UNIVERSIDAD DE GUADALAJARA
CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

Análisis de algoritmos
Mtro. Jorge Ernesto Lopez Arce Delgado

Avance Presentación: Divide y vencerás

Integrantes:
Braulio Hurtado Escoto
Jorge Daniel Hernández Reyes
Juan Pablo Solis Regin

Introducción

En el ámbito de la seguridad computacional, la protección de contraseñas es un aspecto fundamental para proteger nuestra información personal y no permitir el acceso no autorizado a ella. Sin embargo, muchas personas utilizan contraseñas bastante cortas o que se basan en patrones que aunque son fáciles de recordar, también son las más fáciles de adivinar y por ende su información se ve vulnerable ante los delincuentes informáticos dedicados al robo de información personal. Este tipo de contraseñas se ven opacadas cuando un atacante utiliza un diccionario de contraseñas para intentar acceder a la información, pues al tener contraseñas filtradas ya definidas, las utiliza para ahorrar tiempo y encontrar coincidencias que le permiten acceder a las cuentas.

Para el estudio de la fortaleza de las contraseñas, se implementó un algoritmo de búsqueda que mediante la técnica de divide y vencerás, permite dividir el espacio de búsqueda en subconjuntos y procesarlos de manera paralela, de modo que si se tienen que descifrar varias contraseñas dado un grupo determinado, el algoritmo utiliza hilos de la computadora para intentar descifrar las contraseñas de manera paralela, de modo que intenta averiguar distintas contraseñas a la vez realizando combinatorias de caracteres.

Desde este enfoque de divide y vencerás, se reduce el tiempo de ejecución del programa y el algoritmo se vuelve eficiente, buscando y descifrando varias contraseñas a la vez sin necesidad de esperar a que encuentre alguna para continuar con su búsqueda. El algoritmo funciona en un entorno local como simulación más que nada para corroborar las distintas contraseñas que le sean ingresadas y muestra de manera detallada la cantidad de intentos fallidos así como el tiempo que requiere el algoritmo para lograr descifrar una contraseña.

Objetivo general

Evaluar la efectividad de un algoritmo de búsqueda de contraseñas mediante la técnica de divide y vencerás, analizando su rendimiento en términos de tiempo, número de contraseñas encontradas y eficiencia en la distribución de tareas.

Objetivos particulares

- Diseñar un diccionario base y aplicar reglas de mutación para generar variantes de contraseñas.
- Dividir el espacio de búsqueda en subconjuntos para ejecutarse de forma paralela.
- Implementar workers encargados de realizar búsquedas independientes.
- Registrar intentos, tiempos de ejecución y contraseñas encontradas.
- Unificar y analizar los resultados para determinar el desempeño del algoritmo.

"***Algoritmo de búsqueda de contraseñas por diccionario y tiempo***"

El algoritmo de búsqueda de contraseñas por diccionario y tiempo es un algoritmo que tiene usos desde distintos tipos de enfoques, uno de ellos es corroborar que tan efectivas son las estrategias de búsqueda (como un diccionario) sobre conjuntos de contraseñas creadas. El algoritmo puede ser utilizado en auditorías internas de contraseñas de prueba, estudiando la resistencia y vulnerabilidad de contraseñas así como trabajar con datos de usuarios sobre los que se tenga autorización para observar qué tan frágil suelen ser las contraseñas relacionadas con datos personales.

Algoritmo abordado con divide y vencerás

El algoritmo puede ser visto desde la estrategia de divide y vencerás partiendo desde un enfoque grupal, donde distintos candidatos son divididos en subconjuntos y cada uno de los subconjuntos será “atacado” probando con distintas contraseñas hasta llegar a dar con el objetivo y lograr averiguar cuántas contraseñas se logró descifrar en cada uno de los subconjuntos, para al final unir los resultados y observar el porcentaje de contraseñas descifradas del conjunto total. Cada uno de los subconjuntos será atacado de manera paralela, lo que permitirá observar las coincidencias y las principales diferencias entre las distintas contraseñas objetivo.

Descripción de la propuesta

Se planea elaborar una lista de objetivos que se intentarán buscar, diseñar un gran diccionario con posibles candidatos, implementación de reglas de mutación así como parámetros de tiempo. Se partirá el diccionario en bloques o por criterios así como partición de los distintos objetivos. Tendremos distintos workers, cada uno de ellos con un bloque de candidatos los cuales aplicará contra los objetivos y se registraran el número de intentos y el tiempo que se tardó en ejecutar las distintas pruebas. Una vez concluidas las pruebas, se reunirán los resultados y se eliminan los duplicados, ordenándolos por tiempo de llegada al objetivo y se evaluará la efectividad del algoritmo por medio de la cantidad de objetivos que fueron descifrados correctamente. Por último se tendrán distintos criterios parada:

- Cuando se encuentran todas las contraseñas esperadas
- El diccionario es agotado
- Se alcanza un límite de pruebas/ se alcanza un límite de tiempo.

División del problema

El problema es dividido de distintas maneras con la finalidad de lograr el objetivo y descifrar la mayor cantidad de contraseñas posibles en un tiempo determinado. Las contraseñas objetivo pueden ser divididas en conjuntos por rango de palabras o por la longitud de las contraseñas objetivo. Si existen muchos objetivos, los workers revisan el diccionario por completo (no solo un subconjunto) y lo aplican en uno o varios objetivos para conseguir encontrar coincidencias.

Resolución de subproblemas

El worker tiene la tarea de iterar las distintas contraseñas candidatas para dar con el objetivo, generar variantes de contraseñas por medio del diccionario y guardar resultados en caso de dar con el objetivo

Se mantiene un registro con cada uno de los objetivos sobre cuantas pruebas fueron realizadas, tiempo probado con cada candidato, memoria utilizada y latencia con la que se verificaron los resultados.

Enumeración de reglas aplicadas en los distintos candidatos para evitar repetir contraseñas candidatas que ya fueron probadas y comenzar a realizar pruebas con un nuevo candidato hasta agotar las posibles combinaciones.

Cómo se combinan los resultados

Al finalizar el trabajo de cada worker, los resultados se integran para obtener una visión general del desempeño del algoritmo. Primero se juntan todas las contraseñas encontradas y se eliminan las repetidas. Luego, se ordenan según el tiempo en que fueron descubiertas para medir la eficiencia de cada bloque. Finalmente, se calcula el porcentaje total de contraseñas descifradas y se identifica qué parte del proceso fue más efectiva. El algoritmo concluye cuando se logran descifrar todas las contraseñas, se termina el diccionario o se alcanza el límite de tiempo o intentos.

Participación de los integrantes

Integrante	Funciones principales	Actividades específicas	Entregables / Resultados
Jorge Daniel Hernández Reyes	Diseño de presentación, visualización de resultados y edición del reporte final.	- Diseñar la presentación visual del proyecto (gráficas y visualización de tiempos de ejecución). *(24/Oct/25)* - Verificar que los resultados obtenidos sean correctos y completos, comparando los datos teóricos y prácticos. *(25/Oct/25)* - Dar formato al reporte final, organizando las secciones, tablas y figuras. *(25/Oct/25)* - Elaborar la presentación en diapositivas para la exposición del equipo. *(25/Oct/25)*	- Reporte final completo con formato uniforme. *(26/Oct/25)* - Presentación para exposición. *(26/Oct/25)* - Material visual del proyecto. *(24/Oct/25)* -Verificación de resultados de reporte. *(25/Oct/25)*

Braulio Hurtado Escoto	Implementación en Python, gestión de datos y análisis de resultados, y redacción del reporte final.	- Programar el algoritmo principal en Python, aplicando el enfoque de *Divide y Vencerás*. *(20/Oct/25)* - Implementar el módulo del "diccionario" para almacenar contraseñas no descifradas. *(21/Oct/25)* - Realizar pruebas de rendimiento y medición de tiempos utilizando distintos conjuntos de contraseñas. *(21/Oct/25)* - Elaborar gráficas comparativas entre tiempo de ejecución y cantidad de datos procesados. *(23/Oct/25)*	- Código funcional del algoritmo. *(22/Oct/25)* - Resultados experimentales y gráficas. *(22/Oct/25)* - Reporte técnico del comportamiento del programa. *(23/Oct/25)*
Juan Pablo Solis Regin	Investigación teórica, validación de resultados y redacción del reporte final.	- Investigar los fundamentos del algoritmo de búsqueda por diccionario y el enfoque de *Divide y Vencerás*. *(20/Oct/25)* - Analizar cómo se aplican las estrategias de división de subproblemas para optimizar la búsqueda de contraseñas. *(20/Oct/25)* - Documentar el funcionamiento teórico del algoritmo y la relación entre complejidad temporal y espacial. *(24/Oct/25)* - Validar los resultados obtenidos con las pruebas prácticas y contribuir a la	- Marco teórico del proyecto. *(20/Oct/25)* - Análisis de complejidad. *(20/Oct/25)* - Secciones de introducción y conclusiones del reporte. *(26/Oct/25)* -Verificación de resultados de reporte. *(25/Oct/25)*

		redacción del reporte y presentación. *(25/Oct/25)*	

Pseudocódigo

- **Constantes iniciales**

```
DEFINIR SMALL_DICTIONARY COMO [
   "password", "123456", "qwerty", "admin", "letmein",
   "secret", "welcome", "abc123", "monkey", "dragon"
]
```

- **Función auxiliar para formatear tiempo**

```
FUNCIÓN format_seconds(segundos):
   SI segundos < 1 → retornar (segundos * 1000) + " ms"
   SI segundos < 60 → retornar segundos + " s"
   SI minutos < 60 → retornar minutos + " min"
   SI horas < 24 → retornar horas + " h"
   EN OTRO CASO retornar días + " d"
FIN FUNCIÓN
```

- **Clase BruteWorker (procesa la búsqueda)**

```
CLASE BruteWorker:
   ATRIBUTOS:
      stop_event ← bandera para detener el proceso
      gui_log ← función para mostrar mensajes
      found ← diccionario con contraseñas encontradas

   MÉTODOS:
   MÉTODO stop():
      activar stop_event
   MÉTODO is_stopped():
      retornar si stop_event está activado

   MÉTODO dictionary_attack_multi(targets, wordlist, delay):
      limpiar stop_event
      iniciar temporizador
      attempts ← 0
      convertir targets en conjunto (targets_set)

      PARA cada palabra w en wordlist:
         SI is_stopped() → mostrar “detenido” y salir
         aumentar attempts en 1
         SI delay > 0 → pausar delay segundos
         SI w ∈ targets_set Y no está en found:
            calcular tiempo transcurrido
```

mostrar “contraseña encontrada”
guardar (palabra, intentos, tiempo)
CADA 50 intentos → mostrar progreso
mostrar “diccionario terminado”
retornar found

MÉTODO _chunk_worker(targets_set, charset, max_len, prefix, ...):
PARA longitud desde len(prefix) hasta max_len:
SI proceso detenido → salir
generar combinaciones de sufijos con caracteres del charset
PARA cada combinación candidate:
aumentar contador de intentos
SI candidate ∈ targets_set:
registrar contraseña encontrada y tiempo
SI intentos ≥ límite → detener proceso

MÉTODO brute_force_divide_and_conquer(targets, charset, max_len, partition_len, ...):
limpiar stop_event y resultados
iniciar temporizador
crear conjunto de targets
generar lista de prefijos de longitud partition_len

determinar número de particiones y hilos disponibles
mostrar configuración inicial

PARA cada prefijo en la lista:
asignar tarea (_chunk_worker) en hilo independiente

ESPERAR a que terminen los hilos o se cumpla el tiempo máximo
calcular tiempo total
mostrar si todas las contraseñas fueron encontradas o no
retornar resultados encontrados
FIN CLASE

- **Clase App (Interfaz gráfica y control general)**

CLASE App:
AL INICIAR:
configurar ventana principal con título y tamaño
crear campos para ingresar contraseñas y parámetros
mostrar diccionario en una pestaña
crear pestaña para fuerza bruta (divide y vencerás)
crear registro de resultados (log)

MÉTODO start_dictionary():
leer contraseñas objetivo
leer diccionario

leer retardo (delay)
SI hay un hilo activo → mostrar aviso
iniciar hilo de ejecución llamando a _run_dictionary()

MÉTODO _run_dictionary(targets, wl, delay):
ejecutar dictionary_attack_multi(targets, wl, delay)
mostrar “diccionario terminado”

MÉTODO start_bruteforce():
leer contraseñas objetivo
leer parámetros: charset, maxlen, partlen, límite intentos, hilos
calcular total estimado de combinaciones
confirmar con usuario si es muy alto
iniciar hilo llamando a _run_brute()

MÉTODO _run_brute(...):
ejecutar brute_force_divide_and_conquer con los parámetros dados
mostrar “divide y vencerás terminado”

MÉTODO start_benchmark():
leer parámetros del charset, longitudes, etc.
PARA cada longitud desde 1 hasta maxlen:
definir objetivo de prueba (peor caso)
medir tiempo de ejecución del algoritmo divide y vencerás
guardar longitud y tiempo
mostrar gráfica “n vs tiempo”

MÉTODO _show_plot(longitudes, tiempos):
generar gráfico con Matplotlib (eje X = longitud, eje Y = tiempo)

MÉTODO stop():
detener hilo activo y mostrar mensaje
FIN CLASE

- **Programa principal**

INICIO:
crear instancia de App
ejecutar interfaz principal (app.mainloop)
FIN