an array. When the array is full and a push() is requested, there are two common strategies to growing the array: increase the array by a constant number of elements or double the size of the array. Analyse both strategies to find the amortized costs associated with a push() operation.

25. Write pseudocode or Java code for a deque that uses a circular buffer

26. Consider a table implementation based on a linked list. Write pseudocode for the set() function, assuming duplicates are not permitted.

BST, 2-3-4 and Red-Black Trees

27. The main issue with BSTs is their tendency to become unbalanced. This can be a particular issue if the input keys have a lot of duplicates (e.g. insert $\{1,1,1,1,1,1,1\}$ gives a very unbalanced tree. Suggest a way in which duplicate entries in a BSt could be addressed such as to produce a more balanced result.

28. Using the sequence $\{10,85,15,70,20,60,30,50,65,80,90,40,5,55\}$ show how to create:

i) a 2-3-4 tree;

ii) a red-black tree.

29. Show how to delete a node from a red-black tree.

30. Find a sequence of ten numbers that results in a worst-case red-black tree (i.e. the 10-node red-black tree with the longest path from root to leaf).

B-Trees

31. When analysing trees, complexities are often O(height of tree). Given that B-trees can be used to produce trees of lower height than the equivalent BST, explain why;

i) B-trees are not a better choice than Red-Black trees when the tree is stored purely in memory.

ii) B-trees are a better choice than Red-Black trees when the tree is stored on disk.

32. What are the minimum and maximum number of keys in a B-tree of minimum degree 700 (i.e. t=700) and height h?

Hash Tables

33. Prove the search complexity results from lectures for a hash table that uses chaining to resolve collisions. i.e. $O(1+\alpha)$.

34. Suggest how to delete hash table entries when resolving collisions using:

i) chaining;

ii) open addressing.

35. How would you use a hash table to create a basic spell checker (i.e. decide whether or not a word is correctly spelt)? For an incorrectly-spelt word, how would you efficiently provide a set of correctly-spelt alternatives?

Priority Queues

36. Consider a priority queue implemented using i) a doubly linked list and ii) a singly linked list. For each case state and justify the complexities of the standard priority queue functions.

37. Recall that modern computers give the appearance of running multiple applications simultaneously, whilst actually running each in turn for very short time periods. Priority queues are often used to queue the applications so that those deemed to be more important get to the front of the queue, ahead of any 'background' applications. Unfortunately this can lead to *starvation*, where the low priority applications never make it to the head of the queue because higher priority applications keep jumping in first. Suggest how you might address this.

38. Create a Java class to represent a node within a binomial heap.

39. i) Create two binomial heaps: one containing binomial trees of order 0,1, and 3; and one containing trees of order 1, 2 and 3.

ii) Show how to merge your two heaps.

iii) Show how to extract the minimum from your merged heap.