**Hash Tables**

insert O(1)

lookup O(1)

delete O(1)

search O(1)

    // Looping through keys will result in BigO = O(n) because of the for loop that loops over every key.

    keys() {

        // create an empty array to store each key in our hashmap

        const keysArray = [];

        console.log(this.data.length);

        // loop through the hashmap (ALL 50 indexes in the array) causing O(n)

        for (let i = 0; i < this.data.length; i++) {

            if (this.data[i]) {

                // i = the index of the array.  [0][0] = the first element in each array which is the KEY

                keysArray.push(this.data[i][0][0])

            }

        }

        return keysArray;

    }

}

note\* because we must loop through every item in a hash table to get the keys, using objects is not a very good practice because it causes the bigO = O(n)