Assignment 5. 2020-10-14,

12 minutes

Question 1 (Algorithm to find GCD). This algorithm is often used to find the *greatest common divisor* of nonnegative integers a and b.

```
GCD-One(a, b) :
if a == 0 :
    return b :
if b == 0 :
    return a :
while b > 0 :
    (Remainder when a is divided by b)
    t = a mod b
    a = b
    b = t
return a
```

The worst case time for this algorithm is achieved when we input Fibonacci numbers (it has to run the longest relative to the input size). For example, if $a = 144$, $b = 89$, then:

$$(144, 89) \to (89, 55) \to (55, 34) \to$$

$$\to (34, 21) \to (21, 13) \to (13, 8) \to (8, 5)$$

$$\to (5, 3) \to (3, 2) \to (2, 1) \to (1, 0).$$

At the last step $a = 1$ and $b = 0$, so it returns $a = 1$ which equals $\gcd(144, 89)$.
It is known that n-th Fibonacci number

$$F_n \approx \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^n .$$

It grows as a geometric progression.
Write the time complexity of finding GCD of two numbers (a, b) in terms of n, where n is the total number of digits in numbers a and b.
Express your answer, using the "Big-O-Notation".

Question 2 (Another algorithm for GCDD). Modify the above algorithm - instead of dividing with remainder, we subtract b from a repeatedly (and we swap a and b, whenever $b > a$).

```
GCD-Two(a, b) :
if a == 0 :
    return b :
if b == 0 :
    return a :
while b > 0 :
    a = a - b
    if b > a :
        (a becomes b and vice versa)
        swap(a, b)
return a
```

For example, if $a = 75$, $b = 30$ (GCD is 15), we run it like this:

$$(75, 30) \to (45, 30) \to (15, 30)_{swap} \to$$

$$\to (30, 15) \to (15, 15) \to (0, 15)_{swap} \to (15, 0).$$

Write the time complexity of finding GCD of two numbers (a, b) in terms of n (where n is your input length).
Express your answer, using the "Big-O-Notation".

Solutions

Question 1. Answer: $O(n)$.
The algorithm GCD-ONE(a, b) has time complexity $O(n)$.
Intuitively, if we have 100-digit numbers, then we would need $100k$ steps for the algorithm to complete - so it is linear. (Here we assume that all arithmetic operations take constant time; the algorithm may take longer, if numbers a, b are so large that they exceed 2^{64} or other CPU register size limit.)
It was told in this problem that the worst-case complexity is achieved, if both arguments to GCD-TWO(a, b) are Fibonacci numbers. The lengths of a and b cannot exceed n digits (in decimal notation), therefore $a, b < 10^n$. If any of them is the k-th Fibonacci number (e.g. $a = F_k$, then we would spend $c \cdot k$ steps before the algorithm reaches $F_0 = 0$. We get

$$F_k \approx \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^k < 10^n,$$

$$\left(\frac{1+\sqrt{5}}{2} \right)^k < \sqrt{5} \cdot 10^n,$$

$$k < \frac{\ln(\sqrt{5} \cdot 10^n)}{\ln \frac{1+\sqrt{5}}{2}} = c \cdot n.$$

Therefore, the number of while-loop iterations k is $O(n)$.

Question 2. Answer: $O(10^n)$.
The worst case (maximum number of subractions) for the algorithm GCD-TWO(a, b) happens, if $a = 10^n - 1$ (the largest n-digit integer having n digits $= 9$) and $b = 1$. In this case we will subtract b from a $O(10^n)$ times.
Note 1: Time complexity should NOT expressed in terms of actual arguments a and b, but it only depends on n, where n denotes the length of its input (total number of digits of a and b).
Note 2: Answer $O(\log n)$ for GCD-ONE(a, b) or $O(n)$ for GCD-TWO(a, b) would be true, if the input for a, b is written in *unary counting system.*

Grading:

- Correct answers ($O(n)$ and $O(10^n)$ respectively) –10 points.
- Answers $O(\log n)$ and $O(n)$ with some justification (why remainders are much faster than subtraction) –7 points.
- $O(\log n)$ and $O(n)$ (but without any justification) –5 points.
- Similar to $O(\log n)$ and $O(n)$ (but a, b used instead of n) –4 points.
- Just $O(\log n)$ for the 1st item –2 points.