Álgebra Abstracta (CCOMP3-1) **Nombres:** Luigi Anthony Cabrera Huanqui
2021-2
Laboratorio 02b
01/10/2021
Tiempo: — Prof. José Chávez

Este documento contiene 2 páginas (incluyendo esta portada) y 2 problemas (o preguntas). Compruebe si falta alguna página. Ingresar toda la información solicitada en la parte superior de esta página, y ponga sus iniciales en la parte superior de cada página, en caso de que las páginas se separen.

Usted no puede utilizar libros, apuntes, o calculadora en este examen/práctica.

En este examen/práctica se aplican las siguientes reglas:

Si usted usa algún teorema, entonces debe indicarlo y explicar por qué se debe aplicar el teorema.

Organice su respuesta en una forma ordenada y coherente. Un teorema no puede ser demostrado con un ejemplo; pero para demostrar que una proposición no es verdadera, puede dar un contraejemplo.

Supongamos que quiere demostrar que **un objeto existe y es único**. Primero muestre que el objeto realmente existe. Para demostrar que es único, supongamos que hay dos objetos: r y s. Entonces demuestre que $r = s$.

Una respuesta correcta, no respaldada por cálculos o explicación **no recibirá puntaje**; una respuesta incorrecta apoyada mediante cálculos y explicaciones sustancialmente correctos **podría recibir un puntaje parcial**.

Si necesita más espacio, use el reverso de las páginas; indique claramente cuándo ha hecho esto.

Problema	Points	Score
1	5	
2	15	
Total:	20	

No escriba en la tabla de la derecha.

Theorem 1. *Si $a > b \geq 1$, donde $a, b \in \mathbb{N}$ y EUCLID(a, b) requiere $k \geq 1$ pasos para obtener el máximo común divisor. Entonces $a \geq F_{k+2}$ y $b \geq F_{k+1}$.*

Proof. La demostración procede por inducción.

Base: Asumimos que $k = 1$. Entonces

$$b \geq 1 = F_2 = F_{1+1} \qquad (k = 1),$$

y además $a \geq b + 1 \geq 2 = F_3 = F_{1+2} \quad (k = 1)$.

Paso inductivo: Suponemos que el teorema se cumple si se requieren $n \geq 1$ pasos. Entonces debemos probar que el teorema se mantiene para $n + 1$.

Como EUCLID(a, b) llama a EUCLID$(b, a \mod b)$ de manera recursiva, entonces este último realizará n pasos para calcular el máximo común divisor. Utilizando la hipótesis inductiva se tiene que $b \geq F_{n+2}$ (aquí se prueba parte del teorema) y $(a \mod b) \geq F_{n+1}$. Si sumamos ambas desigualdades obtenemos lo siguiente

$$\begin{aligned} b + (a \mod b) &\geq F_{n+2} + F_{n+1} \\ b + (a - \lfloor \tfrac{a}{b} \rfloor b) &\geq F_{n+3} \\ a - b(\lfloor \tfrac{a}{b} \rfloor - 1) &\geq F_{n+3} \\ a - b(\lfloor \tfrac{a}{b} \rfloor - 1) &\geq F_{n+3} \qquad (\lfloor \tfrac{a}{b} \rfloor \geq 1) \end{aligned}$$

Al final se tiene que $a \geq F_{n+3}$.

□

1. (5 points) Calcular el tiempo computacional del Algoritmo de Euclides. Detalle y sustente su respuesta.

   El tiempo de ejecución del algoritmo de Euclides puede ser calculado llevando cuenta de las llamadas recursivas que hace la función. En cada una de estas llamadas se hace una operación modulo, una comparación y un retorno (El valor del GCD o una llamada recursiva)

   Dentro de las operaciones mencionadas, la de mayor complejidad temporal es el modulo, por lo que para calcular el tiempo se contarán el número de operaciones modulo que se harán. Esto es igual al número de llamadas recursivas.

   Por otro lado, se puede reconstruir el peor caso del algoritmo de Euclides:

A	B
(F_{10}) 55	(F_9) 34
(F_9) 34	(F_8) 21
(F_8) 21	(F_7) 13
(F_7) 13	(F_6) 8
(F_6) 8	(F_5) 5
(F_5) 5	(F_4) 3
(F_4) 3	(F_3) 2
(F_3) 2	(F_2) 1

   Podemos notar que el peor caso se generaliza al hacer la llamada de números de Fibonacci consecutivos F_{k+2} y F_{k+1}. Esto denota que si A >= F_{k+2} y B >= F_{k+1} entonces se requerirán un máximo de k pasos para encontrar GCD(A, B). (Teorema probado en clase)

   Para obtener el O(n) (Número de operaciones del algoritmo) hacemos lo siguiente:

   Denotamos que A > B >= 1

   Si el algoritmo requiere de N pasos, entonces B >= F_{n+1}

   Esto a la vez es: B >= F_{n+1} >= Φ^(N-1)

   Resolviendo B >= Φ^(N-1), resulta N-1 <= $\log_{\Phi}(B)$ ***(I)***

   Se sabe que 1/5 < $\log_{10}(\Phi)$ ***(II)***

   Multiplicando ***(I) y (II)***:

   (N-1) * 1/5 < $\log_{\Phi}(B) * \log_{10}(\Phi)$

   (N-1) / 5 < [$\log_{10}(B)$ / ~~$\log_{10}(\Phi)$~~] * ~~$\log_{10}(\Phi)$~~ // Cambio de base (Propiedad de Logaritmos)

   N – 1 < 5 * $\log_{10}(B)$

   N < 5 * $\log_{10}(B)$ //Termino 1 constante se desprecia en O(n)

   Se sabe que $\log_{10}(B)$ >= número de dígitos de B. Por lo tanto:

   Se puede concluir que el número de operaciones aproximado del algoritmo de Euclides no es mayor a 5 veces el número de dígitos del término B.

   De esto para obtener el tiempo computacional del algoritmo se obtiene que **S(x) = x * 5 * $\log_{10}(B)$**
   Donde ***S(x)*** son los segundos que tardaría el algoritmo y ***x*** (decimal muy pequeño) los segundos que se tarda un programa en realizar una operación modulo.

2. (15 points) Implementar el Algoritmo Extendido de Euclides.

- Implementar un programa que permita ingresar dos números a y b (enteros positivos) y que retorne $\{\gcd(a, b), x, y\}$, donde $\gcd(a, b) = ax + by$ $(x, y \in \mathbb{Z})$.

```python
def ext_euclid(a,b):
    if a % b == 0:
        return b,0,1
    r, x_, y_ = ext_euclid(b, a%b)
    x, y = y_, x_ - int(a/b)*y_
    return r,x,y

while True:
    print("Write 0 to exit")
    a = int(input("Write A: "))
    if a == 0: break
    b = int(input("Write B: "))
    if b == 0: break
    r, x, y = ext_euclid(a, b)
    print(f"\nGCD({a}, {b}) = {r}\nx = {x}\ty = {y}\n\n")
```

- Enviar un enlace al repositorio (única forma de envío). En este colocará un README.md con una breve descripción del programa y las instrucciones para ejecutarlo. Colocar un ejemplo del resultado esperado.

Link del Repositorio: https://github.com/LuigiCabrera/Tarea-Euclides-GCD-