1. El lenguaje de programación utilizado y por qué fue seleccionado.

El lenguaje que se utilizo para este primer proyecto de la materia fue Python, por lo que este lenguaje tiene una biblioteca llamada itertools que es muy útil para manejar combinaciones y permutaciones que es una de las principales características en este problema además este lenguaje maneja una gestión de memoria automática ósea que administra la asignación y liberación de memoria.

1. Descripción del algoritmo en pseudocódigo. ¿Por qué funciona?

Imagen que contiene Escala de tiempo

Descripción generada automáticamente

El algoritmo funciona porque es eficiente por lo que se centra en reducir el espacio de búsqueda encontrando solo las letras comunes entre las columnas correspondientes de ambas cuadrillas. Esto minimiza el número de combinaciones que se deben considerar.

Además, las combinaciones se generan y ordenan una vez, lo que permite seleccionar la k-ésima contraseña de manera directa.

1. Descripción del mecanismo de backtracking:
   * ¿Cuál es el criterio que utiliza para definir que no tiene sentido continuar con la exploración de la solución porque no va a conducir a una solución?

**R//** En este problema, el backtracking no se aplica directamente. Pero, el código implementa una especie de poda al solo considerar las letras que están presentes en la cuadricula1 y cuadricula2, reduciendo así el número de combinaciones posibles.

* + ¿Cuál es la decisión que se revisa cada vez que se aplica backtracking?

**R//** El algoritmo revisa si una combinación es posible al comparar las letras presentes en las columnas correspondientes de las dos cuadriculas. Si no hay letras en común en alguna columna, el espacio de búsqueda se reduce.

1. ¿Cómo cambiaría el algoritmo si, en lugar de preguntar por el k-esimo password, se preguntara por el número total de passwords? Justifique su respuesta.

Si en lugar de encontrar la k-ésima contraseña, se preguntara por contar el número total de contraseñas posibles, simplemente se devolvería el tamaño de la lista de combinaciones posibles antes de ordenarlas ósea que no sería necesario cambiar la estructura como tal del algoritmo.

1. La definición de las clases utilizadas, en caso de haberlo hecho. Para cada clase debe definirse cuál es el dominio, i.e. los valores que son válidos, y cuáles son los métodos. Los métodos deben ser definidos siguiendo la misma estructura que se propone a continuación para las funciones.

**R//** No se utilizaron clases para este algoritmo por lo que se centra en una serie de acciones o pasos que el algoritmo sigue para resolver el problema.

1. La definición de cada función. Siguiendo la línea propuesta en CORBA[1](https://github.com/EAFIT-ST0247-Estruc-de-Datos-y-Alg-II/backtracking-uva-1262-password-juanlotero2552/blob/main/README.md#user-content-fn-1-d8e53ffa229fe10c1b670e84aad6f692) se deben incluir los siguientes elementos:



**Parámetros**:

* K (int): El índice de la contraseña deseada.
* cuadricula1, cuadricula2 (listas de strings): Las dos configuraciones de las cuadriculas.

**Retorno**:

* string: La k-ésima contraseña o "NO" si no existe.

**Excepciones**: No maneja excepciones por lo que sé asume que las entradas siempre serán válidas.

**Descripción**: Encuentra la k-ésima contraseña válida entre las posibles combinaciones de letras.



**Parámetros**: Ninguno.

**Retorno**: Ninguno.

**Excepciones**: Ninguna.

**Descripción**: Controla el flujo general del programa, incluyendo la lectura de entradas y la impresión de resultados.

1. La descripción de las variables. Se debe realizar una tabla que tenga la siguiente información:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nombre Variable** | **Tipo de Dato** | **Uso** | **Visibilidad** | **Ciclo de Vida** |
| T | int | Número de casos de prueba | Local en main | Se crea y destruye en main |
| K | int | Índice de la contraseña | Local en main | Se crea y destruye en main |
| cuadricula1 | lista de strings | Configuración de cuadricula 1 | Local en main | Se crea y destruye en main |
| cuadricula2 | lista de strings | Configuración de cuadricula 2 | Local en main | Se crea y destruye en main |
| columnas1 | lista de strings | Columnas de cuadrilla 1 | Local en encontrarKEsimaContrasena | Se crea y destruye en encontrarKEsimaContrasena |
| columnas1 | lista de strings | Columnas de cuadrilla 1 | Local en encontrarKesimaContrasena | Se crea y destruye en encontrarKEsimaContrasena |
| intersecciones | lista de listas | Letras comunes por columna | Local en encontrarKEsimaContrasena | Se crea y destruye en encontrarKEsimaContrasena |
| combinaciones | lista de tuplas | Combinaciones posibles | Local en encontrarKEsimaContrasena | Se crea y destruye en encontrarKEsimaContrasena |
| resultado | string | Contraseña k-ésima o "NO" | Local en main | Se crea y destruye en main |