

CS 535 Homework 2

Due: September 23 (W), in class.

The problems below are all search problems. In each case, you are asked to do better than brute force. If you are registered as an undergraduate student, please answer questions 1,2 and 3. Question 4 is a bonus problem. If you are registered as a graduate student, please answer questions 2, 3 and 4. Question 1 is a bonus problem for you.

1. An array A contains $n - 1$ unique integers in the range $[0, n - 1]$. That is, there is one number from this range that is not in A . Your goal is to determine the missing number.
 - a. Describe a brute force way of solving this problem. What is its worst-case runtime?
 - b. Design an $O(n)$ -time algorithm for the problem. Please make sure that you explain why your algorithm works and why its runtime is $O(n)$.
2. Suppose that each array of $n \times n$ array A consists of 1's and 0's such that in any row of A , all the 1's come before all the 0's in that row. Our problem is to find a row of A that contains the *least* number of 1's.
 - a. Describe a brute force way of solving this problem. What is its worst-case runtime?
 - b. Design an $O(n)$ -time algorithm for the problem. Please make sure that you explain why your algorithm works and why its runtime is $O(n)$.
3. There are n coins, $n - 1$ of which are genuine and one of which is fake. All the genuine coins have the same weight; the fake coin is either lighter or heavier. To find the fake coin, we are given a balance beam with two pans.
 - a. Describe a procedure that would find the fake coin using as few weighings as possible when there are $n = 3$ coins. How about when $n = 9$?
 - b. Generalize your method above to an arbitrary n . How many weighing are needed by your procedure?
4. Suppose we now have an $n \times n$ array B that contains *distinct* numbers such that (i) the numbers in each row are increasing from left to right and (ii) the numbers in each column are increasing from top to bottom. Assume B is already stored in memory. We want to answer queries of the form “Given x , does B contain the number x ?” efficiently.

Consider the following algorithm:

Find(A, n, x)

$row \leftarrow 0$

$col \leftarrow n - 1$

while ($row < n$ and $col > -1$)

 if $A[row][col] > x$

$col \leftarrow col - 1$

 else

 if $A[row][col] < x$

$row \leftarrow row + 1$

 else

 return("x is in $A[row][col]$ ")

return("x is not in the array")

a. *Warm up!* Construct a 4×4 array with the properties of B . Show how the algorithm works when $x = 10$ and $x = 100$. That is, mark $B[row][col]$ at every iteration of the algorithm.

b. Determine the runtime of the algorithm.

c. Now argue why the algorithm is correct; that is, why will it always produce the correct answer?