1. **Describir el problema planteado y la estrategia algorítmica aplicada**
   Problema: Calcular el Fibonacci de un número
   Estrategia Algorítmica: Programación dinámica
2. **Implementar el problema plantado en un lenguaje de programación sin aplicar una estrategia algorítmica**

onacci.py > ...

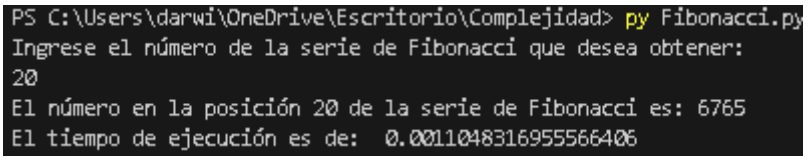
```python
import time

class Fibonacci:
    def fibonacci(self, n):
        if n < 2:
            return 1
        return self.fibonacci(n - 1) + self.fibonacci(n - 2)

fb = Fibonacci()

print("Ingrese el número de la serie de Fibonacci que desea obtener:")
n = int(input())
if n < 0:
    print("El número debe ser positivo y mayor a 0")
else:
   n = n-1
   tiempoInicial = time.time()
   print("El número en la posición", n+1, "de la serie de Fibonacci es:", fb.fibonacci(n))
   tiempoFinal = time.time()
   print("El tiempo de ejecución es de: ", tiempoFinal - tiempoInicial)
```

3. **Implementar el problema plantada en un lenguaje de programación aplicando una estrategia algorítmica**

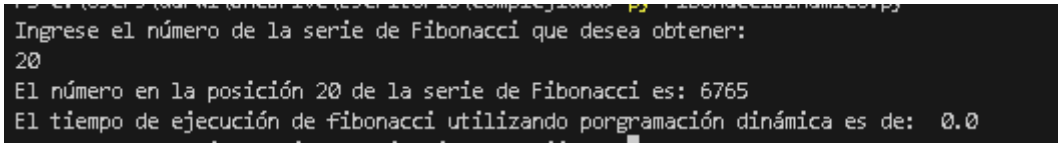
```python
import time
class FibonacciDinamico:
    def fibonacciDinamico(self, n, memoria):
        if n < 2:
            return 1
        if memoria[n] != -1:
            return memoria[n]

        memoria[n] = self.fibonacciDinamico(n - 1, memoria) + self.fibonacciDinamico(n - 2, memoria)
        return memoria[n]

fbd = FibonacciDinamico()
print("Ingrese el número de la serie de Fibonacci que desea obtener:")
n = int(input())
memoria = [-1] * (n + 1)
if n < 0:
    print("El número debe ser positivo y mayor a 0")
else:
   n = n-1
   tiempoInicial = time.time()
   print("El número en la posición", n+1, "de la serie de Fibonacci es:", fbd.fibonacciDinamico(n, memoria))
   tiempoFinal = time.time()
   print("El tiempo de ejecución de fibonacci utilizando porgramación dinámica es de: ", tiempoFinal - tiempoInicial)
```

4. **Comparar y analizar los resultados de los dos algoritmos implementados**
   Comparativa en tiempo de ejecución:

   Algoritmo sin estrategia algorítmica



   Algoritmo con estrategia algorítmica



- El algoritmo que implementa la estrategia algoritmos de programación dinámica resulta ser más eficiente en términos de tiempo

5. **Conclusiones**

- Como se puede observar en las imágenes de la comparativa es mucho más eficiente implementar la programación dinámica si lo que se necesita es un algoritmo rápido pero a su vez debemos sacrificar mas memoria debido a las características de esta estrategia que implementa una especie de memoria temporal para almacenar los resultados de subproblemas