Universidad Autónoma de Baja California
Facultad de Ingeniería Arquitectura y Diseño

Actividad 13. Comparar los tiempos de ejecución real de un problema al utilizar programación dinámica

Daniel Rios Rodriguez

372800

Ingeniero en software y tecnologías emergentes

grupo 952

Análisis de Algoritmos

Héctor Zatarain Aceves

Ensenada, Baja California a 10 de noviembre de 2024.

Código

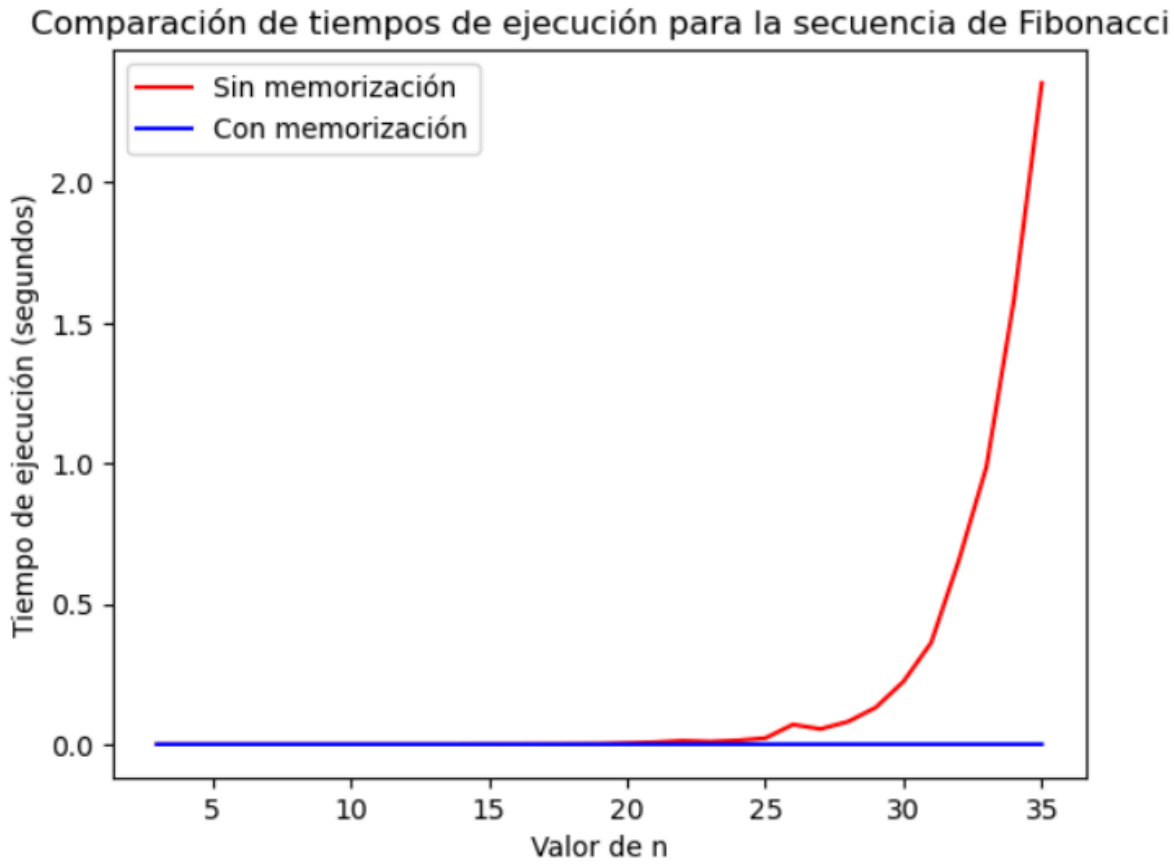
```python
import time
import matplotlib.pyplot as plt

# Función recursiva de Fibonacci sin memorización
def fib(n):
    if n <= 1:
        return n
    else:
        return fib(n - 1) + fib(n - 2)

# Función recursiva de Fibonacci con memorización (Programación Dinámica)
def fib_memo(n, memo={}):
    if n in memo:
        return memo[n]
    if n <= 1:
        return n
    else:
        memo[n] = fib_memo(n - 1, memo) + fib_memo(n - 2, memo)
        return memo[n]

# Función para medir el tiempo de ejecución
def measure_time(func, *args):
    start_time = time.perf_counter()  # Temporizador de alta precisión
    result = func(*args)
    end_time = time.perf_counter()
    execution_time = end_time - start_time
    return result, execution_time

# Valores de prueba para las funciones de Fibonacci
numbers_to_test = range(3, 36) #aqui ya se paso del minuto, por eso lo paro ahi

# Listas para almacenar los tiempos de ejecución
times_recursive = []
times_memo = []

# Medir tiempos para la función recursiva sin memorización
print("Resultados sin memorización:")
for number in numbers_to_test:
    _, exec_time = measure_time(fib, number)
    times_recursive.append(exec_time)
    print(f"fib({number}) took {exec_time:.10f} segundos")

# Medir tiempos para la función con memorización
print("\nResultados con memorización:")
for number in numbers_to_test:
    _, exec_time = measure_time(fib_memo, number)
    times_memo.append(exec_time)
    print(f"fib({number}) took {exec_time:.10f} segundos")
```

```
# Graficar los resultados
plt.plot(numbers_to_test, times_recursive, label="Sin memorización", color='red')
plt.plot(numbers_to_test, times_memo, label="Con memorización", color='blue')
plt.xlabel("Valor de n")
plt.ylabel("Tiempo de ejecución (segundos)")
plt.title("Comparación de tiempos de ejecución para la secuencia de Fibonacci")
plt.legend()
plt.show()
```

Resultados

```
Resultados sin memorización:
fib(3) took 0.0000031000 seconds to compute
fib(4) took 0.0000050000 seconds to compute
fib(5) took 0.0000026000 seconds to compute
fib(6) took 0.0000034000 seconds to compute
fib(7) took 0.0000051000 seconds to compute
fib(8) took 0.0000081000 seconds to compute
fib(9) took 0.0000127000 seconds to compute
fib(10) took 0.0000200000 seconds to compute
fib(11) took 0.0000322000 seconds to compute
fib(12) took 0.0000512000 seconds to compute
fib(13) took 0.0000826000 seconds to compute
fib(14) took 0.0001322000 seconds to compute
fib(15) took 0.0002174000 seconds to compute
fib(16) took 0.0003313000 seconds to compute
fib(17) took 0.0005332000 seconds to compute
fib(18) took 0.0008513000 seconds to compute
fib(19) took 0.0013770000 seconds to compute
fib(20) took 0.0039140000 seconds to compute
fib(21) took 0.0067628000 seconds to compute
fib(22) took 0.0121811000 seconds to compute
fib(23) took 0.0090629000 seconds to compute
fib(24) took 0.0134808000 seconds to compute
fib(25) took 0.0215337000 seconds to compute
fib(26) took 0.0699730000 seconds to compute
fib(27) took 0.0541944000 seconds to compute
fib(28) took 0.0805395000 seconds to compute
fib(29) took 0.1309578000 seconds to compute
fib(30) took 0.2233953000 seconds to compute
fib(31) took 0.3616382000 seconds to compute
fib(32) took 0.6542879000 seconds to compute
fib(33) took 0.9877731000 seconds to compute
fib(34) took 1.5852633000 seconds to compute
fib(35) took 2.3543719000 seconds to compute
```

```
Resultados con memorización:
fib(3) took 0.0000048000 seconds to compute
fib(4) took 0.0000025000 seconds to compute
fib(5) took 0.0000018000 seconds to compute
fib(6) took 0.0000007000 seconds to compute
fib(7) took 0.0000009000 seconds to compute
fib(8) took 0.0000006000 seconds to compute
fib(9) took 0.0000006000 seconds to compute
fib(10) took 0.0000005000 seconds to compute
fib(11) took 0.0000006000 seconds to compute
fib(12) took 0.0000015000 seconds to compute
fib(13) took 0.0000005000 seconds to compute
fib(14) took 0.0000007000 seconds to compute
fib(15) took 0.0000006000 seconds to compute
fib(16) took 0.0000005000 seconds to compute
fib(17) took 0.0000005000 seconds to compute
fib(18) took 0.0000005000 seconds to compute
fib(19) took 0.0000007000 seconds to compute
fib(20) took 0.0000005000 seconds to compute
fib(21) took 0.0000005000 seconds to compute
fib(22) took 0.0000005000 seconds to compute
fib(23) took 0.0000015000 seconds to compute
fib(24) took 0.0000005000 seconds to compute
fib(25) took 0.0000005000 seconds to compute
fib(26) took 0.0000005000 seconds to compute
fib(27) took 0.0000006000 seconds to compute
fib(28) took 0.0000006000 seconds to compute
fib(29) took 0.0000005000 seconds to compute
fib(30) took 0.0000006000 seconds to compute
fib(31) took 0.0000005000 seconds to compute
fib(32) took 0.0000005000 seconds to compute
fib(33) took 0.0000006000 seconds to compute
fib(34) took 0.0000007000 seconds to compute
fib(35) took 0.0000006000 seconds to compute
```



Conclusión

Al analizar las dos versiones de la función Fibonacci, podemos observar que la versión sin memorización se ralentiza significativamente conforme aumenta el valor de nnn. Esta desaceleración ocurre debido a la repetición de cálculos, ya que la función vuelve a realizar las mismas operaciones múltiples veces. En cambio, la versión con memorización se muestra considerablemente más rápida, ya que almacena los resultados de los cálculos previos y los reutiliza en lugar de recalcularlos.

Referencias

1. Blackboard. (n.d.). *Archivo*. &Copy; 1997-2015 Blackboard Inc. Todos Los Derechos Reservados. Patente De EE. UU. N.º 7,493,396 Y 7,558,853. Patentes Adicionales Pendientes. https://uabc.blackboard.com/ultra/courses/_1151422_1/outline/file/_22934777_1