School of Computing and Information Systems COMP90038 Algorithms and Complexity Tutorial Week 3

Sample Answers

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?

Answer: Ssignment Project Exam Help

- (a) All elements (
- (b) Some element that the simple state of the some row (and the simple state of the some row (and the simple state of the simp
- 8. Design an algorithm to check whether two given words are anag can be obtained find the other permanage its constraints. For as S Street and the other permanage are anagrams.

Answer: Note that we cannot just say "for each letter in the first word, check that it occurs in the second, and vice versa." (Why not?)

A nice solution sorts the letters in each word and simply compares the two results. This solution scales well, that is, has good performance even as the words involved grow longer.

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, 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.

Answer: Let m be the length of d and let n be the length of s. The obvious algorithm is to scan through s and, for each symbol x met, go through d to see if that was a symbol we were looking for. In the worst case, when no symbols from d are in s, this means making mn comparisons. (We could also structure the search the other way round: for each element of d, find its first occurrence, and then, based on that, decide which occurrence was the earliest of them all; this also leads to mn comparisons in the worst case.)

Knowing that there are 256 possible symbols that may be used in d allows us to introduce an array first, indexed by the symbols. (If we are talking ASCII characters then, conveniently, the indices run from 0 to 255.) Initialise this array so all its elements are, say, -1. Now scan through s. For each symbol x in s, if first[x] is -1, replace it with x's position. Note that we have not considered d at all so far; we have only recorded, in first, where the first occurrence of each symbol is (and left the value as -1 for each symbol that isn't in s).

Now all we need is a single scan through d, to find the value of first[x] for each symbol x in d. As we find these, we keep the smallest such value. It this value is -1, none of the symbols from d was found in s. Otherwise the value is the first occurrence of a symbol from d.

The complexity in this case is n + m. For most reasonable values of m and n, this is smaller than mn, so we have a better algorithm, even taking the overhead of maintaining first into account. Essentially we have traded in some space (the array first) to gain some time. This kind of time/space trade-off is a familiar theme in algorithm design.

kind of time/space trade-off is a fartijar theme in algorithm design. Help

10. Gaussian elimination, the classical algorithm for solving systems of n linear equations in n unknowns, requir $\frac{1}{3}$ ic operation.

(a) How much https://eduassistpro.github.io/ ns,

(b) You plan to buy a computer that is 1000 times faster than what By what factor will the vector puter increase the size of system amount of time and the bloomputer increase the size of system.

Answer:

- (a) The answer is: 8 times longer, and it does not depend on the particular numbers of equations given in the question. We need $\frac{1}{3}(2n)^3$ operations for 2n equations and $\frac{1}{3}n^3$ operations for n. The ratio is $\frac{\frac{1}{3}(2n)^3}{\frac{1}{3}n^3} = \frac{8n^3}{n^3} = 8$.
- (b) If the new machine needs t_{new} units of time for a job, the old one needs $t_{old} = 1000 \ t_{new}$ units. Let m be the number of equations handled by the new machine in the time the old machine handles n. The new machine handles m in time $\frac{1}{3}m^3$ and n in time $\frac{1}{3}n^3$. So we have

$$\frac{1}{3}m^3 = 1000 \cdot \frac{1}{3}n^3$$

Solving for m we get

$$m = \sqrt[3]{1000 \ n^3} = 10n$$

So we can now solve systems that are 10 times larger.

- 11. For each of the following pairs of functions, indicate whether the components have the same rate of growth, and if not, which grows faster.
 - (a) n(n+1) and $2000 n^2$ (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 $(\log_2 n)^2$.

Answer:

- (a) Same rate of growth.
- (b) $0.001n^3$ grows faster than $100n^2$. Namely, as soon as n>1, we have $n^2< n^3$. So pick the constant c to be 10,000. When n > 1, we have $100n^2 < 10,000 \cdot (0.001n^3)$.
- (c) Same rate of growth.
- (d) $\log_2 n^2 = 2\log_2 n$, so $\log_2 n^2$ has the same rate of growth as $\log_2 n$. However, $\log_2^2 n$ grows faster; namely $\lim_{n\to\infty} \frac{\log_2 n \cdot \log_2 n}{\log_2 n} = \lim_{n\to\infty} \log_2 n = \infty$.
- (e) Same rate of growth, since $2^n = 2 \cdot 2^{n-1}$.
- (f) nArssignment; Projecta xam Help
- 12. Eight balls of equal size are on the table. Seven of them weigh the same, but one is slightly heavier. You have a b

ball using only that the ball using only the ball using only that the ball using only that the ball using only the ball using the ball using

Answer: Put th

compare the two remaining balls. Otherwise keep just the thr pan. If one is heavier, that's the heavy ball, otherwise the heavy bal

- 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 must pick to have at least one matching pair in the best case? In the worst case?
 - **Answer:** The best case is obviously 2 gloves. To analyse the worst case, note that we might pick as many as 4 red, 3 yellow, and 2 green gloves, and yet not have a pair, because for each colour, we happened to take gloves of the same orientation, or "handed-ness". So to be absolutely certain that we have a pair, we need to pick 4+3+2+1=10 gloves.
- 14. After washing 5 distinct pairs of socks, only 8 socks come back from the clothes line. 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?

Answer: There are $\binom{10}{2} = 45$ ways to select two socks from 10. In five of these cases the two socks have the same colour. These five cases correspond to the best-case scenario, in that we are left with four complete pairs. The probability of this happening is $\frac{5}{45} = \frac{1}{9}$.