UNIVERSIDAD DE GUADALAJARA
CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

ANÁLISIS DE ALGORITMOS - IL355 - D06
Profesor: Jorge Ernesto Lopez Arce Delgado

Alumno: Juan Pablo Solis Regin
Participación Fibonacci con/sin Programación dinámica

Juan Pablo Solis Tegin - Analisis de algoritmos

Programación Dinámica

Consiste en una tecnica de diseño de algoritmos que se inventó para la optimización en la toma de decisiones en etapas multiples. Un ejemplo puede ser la serie de Fibonacci. Es una tecnica de solución de problemas con subproblemas superpuestos.
El paradigma de la programación dinamica sugiere la solución de cada subproblema una única vez y guardar un registro de los resultados donde la solución al problema original pueda ser obtenida.

En Fibonacci pueden calcularse mediante: $F(n) = F(n-1) + F(n-2)$ para cada $n>1$ y con las dos condiciones iniciales: $F(0) = 0$, $F(1) = 1$

La programación dinámica puede interpretarse como un intercambio en el espacio y tiempo. Si el algoritmo se refina lo suficiente puede evitarse el uso adicional de memoria, para esto, debemos definir una recurrencia con respecto a las soluciones de los subproblemas que componen al problema principal
"Una solución óptima a cualquier instancia de un problema de optimización está compuesta de las soluciones óptimas de sus sub-instancias"
Para desarrollar un algoritmo bajo el paradigma se sigue lo siguiente
1.- Describir la estructura de la solución optima
2.- Definir de forma recursiva el valor de la solución optima
3.- Calcular el valor optimo usando programación dinamica
4.- Construir solución óptima a partir de la información

Código:

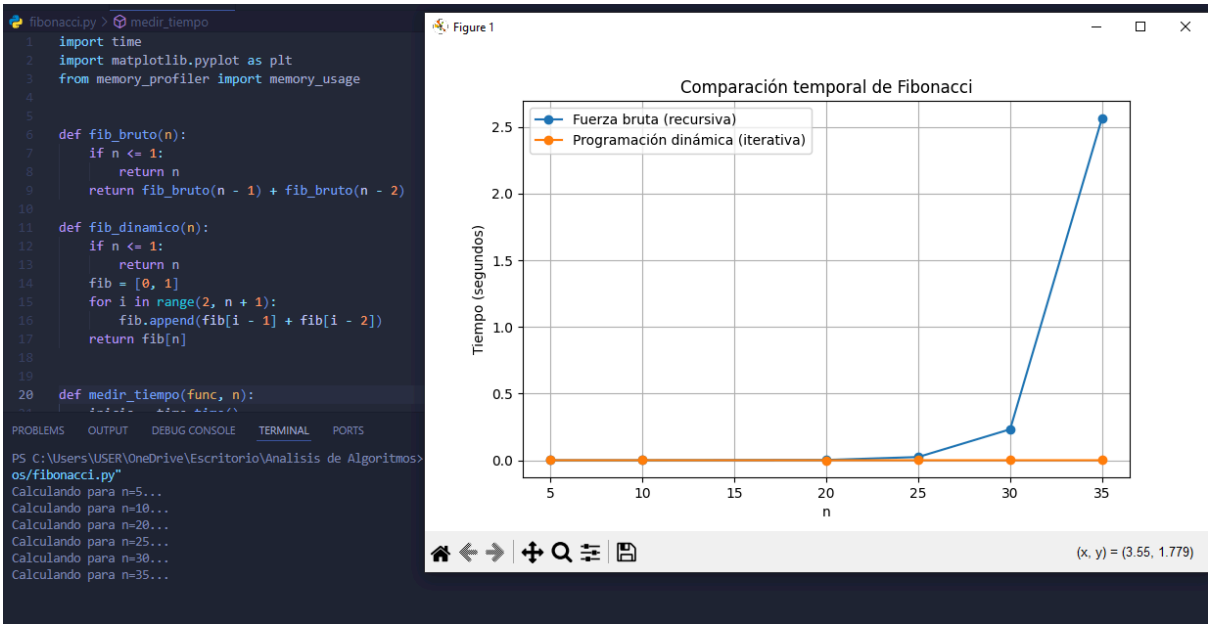
```python
import time

import matplotlib.pyplot as plt

from memory_profiler import memory_usage


def fib_bruto(n):

    if n <= 1:

        return n

    return fib_bruto(n - 1) + fib_bruto(n - 2)


def fib_dinamico(n):

    if n <= 1:

        return n

    fib = [0, 1]

    for i in range(2, n + 1):

        fib.append(fib[i - 1] + fib[i - 2])

    return fib[n]


def medir_tiempo(func, n):

    inicio = time.time()

    func(n)
```

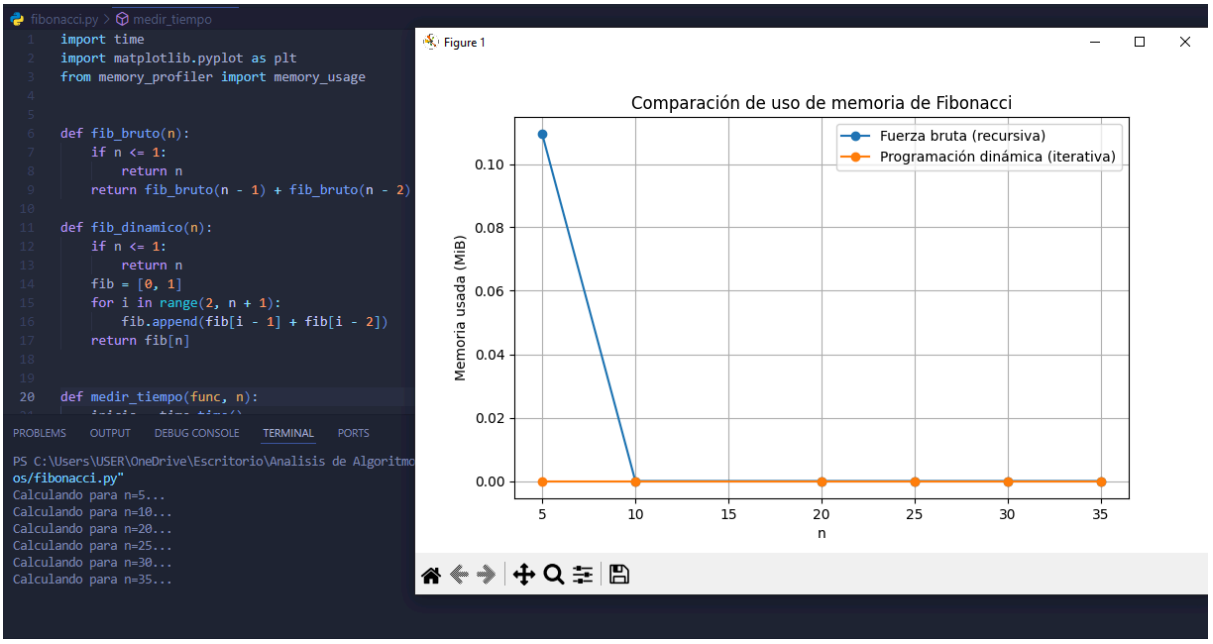
```
    fin = time.time()
    return fin - inicio

def medir_memoria(func, n):
    uso = memory_usage((func, (n,)), max_iterations=1)
    return max(uso) - min(uso)

if __name__ == "__main__":
    ns = [5, 10, 20, 25, 30, 35]
    tiempos_bruto = []
    tiempos_dinamico = []
    memoria_bruto = []
    memoria_dinamico = []

    for n in ns:
        print(f"Calculando para n={n}...")



        t_b = medir_tiempo(fib_bruto, n)
        t_d = medir_tiempo(fib_dinamico, n)
        tiempos_bruto.append(t_b)
        tiempos_dinamico.append(t_d)
```

```
        m_b = medir_memoria(fib_bruto, n)
        m_d = medir_memoria(fib_dinamico, n)
        memoria_bruto.append(m_b)
        memoria_dinamico.append(m_d)

    plt.figure(figsize=(8, 5))
    plt.plot(ns, tiempos_bruto, marker='o', label="Fuerza bruta (recursiva)")
    plt.plot(ns, tiempos_dinamico, marker='o', label="Programación dinámica (iterativa)")
    plt.xlabel("n")
    plt.ylabel("Tiempo (segundos)")
    plt.title("Comparación temporal de Fibonacci")
    plt.legend()
    plt.grid(True)
    plt.show()


    plt.figure(figsize=(8, 5))
    plt.plot(ns, memoria_bruto, marker='o', label="Fuerza bruta (recursiva)")
```

```
    plt.plot(ns, memoria_dinamico, marker='o',
label="Programación dinámica (iterativa)")

    plt.xlabel("n")

    plt.ylabel("Memoria usada (MiB)")

    plt.title("Comparación de uso de memoria de Fibonacci")

    plt.legend()

    plt.grid(True)

    plt.show()
```

Ejecución del programa:




fibonacci.py > medir_tiempo
import time
import matplotlib.pyplot as plt
from memory_profiler import memory_usage
def fib_bruto(n):
if n <= 1:
return n
return fib_bruto(n - 1) + fib_bruto(n - 2)
def fib_dinamico(n):
if n <= 1:
return n
fib = [0, 1]
for i in range(2, n + 1):
fib.append(fib[i - 1] + fib[i - 2])
return fib[n]
def medir_tiempo(func, n):
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\USER\OneDrive\Escritorio\Analisis de Algoritmo
os/fibonacci.py"
Calculando para n=5...
Calculando para n=10...
Calculando para n=20...
Calculando para n=25...
Calculando para n=30...
Calculando para n=35...
Figure 1
Comparación de uso de memoria de Fibonacci
Fuerza bruta (recursiva)
Programación dinámica (iterativa)
Memoria usada (MiB)
0.00
0.02
0.04
0.06
0.08
0.10
5
10
15
20
25
30
35
n