



Nombre: Luis Alberto Vargas González.

Fecha: 13/02/2025.

U2 A1: Calcular números Fibonacci en tiempo logarítmico.

Clase: Análisis y Diseño de Algoritmos.

Maestría en Ciberseguridad.

Introducción.

En esta práctica de programación se realizó un algoritmo que de manera logarítmica puede calcular $F(n)$ para poder dar el resultado de dicho $F(n)$ que además se diferencia de listar los primero n números de Fibonacci, es decir; no es lo mismo calcular el n -ésimo número de Fibonacci que listar los primero n números de la secuencia de Fibonacci: por ejemplo:

$$F(10) = 1+2+3+4+5+6+7+8+9+10 = 55 \text{ por lo tanto } F(10) = 55$$

Mientras que: Calcular los primeros 10 números de la secuencia de Fibonacci se realiza de esta manera:

$$F(0) = 0$$

$$F(1) = 1$$

$$F(2) = 1$$

$$F(3) = 2$$

$$F(4) = 3$$

$$F(5) = 5$$

$$F(6) = 8$$

$$F(7) = 13$$

$$F(8) = 21$$

$$F(9) = 34.$$

Teniendo esto aclarado, procedo a mostrar el código fuente:

```
# Este import es necesario para manejar número grandes
en Python sin problemas de límite de dígitos que puede
imprimir Python

import sys

sys.set_int_max_str_digits(10000)

#Esta es la función donde se calcula el n-ésimo número
de Fibonacci usando la matriz de transformación

def fibonacci(n):

    if n == 0:

        return 0

    base_matrix = [[1, 1], [1, 0]]

    result = matrix_power_iterative(base_matrix, n)

    return result[0][1]

# Esta función eleva una matriz a la potencia n de
manera iterativa.

def matrix_power_iterative(matrix, n):

    result = [[1, 0], [0, 1]] # Matriz identidad
```

```
while n > 0:

    if n % 2 == 1:

        result = multiply_matrices(result, matrix)

    matrix = multiply_matrices(matrix, matrix)

    n //= 2

return result

# Esta función multiplica dos matrices 2x2 cumpliendo
# con la propiedad de la multiplicación de matrices.

def multiply_matrices(A, B):

    return [
        [A[0][0]*B[0][0] + A[0][1]*B[1][0],
         A[0][0]*B[0][1] + A[0][1]*B[1][1]],
        [A[1][0]*B[0][0] + A[1][1]*B[1][0],
         A[1][0]*B[0][1] + A[1][1]*B[1][1]]
    ]

# Esta es la parte principal del programa donde se
# solicita al usuario el número de Fibonacci que desea
# calcular.

try:
    entrada = input ("¿Qué número de Fibonacci deseas
calcular (n)? ").strip()

    print(f"[DEBUG] Entrada recibida: '{entrada}'")

    n = int(entrada)
```

```
resultado = fibonacci(n)

print(f"\nEl número F({n}) de la secuencia de
Fibonacci es:\n{resultado}")

except Exception as e:
    print(f"X Ocurrió un error: {e}")
```

Brevemente explicaré lo que hace este programa: cómo podemos ver tenemos solamente 3 funciones, donde cada una de ellas realiza su tarea particular; En la función llamada `multiply_matrices` se definen las matrices que se operan, las cuales son 2 matrices de 2x2 que se necesitan fundamentalmente para realizar los cálculos que de otra manera sería hacerlos recursivos o con ciclos `for`, lo cual es ineficiente al momento de calcular $F(n)$ de tamaños grandes; es decir: $F(10,000)$ o más, y es clave para elevar la matriz a la potencia n .

La función llamada `matrix_power` es la parte clave del algoritmo ya que es la encargada de elevar la matriz 2x2 resultante a la potencia n usando `divide y vencerás` también conocida como exponenciación rápida. Con lo que logramos conseguir un tiempo $O(\log n)$.

La tercera y principal función llamada `Fibonacci` es la encargada de mandar llamar a la función `matrix_power`, en donde evaluamos si el

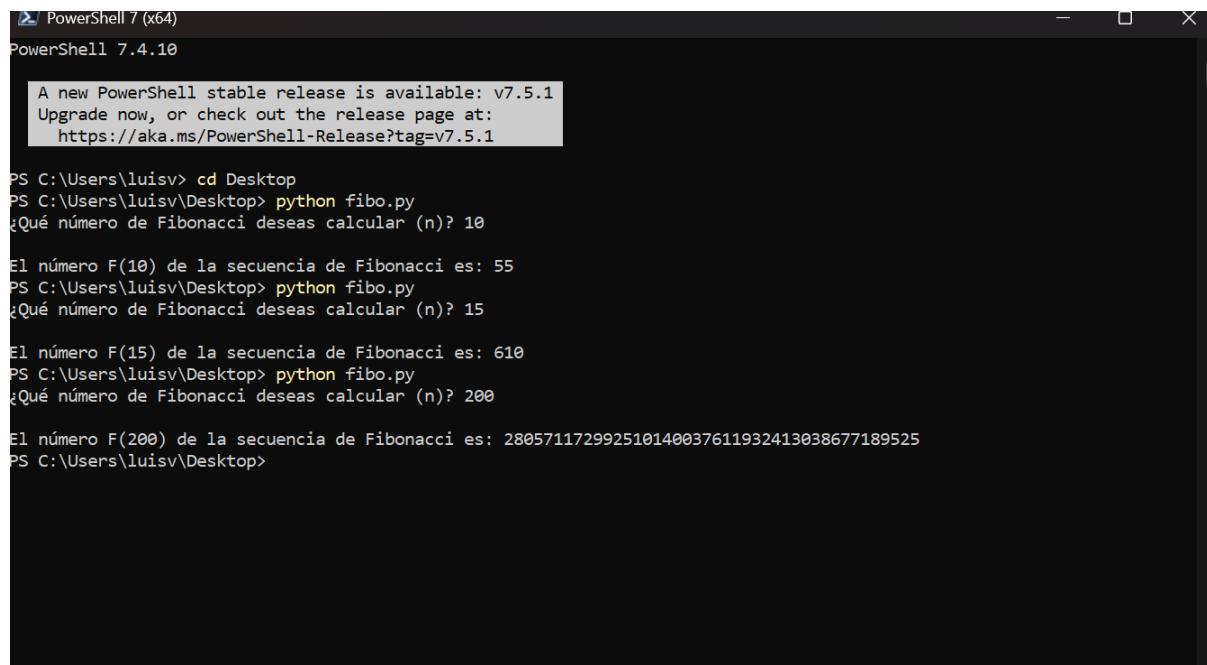
usuario pide F (0) simplemente se entrega lo que la secuencia de Fibonacci dicta = 0. Después se usa la matriz base que representa:

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n = \begin{bmatrix} f_{n-1} & f_n \\ f_n & f_{n+1} \end{bmatrix}$$

Para que justo después, se manda a llamar a la función matrix_power que será la que nos ayudará a elevar la matriz 2x2 a la potencia n.

Al final del programa simplemente lo que se hace es pedir al usuario cual es número de Fibonacci que desea calcular para que lo ingrese por pantalla y se hace un manejo de errores para evitar que se ingresen números negativos, espacios o caracteres extraños.

Resultados:



```
A new PowerShell stable release is available: v7.5.1
Upgrade now, or check out the release page at:
https://aka.ms/PowerShell-Release?tag=v7.5.1

PS C:\Users\luisv> cd Desktop
PS C:\Users\luisv\Desktop> python fibo.py
¿Qué número de Fibonacci deseas calcular (n)? 10
El número F(10) de la secuencia de Fibonacci es: 55
PS C:\Users\luisv\Desktop> python fibo.py
¿Qué número de Fibonacci deseas calcular (n)? 15
El número F(15) de la secuencia de Fibonacci es: 610
PS C:\Users\luisv\Desktop> python fibo.py
¿Qué número de Fibonacci deseas calcular (n)? 200
El número F(200) de la secuencia de Fibonacci es: 280571172992510140037611932413038677189525
PS C:\Users\luisv\Desktop>
```

```

PowerShell 7 (x64)
PS C:\Users\luisv\Desktop> python fibo.py
¿Qué número de Fibonacci deseas calcular (n)? 25000
[DEBUG] Entrada recibida: '25000'

El número F (25000) de la secuencia de Fibonacci es:
219543835551739361728789719148417289228498152223021557731871127306623039982943883267310466977945532854344778951126287924932107427759864934238982127116700031702303758195833
103609129477839048944610994835317536477896623484077580249370783686221102809614283525924447085974295024795917389779716235647751262119867748993044912978797302569454458871015
985618526962480184094692721968792533798209860951746361593836607992765067718042626749506787801244175219878287247486196345097306999352486464806289948617201634083726217762541
24881281835658238936811171475547846565577280497290764697852586918352378324519813894991285393472866714923971764852964557540614339146367544912483895917134
63111366674389437267869826078695826993694733293844620322484164784628106616969533371416202586412329625883384747698736328639269159478703352162342891731746210915161327957
642827991686946753866973938568721388543395668827691253289167033978959596735738359525891786556981678374415398676775162153506228598578474327975563879481946834661712956
2126117440542434702915102798877664360166812897585744089396265642829124238611310046123814278414650371177742652631522963800861498985733549162206553432881243589562482838767528
0866957146161887396654282446429461137094686918446980196032624875044012403229642652200211413566229675637538223415333791998897080651304124758742321036677825587496967438963323
643664961380931226103957879354678384278782580287252788429759993550245420157048162199750895573315081880852037989646753756752229971362236436339196695178113936201542947482459
3874336723409381728045838357307674886232976129648498042175099200883873811637124216347319672854286456837684848242693677686836778280888135886196161585831624637343362732327552
97474756972480353727979625063567322876787930440004157536448480243085486937052414964517820347449371129478269464215660734325599565755735875841763266741315151019706191944832517
439819749986146566254875843595251516514152782221210991882782957537382816869753025838953564211351196340080472763785337652644961997217464477681641834114451325479919344679587
8864549893310646083697994854625181648857277988288593879260922362597888507736945158642688176815804045153250828445893153468311287886583311992540991763847799991797925439987689811
2513498993904590424817166245312871883244364365099746836338994730194737864557120355348799791656738567199660054240466328267162556672765028694
12549646415867687598856862761032427516753856297488628469829226804403487894925748004458724290064662524859828499781439492458792866265264972647899134884985497638499802962524
2180637877875980574226647126518841802583683779468657171660689111392447419988877793699975809828803493650288777889811294753703481574971953587841672207272659743385198090659815
228978245669486793977606757566591486880896920978422383157810866125849476736047695152793249991877988265665735184297392794230171504223232002618663752888167498561
2714461724292676663868182268631496452098433691184295182860957007815373514433730887148436498978862776886575237633447764534821
4480839955613169549188562677937143917648435844783246849170485832119857130535198485855731889923504434894589095948339246350896691340898956442872524135291934193168585778
048761748443613818921897385667695943775275592131231388279743477431944154092268633257541858191595977996053363977440738568977520434776378534586571292901513722292518986
45659537543312818128869158706110313821835682157694778786799429898388113504566763880380687717624702899566977108886376588674201182777758188902985637680426667207086125
454566347073649977939296438991171498259602076764251209713693539624486981362762428578241131362502625697631059664478643090868563262147064965159771754255927856256288874776182886
436842315814581195562340152724738932326124739985258694598667126668953198451357896574430242403711592312334013691251782657924188622557294188627768976724
85813282671420014662160662927684734823836148848843768896915208176450633425113967914556457671763247767088968843917827678882219172017224191468212141892708821
026945953210588856120736265733886859475366175363807819378822094601515084811994998362588427960455735781524535484606494472109199688016107826513060762733683510203030086762173548
775277687700680712586552800533257194717286937278119474727043191204388701842373934281548349395735781523881657570036455166629368423887204786518611840180935044425
01227712773161269474733971700990749982353993128849068779213296458657447715656961712014176365975048378788348501866718841589991682832342508384065614969398862673252457977
4434949835486764988599358506303560891286566017381494478493885189286518492665697761115844994420424935514558824513414979794612868169975909931522311225531420018197889147344
26882332779146194518184115233417128511186680623717698430410850715433693511516997722946135426696091280474235508917829702134699996115304962265387488661477615392184
92570083835218943289376111049469663775103390518034337398717853672753632703663030868995573131791980484367873558679566684593106221965953389643339619134517821744496434798010
699032535268995196436613467544838162268073417703658723177596875

```

Conclusión:

Como pudimos observar, si bien, la manera de abordar el problema y su solución no son tan simples, tampoco son tan complejas a primera vista, ya que es mi primera vez que abordo el problema de Fibonacci de esta manera, ya que en años anteriores simplemente lo abordaba con recursividad para poder determinar tanto el n-ésimo número de Fibonacci, así como la lista desde el primer número hasta el último número que se hubiese ingresado.

Mientras que como observamos, este algoritmo consigue una notación asintótica de $O(\log n)$, un algoritmo genérico de Fibonacci recursivo es exponencial; es decir $O(2^n)$ por lo que para calcular $F(n)$ muy grandes como 1,000,000 tardaría muchísimo tiempo en hacerlo ya que solamente para calcular $F(50)$ se hacen millones de llamadas recursivas y fue posible calcular incluso $F(25000)$ sin que se eleve tan alto nuestro consumo de hardware en CPU y RAM.