BYU IDAHO CSE 212 | Programming with Data Structures

W06 Prepare: Reading

Table of Contents

Maps

I. MAKING THE MAP FROM THE SET

The **map** is most commonly viewed as a table. In the map below, the first column contains a unique value called the **key**. The second column contains a value (not necessarily unique) called the **value**. The key forms a set because all the values are unique. A map is created in software by associating a value with each item in the key set. In the picture below, the table shows the key and value pairs. The keys are hashed into what looks like a set. The values are then stored with each key value.

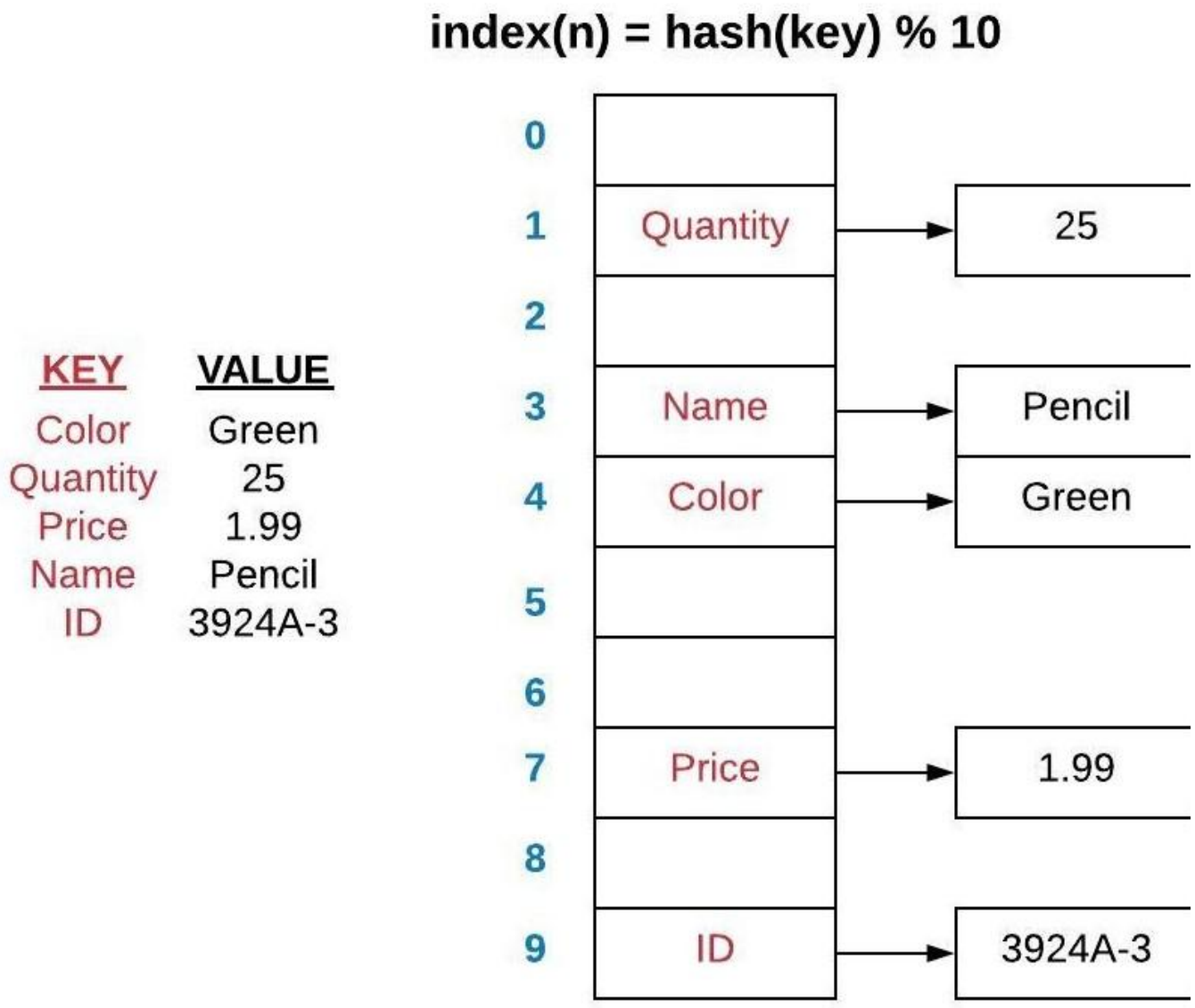

Figure 1 - Map of Colors

II. MAPS IN PYTHON

The performance of the map is the same for the set. Some programming languages call this a hashtable or a hashmap. In Python, this is called a dictionary. The dictionary (like the set) can be represented using a curly braces to specify a comma separated list of key/value pairs:

```
my_map = {"key1" : value1, "key2" : value2, "key3" : value3}
empty_map1 = {}
empty_map2 = dict()
```

Common Map Operation	Description	Python Code	Performance

put(key, value)	Adds (or replaces) a row in the map with the specified key and value.	**my_map[key] = value**	O(1) - Performance of hashing the key (assuming good conflict resolution) and assigning the value
get(key)	Gets the value for the specified key.	**my_map[key]**	O(1) - Performance of hashing the key (assuming good conflict resolution) and getting the associated value
remove(key)	Removes the row from the map containing the specified key.	**del my_map[key]**	O(1) - Performance of hashing the key (assuming good conflict resolution) and removing the associated value
member(key)	Determines if "key" is in the map.	**if key in my_map:**	O(1) - Performance of hashing the key (assuming good conflict resolution)
size()	Returns the number of items in the map.	**length = len(my_map)**	O(1) - Performance of returning the size of the map

III. COMPLEX DICTIONARY DATA

A Python dictionary can store any value including other dictionaries. Consider the following example which is data that was received from a website about the location of the International Space Station . This data is called **JSON** (JavaScript Object Notation) data and is a common data format. JSON data looks like a Python dictionary.

```
space_station = {"timestamp": 1584638006,
                 "message": "success",
                 "iss_position": {"longitude": "-149.9053",
                                  "latitude": "-35.9225"}}

longitude = space_station["iss_position"]["longitude"]
latitude = space_station["iss_position"]["latitude"]

print("Space Station at Lon: {} Lat: {}".format(longitude, latitude))
```

Notice how "iss_position" is a key in the dictionary. The value for this key is another dictionary. To get access to the "longitude" key, we need to go through the "iss_position". The use of square brackets is needed twice.

Here is another example of data that was received about the number of people that are in space.

```
astronauts = {"people": [
```

```
                {"craft": "ISS", "name": "Andrew Morgan"},
                {"craft": "ISS", "name": "Oleg Skripochka"},
                {"craft": "ISS", "name": "Jessica Meir"}],
              "message": "success",
              "number": 3}

for person in astronauts["people"]:
    print(person["name"])
```

Notice that the "people" key has a list for a value (using the square brackets). To display the "name" of each person, we need to loop through each of the dictionaries in the "people" list and display the value associated with the name "key".

IV. BUILDING SUMMARY TABLES

A map is often described as a table. If we have a large set of data that we want to summarize, we can build a summary table using a map. The summary table could contain counts, sums, minimums, maximums, or other mathematical aggregations.

```
letters = ["A", "A", "B", "G", "C", "G", "G", "D", "B"]
summary_table = dict()

for letter in letters:
        # If the letter is not in our summary table yet, add it
        if letter not in summary_table:
                # The key is the letter since we want to summarize how
                # many letters we have.  Since it the first time we
                # have seen this letter, we will set the value to 1.
                summary_table[letter] = 1

        # If the letter is in the table, then update the value
        else:
                # We want to increase the value by 1 since we have
                # already seen this letter before
                summary_table[letter] += 1

print(summary_table)
# {'A': 2, 'B': 2, 'G': 3, 'C': 1, 'D': 1}
```

Key Terms

» **JSON** - JavaScript Object Notation. A format used frequently to share data between software. JSON data uses maps and lists. The syntax of JSON is the same as Python.

» **key** - The keys in the map form a set. Each key is unique. Keys are used to lookup value from the map.

» **map** - A data structure that is based on the set. In addition to storing the unique values in the set, the map also includes a value associated with each key. The map has the same O(1) behavior as the set.

» **value** - The value associated with each key within a map. Frequently, these values are referred to as key-value pairs.