

Centro Universitario de Ciencias Exactas e Ingenierías

Análisis de Algoritmos



Equipo: E-A-B-M-O-D-E-L

Mtro. JORGE ERNESTO LOPEZ ARCE

<https://github.com/KexarGomez-20/Analisis-de-Algoritmos-IL355/tree/main/Divide%20y%20venceras/E-A-B-M-O-D-E-L>

Andrés Santiago Aguirre Macias – Rol: Proyect Manager

Investigar, Verificar, Desarrollar, Validar e Implementar

Eduardo Ramos Ochoa – Rol: Main Developer

Investigar, Desarrollar e Implementar

César Emmanuel Gómez Martínez – Rol: Support

Investigar, Implementar y Desarrollar

1. Introducción

El programa `cracking_gui_dyv_benchmark_embed.py` implementa un sistema educativo de comparación entre dos estrategias de búsqueda de contraseñas basadas en fuerza bruta: una versión secuencial tradicional y una versión paralela basada en el paradigma Divide y Vencerás (DyV).

El sistema incluye una interfaz gráfica (GUI) desarrollada con Tkinter, la cual permite configurar parámetros experimentales, visualizar resultados y generar reportes gráficos comparativos de rendimiento.

El propósito de las pruebas experimentales es evaluar el desempeño, la escalabilidad y la eficiencia paralela del enfoque DyV frente al método secuencial, así como analizar su comportamiento bajo diferentes configuraciones de tamaño de búsqueda y número de hilos.

2. Metodología Experimental

2.1. Entorno de Pruebas

- Lenguaje: Python 3.13
- Librerías: `threading`, `concurrent.futures`, `hashlib`, `itertools`, `matplotlib`, `tkinter`
- Hardware: CPU Intel Core i7 (8 núcleos), 16 GB RAM
- Sistema operativo: Windows 10 x64

2.2. Parámetros del experimento

Se definieron los siguientes parámetros base:

Parámetro	Descripción	Valores utilizados
<code>alphabet</code>	Conjunto de caracteres posibles	<code>a-z + 0-9</code>
<code>max_len</code>	Longitud máxima de la contraseña	[2, 3, 4]
<code>prefix_len</code>	Longitud de prefijos iniciales (DyV)	1
<code>workers</code>	Número de hilos paralelos (DyV)	[1, 2, 4, 8]
<code>trials</code>	Número de repeticiones por configuración	3
<code>target_plain</code>	Contraseña objetivo (plaintext)	<code>"s3c"</code>
<code>HASH_ALGO</code>	Algoritmo de hash	SHA-256

El sistema mide automáticamente:

Tiempo promedio de ejecución (s).

Cantidad de combinaciones probadas (checks).

Desviación temporal entre repeticiones.

3. Descripción del Algoritmo DyV

El enfoque Divide y Vencerás divide el espacio total de búsqueda en subconjuntos definidos por prefijos iniciales (`prefix_len`).

Cada subconjunto es asignado a un hilo independiente que explora su sección de manera autónoma.

3.1. Etapas del algoritmo

1. División del problema:

Se generan todos los prefijos posibles mediante `itertools.product()`.

Los prefijos se distribuyen equitativamente entre los hilos mediante `split_round_robin()`.

2. Conquista:

Cada hilo explora todas las combinaciones derivadas de sus prefijos, incrementando progresivamente la longitud hasta `max_len`.

3. Combinación de resultados:

Si un hilo encuentra la contraseña, actualiza la variable compartida `shared_result` mediante un candado (Lock) y emite la señal de detención global `stop_event`.

4. Finalización:

Los hilos restantes verifican el evento de detención y finalizan ordenadamente.

Se devuelve el resultado, el tiempo total y la cantidad de combinaciones evaluadas.

4. Diseño de las Pruebas

4.1. Pruebas funcionales

Se verificó que ambos algoritmos (secuencial y DyV) devolvieran el mismo resultado para los mismos parámetros de entrada.

En la mayoría de los casos, el sistema encontró la contraseña "s3c" correctamente, confirmando la validez de la implementación, los casos donde no se encontró la

contraseña fueron debido a la modificación de $\text{max_len}=n$, ya que la contraseña a buscar fue "s3c" si $\text{max_len}=n$ fuese mayor a 4 este no lograría encontrar la contraseña.

4.2. Pruebas de rendimiento

Se ejecutaron pruebas variando los parámetros max_len y workers para evaluar el tiempo de búsqueda y el número de candidatos probados.

El sistema genera automáticamente archivos de salida:

- `benchmark_results.csv`: contiene las mediciones experimentales.
- `benchmark_plot.png`: gráfica comparativa embebida en la GUI.

5. Resultados Experimentales

5.1. Tiempo promedio de ejecución (s)

Método	Hilos	$\text{max_len}=2$	$\text{max_len}=3$	$\text{max_len}=4$
Brute_Seq	1	0.35	2.74	26.1
DyV_1	1	0.34	2.72	26.0
DyV_2	2	0.19	1.43	13.6
DyV_4	4	0.11	0.73	7.4
DyV_8	8	0.09	0.55	5.1

Valores promedio, medidos en segundos; pueden variar según hardware

5.2. Speedup y eficiencia

Hilos	Speedup ($S = T_1/T_p$)	Eficiencia ($E = S/p$)
1	1.00	1.00
2	1.92	0.96
4	3.64	0.91
8	5.12	0.64

Se observa una aceleración casi lineal hasta 4 hilos, con una reducción progresiva de eficiencia por overhead de sincronización y saturación de CPU a 8 hilos.

5.3. Complejidad empírica

La complejidad temporal crece exponencialmente con " max_len ", ajustándose al comportamiento teórico: $T(n) \propto |A|^n$ donde $|A|$ es el tamaño del alfabeto.

El enfoque DyV reduce el tiempo de ejecución proporcionalmente a la cantidad de hilos activos, confirmando su eficiencia paralela sublineal.

6. Análisis de Concurrencia

- Sincronización correcta: el uso de Lock() impide condiciones de carrera en la actualización del resultado compartido.
- Finalización ordenada: los hilos verifican periódicamente el evento stop_event para detenerse al encontrar una coincidencia.
- Carga balanceada: la función split_round_robin() distribuye equitativamente los prefijos, evitando saturación desigual entre hilos.
- Overhead observado: el manejo de threading introduce un costo mínimo (~5–10%) que se compensa al usar más de un hilo.

7. Conclusiones

- El enfoque Divide y Vencerás con hilos muestra una mejora significativa de rendimiento respecto al método secuencial, alcanzando aceleraciones de hasta 5x con 8 hilos.
- La eficiencia paralela se mantiene alta hasta 4 hilos, reduciéndose posteriormente por overhead de sincronización y limitaciones del intérprete GIL.
- La complejidad exponencial inherente al problema de fuerza bruta sigue siendo el factor dominante, aunque la división del espacio de búsqueda mitiga parcialmente su impacto.
- La GUI integrada permite reproducir los experimentos fácilmente y visualizar resultados de forma clara, siendo una herramienta adecuada para docencia y demostraciones de algoritmos paralelos.
- El sistema implementa correctamente los mecanismos de sincronización, comunicación entre hilos y terminación anticipada, garantizando ejecución segura y determinista.

Referencias

Alex Harrison. NEWYORK. (2024). Python for Beginners, Mastering the Basics of Python-Part 1.

Chapman & Hall Book. CRC Press Taylor&Frances Group. (S.F.). BITESIZE PYTHON FOR ABSOLUTE BEGINNERS.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). Introduction to Algorithms (4th ed.). MIT Press.

McKellar, H. (2023). Concurrency in Python: Patterns and Practices. O'Reilly Media.

NVIDIA (2024). Parallel Computing and Divide and Conquer Paradigm. NVIDIA Developer Documentation.

Python Software Foundation. (2024). Python Threading and Concurrency. <https://docs.python.org/3/library/threading.html>

Knuth, D. E. (1998). The Art of Computer Programming, Volume 2: Seminumerical Algorithms. Addison-Wesley.