6. Hashmaps

Friday, May 6, 2022 3:05 AM

https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html

https://docs.oracle.com/javase/8/docs/api/java/util/Set.html
^ stores only keys, not values

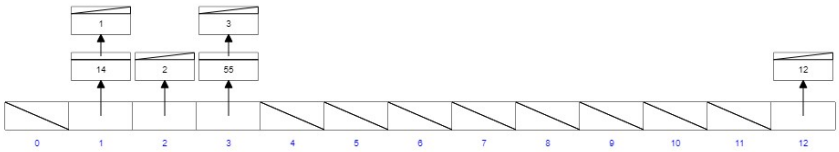
- Key-Value Pair
  key is used as index of where to find the value in memory
  - uses a hash function to generate to memory e.g. MD5

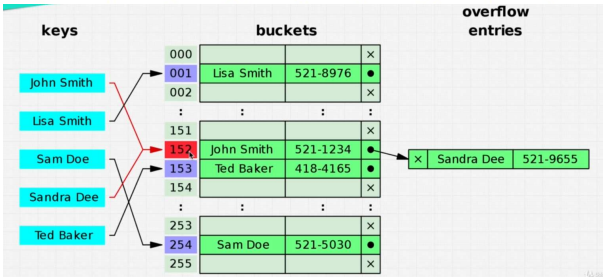
- useful in databases & caches

* optimal for insert, delete, search, O(1) on average

https://www.cs.usfca.edu/~galles/visualization/OpenHash.html

* Main problem is hash collision





- Basically multiple things will be stored at the same address memory space (152 John & Sandra)

- Hash collision will slow down reading and writing with O(n/k) -> O(n)
  k is size of your hash table

- Can use linked list to deal with collision

	Arrays	Hashmaps
search	O(n)	O(1)
look up	O(1)	O(1)
insert	O(N)	O(1)

insert O(1)

delete O(n)

First Unique character in a string practice

Input: string, s,
find the first non-repeating character

Output: return its index or -1 if it doesn't exist

e.g. s = "leetcode",
output: 0, because L is the first non-repeating element

Approach:

1. Populate string into a hashmap, for loop with a repeats value
2. Scan for first unique character, for loop + conditional
repeats 0 = false
1 = repeats from map.containsKey

map

key	value
l	1
e	2
t	1

```
class Solution {
    public int firstUniqChar(String s) {
        // Key, Value
        HashMap<Character, Integer> map = new HashMap<Character,
Integer>();

        // 0 for unique, 1 for repeating
        int repeats = 0;

        // Populate string into hashmap
        for (int i = 0; i < s.length(); i++) {
            if(map.containsKey(s.charAt(i))) {
                map.put(s.charAt(i), repeats+1);
            }
            else {
                map.put(s.charAt(i), repeats);
            }
```

O(n) time with for loops

O(1) space with 26 letters in

```
            map.put(s.charAt(i), repeats + 1);
        }
        else {
            map.put(s.charAt(i), repeats);
        }

    }

    // Scan for first unique character in the string
    for (int i = 0; i < s.length(); i++) {
      if (map.get(s.charAt(i)) == 0){
        return i;
      }
    }
    return -1;
  }
}
```

O(1) space with 26 letters in alphabet

Cleaner solution with map.getOrDefault(key, defaultValue)

```
class Solution {
  public int firstUniqChar(String s) {
    // Key, Value
    HashMap<Character, Integer> map = new HashMap<Character,
Integer>();

    // Populate string into hashmap
    for (int i = 0; i < s.length(); i++) {
      map.put(s.charAt(i), map.getOrDefault(s.charAt(i), 1) - 1);
    }

    // Scan for first unique character in the string
    for (int i = 0; i < s.length(); i++) {
      if (map.get(s.charAt(i)) == 0){
        return i;
      }
    }
    return -1;
  }
}
```

Summary

Pros

Fast lookups

Fast insert

Flexible keys

Cons

unordered

slow key iteration