6–10 August 2018

Plan

Last week's lecture discussed asymptotics and the mathematical analysis of algorithms; this week's lectures will be about brute force methods and recursion.

This week's questions 12–14 are combinatorics-style puzzles. Good problem-solving fun, but not essential.

The exercises

7. One way of representing an undirected graph G is using an *adjacency matrix*. This is an $n \times n$ matrix, where n is the number of nodes in G. In this matrix, we label the rows and the columns with the names of G's nodes. For a pair (x, y) of nodes, the matrix has a 1 in the (x, y) entry if G has an edge connecting x and y; otherwise the entry has a 0. (As the graph is undirected, this is somewhat redundant: (x, y) and (y, x) will always be identical.) A *complete* graphs is one in which x and y are connected, for all x, y with $x \neq y$. We don't exclude the possibility of a *loop*, that is, an edge connecting a node to itself, although loops are rare.

   How can you tell from its matrix whether an undirected graph

   (a) is complete?
   (b) has a loop?
   (c) has an isolated node?

8. Design an algorithm to check whether two given words are anagrams, that is, whether one can be obtained from the other by permuting its letters. For example, *garner* and *ranger* are anagrams.

9. Here we consider a function `first_occ`, such that `first_occ`(d, s) returns the first position (in string s) which holds any symbol from the collection d (and returns -1 if there is no such position). For simplicity let us assume both arguments are given as strings. For example, `first_occ("aeiou", s)` means "find the first vowel in `s`." For $s = $ "zzzzzzzzzzzz" this should return -1, and for $s = $ "Phlogiston", it should return 3 (assuming we count from 0).

   Assuming strings are held in arrays, design an algorithm for this and find its complexity as a function of the lengths of d and s.

   Suppose we know that the set of possible symbols that may be used in d has cardinality 256. Can you find a way of utilising this to speed up your algorithm? Hint: Find a way that requires only one scan through d and only one through s.

10. Gaussian elimination, the classical algorithm for solving systems of n linear equations in n unknowns, requires about $\frac{1}{3}n^3$ multiplications, which is the algorithm's basic operation.

   (a) How much longer should we expect Gaussian elimination to spend on 1000 equations, compared with 500 equations?

   (b) You plan to buy a computer that is 1000 times faster than what you currently have. By what factor will the new computer increase the size of systems solvable in the same amount of time as on the old computer?

11. For each of the following pairs of functions, indicate whether the components have the same order of growth, and if not, which grows faster.

   (a) $n(n+1)$ and $2000n^2$ (b) $100n^2$ and $0.01n^3$

   (c) $\log_2 n$ and $\ln n$ (d) $\log_2^2 n$ and $\log_2 n^2$

   (e) 2^{n-1} and 2^n (f) $(n-1)!$ and $n!$

   Here ! is the factorial function, and $\log_2^2 n$ is the way we usually write $(\log_2 n)^2$.

12. Eight balls of equal size are on the table. Seven of them weigh the same, but one is slightly heavier. You have a balance scale that can compare weights. How can you find the heavier ball using only two weighings?

13. There are 18 gloves in a drawer: 4 pairs of red gloves, 3 pairs of yellow, and 2 pairs of green. You select gloves in the dark, which means you cannot assess colour nor left/right-handedness. You can check the gloves only after the selection has been made. What is the smallest number of gloves you need to pick to have at least one matching pair in the best case? In the worst case?

14. After washing 5 distinct pairs of socks, you discover that two socks are missing. Hence you are left with 4 complete pairs in the best case, and 3 in the worst case. What is the probability of the best case scenario?