The time complexity for each function in the model is as follows:

def __init__(self, size=10)

- has a time complexity of O(1) as it takes a constant time to run, having no loops or recursive functions

def _hash(self, key)

- has a time complexity of O(1) as it takes a constant time to run, having no loops or recursive functions

def add(self, key)

- has a time complexity of O(1) as it takes a constant time to run, having no loops or recursive functions

def contains(self, key)

- has a time complexity of O(n) as in the worst case, the program will have to iterate over every stored value in a single index, from the statement (return key in self.table[index])

def remove(self, key)

- has a time complexity of O(n) for the same reason as def contains(self, key).

def display(self)

- has a time complexity of $O(n^2)$ as it will iterate over every index in the hash table [O(n)], and every array contained within each index [O(n)]. O(n * n) = [$O(n^2)$]

Scope

This is a program that utilizes hashing in the form of a Hash Set, a kind of Hash Table. This program is used to store generated product keys in a hash set for quick data retrieval and verification. By using a hash function, the program is able to use the user input to 'jump' straight to the relevant index in order to check if a value is stored there or not.

User Input

As the program is used to store product keys, only alphanumeric combinations of 25 characters are accepted as input into the hash set. Special characters are not allowed. The program can take any number of user inputs and display the entirety of the current hash table. Multiple values can be stored in the same index to handle collisions.

- **Alphanumeric**
- **EXACTLY 25 characters**
- **ANY number of inputs can be stored**
- **ANY number of collisions are handled by chaining**
- **Duplicates are not stored**