

CS 1050A: Constructing Proofs

Solutions to Quiz 3

Nov 20, 2006

Please answer questions in the space provided. You can use your own extra pages if you want. Answer all questions.

Problem 1 (15 points)

Compute $\gcd(175, 63)$ using Euclid's algorithm. Show all the steps of the algorithm.

$$175 = 2(63) + 49$$

$$63 = 1(49) + 14$$

$$49 = 3(14) + 7$$

$$14 = 2(7) + 0$$

Therefore, $\gcd(175, 63) = 7$.

Problem 2 (20 points)

For each of the following statements, answer whether it is TRUE or FALSE by encircling your response.

1. If $f, g : \mathbb{N} \mapsto \mathbb{N}$ are any functions such that $f(n)$ is $O(g(n))$, then $2^{f(n)}$ is $O(2^{g(n)})$.

TRUE FALSE

FALSE

$f(n)$ is $O(g(n))$

$\Rightarrow \exists c \in \mathbb{R} \text{ and } N \in \mathbb{Z}, f(n) \leq c \cdot g(n) \forall n \geq N$

$\Rightarrow 2^{f(n)} \leq (2^{g(n)})^c$

2. For any two positive integers a and b , $\gcd(a, b) = \gcd(a + b, 2a + b)$.

TRUE FALSE

TRUE

Let $\gcd(a, b) = g_1$ and $\gcd(a + b, 2a + b) = g_2$.

$g_1 | a, g_1 | b \Rightarrow g_1 | a + b$

$g_1 | a, g_1 | b \Rightarrow g_1 | 2a + b$

However, g_2 is $\gcd(a + b, 2a + b)$, $g_1 \leq g_2$. $g_2 | a + b, g_2 | 2a + b \Rightarrow g_2 | (2a + b) - (a + b)$ i.e. $g_2 | a$

$g_2 | a + b, g_2 | a \Rightarrow g_2 | a + b - a$ i.e. $g_2 | b$

However, g_1 is $\gcd(a, b)$, $g_2 \leq g_1$.

Therefore, $g_1 = g_2$.

3. If a, b, n are positive integers such that $n | ab$, then either $n | a$ or $n | b$.

TRUE FALSE

FALSE

Let $n = 12, a = 6, b = 4$. $12 | 6 \cdot 4$ but neither $12 | 6$ nor $12 | 4$.

4. $3n^2 + 4n + 7$ is $O(n^3)$.

TRUE FALSE

TRUE

$\forall n \geq 1, 3n^2 + 4n + 7 \leq 3n^3 + 4n^3 + 7n^3$

Problem 3 (25 points)

Let L_1 and L_2 be two lists of positive integers. L_1 has k elements, L_2 has n elements, and $k \leq n$. Assume that the integers in list L_1 are all distinct, and the same holds for list L_2 . The goal is to decide whether there is an integer that occurs in *both* the lists. Consider two algorithms:

Algorithm-1: For each of the k elements in list L_1 , decide whether it occurs in list L_2 using linear search. If the search succeeds for some element in L_1 , then output YES, else output NO.

Algorithm-2: Sort L_1 and L_2 , merge the two sorted lists into a single sorted list L , and scan L looking for a repeated element (repeated elements, if exist, will appear consecutively in L). If a repeated element is found, output YES, else output NO.

Answer the following questions (there is no need to give justification):

1. Give a reasonable upper bound on the number of comparisons needed by Algorithm-1 (you could give your answer in big-O notation).

Linear search for one element takes atmost n comparisons. So, linear search for k elements will take atmost $n \cdot k$ comparisons. Therefore, Algorithm-1 is $O(n \cdot k)$

2. Give a reasonable upper bound on the number of comparisons needed by Algorithm-2 (you could give your answer in big-O notation).

Case 1 : Using $O(n \log_2 n)$ sorting algorithm (like merge sort)

Step 1 - Sort $L_1 \Rightarrow$ atmost $k \log_2 k$ comparisons

Step 2 - Sort $L_2 \Rightarrow$ atmost $n \log_2 n$ comparisons

Step 3 - Merge L_1 and $L_2 \Rightarrow$ atmost $n + k$ comparisons

Step 4 - Linear search \Rightarrow atmost $n + k$ comparisons

So, there are atmost $k \log_2 k + n \log_2 n + 2n + 2k$ comparisons. Therefore, Algorithm-2 is $O(n \log_2 n)$.

Case 2 : Using $O(n^2)$ sorting algorithm (like insertion sort)

Step 1 - Sort $L_1 \Rightarrow$ atmost k^2 comparisons

Step 2 - Sort $L_2 \Rightarrow$ atmost n^2 comparisons

Step 3 - Merge L_1 and $L_2 \Rightarrow$ atmost $n + k$ comparisons

Step 4 - Linear search \Rightarrow atmost $n + k$ comparisons

So, there are atmost $k^2 + n^2 + 2n + 2k$ comparisons. Therefore, Algorithm-2 is $O(n^2)$.

3. If $k = 10$, and n is very large, then which algorithm takes less number of comparisons?

Algorithm-1 will have $O(10n) = O(n)$ running time.

Algorithm-2 will have $O(n^2)$ (*Case 2*) or $O(n \log_2 n)$ (*Case 1*) running time.

Therefore, Algorithm-1 runs faster.

4. If $k = \frac{n}{10}$, and n is very large, then which algorithm takes less number of comparisons?

Algorithm-1 will have $O(\frac{n^2}{10}) = O(n^2)$ running time.

Algorithm-2 will have $O(n^2)$ (*Case 2*) or $O(n \log_2 n)$ (*Case 1*) running time.

Therefore, Algorithm-2 runs faster in *Case 1* and both run in roughly the same time in *Case 2*.

5. For what value of k (as a function of n), both algorithms run in roughly the same time?

Case 1 : $nk = n \log_2 n \Rightarrow k = \log_2 n$

Case 2 : $nk = n^2 \Rightarrow k = n$