**EJERCICIO 11.3**

Este código implementa una tabla hash en Python y realiza pruebas con diferentes factores de carga y secuencias de sonda. Aquí tienes una explicación detallada:

1. Clase **HashTable**:

**\_\_init\_\_(self, size=128)**: Constructor de la clase que inicializa la tabla con un tamaño predeterminado de 128, pero puedes especificar otro tamaño si lo deseas. La tabla se inicializa con listas de claves y valores, y **used\_cells** se usa para llevar un registro del número de celdas utilizadas.

**\_\_len\_\_(self):** Devuelve el número de celdas utilizadas en la tabla.

**\_\_contains\_\_(self, key):** Comprueba si una clave está presente en la tabla utilizando el método get().

**hash\_function(self, key):** Implementa una función de hash simple utilizando la función hash() de Python y toma el módulo del tamaño de la tabla para obtener el índice.

**rehash(self, old\_hash**): Resuelve colisiones mediante el uso de sonda lineal, incrementando el índice en 1 hasta encontrar una posición vacía.

**\_\_getitem\_\_(self, key):** Obtiene el valor asociado a una clave. Utiliza la función de hash y la sonda para buscar la clave y devuelve el valor correspondiente.

**\_\_setitem\_\_(self, key, value):** Asigna un valor a una clave en la tabla. Utiliza la función de hash y la sonda para encontrar la posición adecuada. Si la carga excede el 50%, se llama a **\_\_growTable()** para duplicar el tamaño de la tabla.

**\_\_growTable(self):** Duplica el tamaño de la tabla, reasigna las claves y los valores antiguos a las nuevas posiciones y reinicia **used\_cells.**

**\_\_delitem\_\_(self, key):** Elimina un elemento de la tabla. Busca la clave usando la función de hash y la sonda, y establece las celdas correspondientes a None.

2. Función **test\_hash\_table():**

- Inicializa una tabla hash y realiza pruebas con diferentes factores de carga (**load\_factors**) y secuencias de sonda (**probe\_sequences**).

- Inserta un número aleatorio de claves en la tabla y muestra el tamaño de la tabla después de la inserción.

- Intenta insertar un número adicional de claves y maneja cualquier excepción que pueda ocurrir (por ejemplo, si la secuencia de sonda se queda sin celdas para probar).

- La función **random.seed(1**) asegura que las pruebas sean reproducibles.

3. Llamada a **test\_hash\_table():**

- Llama a la función de prueba para ejecutar las pruebas con diferentes combinaciones de factores de carga y secuencias de sonda.

Este código proporciona una implementación básica de una tabla hash en Python y realiza pruebas para evaluar su comportamiento con diferentes configuraciones.

**EJERCICIO 11.4**

Este código implementa un esquema de tabla hash con resolución de colisiones mediante una sonda doble (double hashing). Aquí tienes una explicación detallada:

1. Función **multiplicativeHash(key):**

- Implementa una función de hashing multiplicativa. Utiliza una técnica de mezcla de bytes y dos números primos (prime1 y prime2). El resultado es un valor de hash.

2. Función **doubleHashProbe(start, key, size):**

- Genera un intervalo de búsqueda desde una segunda función hash de la clave. Utiliza el esquema de sonda doble para determinar las posiciones a explorar. La función **doubleHashStep(key, size)** se utiliza para obtener el tamaño del paso.

3. Función **doubleHashStep(key, size):**

- Determina el tamaño del paso para una clave dada utilizando un método de hashing multiplicativo. Utiliza dos números primos (PRIME\_1 y PRIME\_2) y la clave para calcular el tamaño del paso.

4. Función **primeBelow(n):**

- Encuentra el número primo más grande por debajo de n. Utiliza una función auxiliar is\_prime(n) para verificar si un número es primo.

5. Función **is\_prime(n):**

- Verifica si un número dado es primo. Utiliza el método clásico de verificación de primos.

6. Función **insertKeys():**

- Crea una tabla hash con tamaño 23 (tableSize). Inserta 20 claves enteras aleatorias en la tabla utilizando la función multiplicativeHash para calcular el índice inicial. En caso de colisión, utiliza el esquema de sonda doble (doubleHashProbe) para encontrar una posición adecuada.

7. Función **generate\_table():**

- Genera una tabla de inserción de 20 valores aleatorios con tamaño 31 (SIZE). Utiliza la función doubleHashProbe para manejar colisiones y mostrar el proceso de inserción.

8. Llamada a **generate\_table():**

- Llama a la función generate\_table() para generar e imprimir una tabla de inserción con 20 valores aleatorios.

En resumen, este código demuestra el uso de una tabla hash con resolución de colisiones mediante sonda doble y funciones de hash multiplicativas. La función generate\_table() muestra el proceso de inserción y cómo se manejan las colisiones en la tabla.