Dictionaries and Sets

The Last of the Library Code

Mattox Beckman

University of Illinois at Urbana-Champaign
Department of Computer Science

Fall, 2022

Outline

Objectives

Hash maps, a.k.a. dictionaries, are the most useful data-structure.
At the end of this lecture you will know:

- How to declare and use hashmaps in your chosen language (C++ or PYTHON)
- Understand some time-complexity considerations regarding these libraries

Motivation

- Arrays are fun, but what's with all the integers?
  - Hashmaps, also called *dictionaries*, allow you to look up a value by supplying a key.
  - E.g., name / phone number, word / definition
- Hash maps can find *any object* we want quickly.
- Sets are like hash maps but we don't care about the value part.
- These, with arrays, are easily the most important data-structure you can know.

Operations

We will show these operations for C++ and PYTHON

- *Declaring* or *Creating* the map.
- *Insert* a key-value pair into the map
- *Lookup* a value given a key
- *Check* if a key is in the map
- *Query* the size
- *Iterate* over the keys or the values
- *remove* a key from the map

Creating and Inserting

- To create these in C++, you will use the `map` STL class.
  - You will need to provide the key and the value as templates.
- Insertion has two forms:
  - "array like" insertion
  - "pair" insertion using `insert`

```
#include <bits/stdc++.h>
using namespace std;

int main() {
  map<string,int> phonebook;
  phonebook["Jenni"] = 8675309;
  phonebook["emergency"] = 911;
  phonebook.insert({"Empire",5882300});
}
```

In-line initialization

- You can also initialize it at compile-time, but this is a bit rare in CP.

```
map<string,int> phonebook;
phonebook = {{"Jenni",8675390},
             {"emergency", 911},
             {"Empire", 5882300}};
```

Lookup

To lookup a specific value, you also have options:

- Use array syntax if you know the value is there.
  - *It will create the key if it doesn't already exist!*

```
cout << phonebook["Jenni"] << " and " << phonebook["H"] <<
```

Returns 8675309 and 0.

Finding Keys

- To check if the key is in the container first, use `contains`

```
if (phonebook.contains("H"))
  cout << "H is " << phonebook["H"] << endl;
```

- Finding a specific value is not supported. Program it yourself!

Size

- To get the number of pairs, use `size()`.
- To check if it's empty, use `empty()`

```
if (phonebook.empty())
  cout << "We don't know anyone." << endl;
else
  cout << "There are " << phonebook.size() << " entries." <
```

Iteration

- To loop over all the keys, we have iterators.
- Note that the order of the keys is arbitrary!
- Also note that the iterator return pairs!

```
for(auto it = phonebook.begin();
    it != phonebook.end();
    ++it)
  cout << it->first << " has phone number " << it->seco
```

Sets

- Use `unordered_set` for fast set operations.
- Use `set` if you want to retrieve the elements in a sorted order.

```
#include <bits/stdc++.h>
using namespace std;

int main() {
  unordered<string> people;
  phonebook.insert("Jenni");
  phonebook.insert("emergency");
  phonebook.insert("Empire");
}
```

Creating and Inserting

- To create in PYTHON, you can initialize an empty version or prepopulate.

```
phonebook = {}

phonebook["Jenni"] = 8675309
phonebook["emergency"] = 911
phonebook["Empire"] = 5882300
```

Also

```
phonebook = {"Jenni":8675390,
             "emergency": 911,
             "Empire": 5882300}
```

Lookup and finding keys

To lookup a specific value, you also have options:

- Use array syntax if you know the value is there.
  - *It will raise an exception if the key doesn't already exist!*

```
if "H" in phonebook:
    print(f"{phonebook['Jenni']} and {phonebook['H']}")
else:
    print(f"{phonebook['Jenni']}")
```

Finding Values

- Unlike C++, you can get the values in a dictionary easily:

```
for i in phonebook.values():
    print(phonebook[i])
```

Size

- To get the number of pairs, use `len()`.

```
print(f"There are {len(phonebook)} entries.")
```

Iteration

- To loop over all the keys, we have iterators.
- Note that the order of the keys is arbitrary!

```
for k in phonebook:
    print(k)
```

Sets

- For sets you have to call the `set()` function to start.
- Use member function `add()` to insert.
- We also have nice utilities like `intersection()`, `difference()`, etc.

Details

- In C++, sets are not hashmaps, they typically use red-black trees.
  - So, $\mathcal{O}(\log_2 n)$ access time.
- In PYTHON it uses open addressed hashing with random probing for collision resolving.