Hash Tables, Maps, and Problems Requiring Them

Josh Ganschow

Dakota State University Computer Club

2.18.25

Introduction

- A hash table is a data structure that uses a hashing algorithm to decide where a specific data element will be stored in the table (array).
- A hashmap is a specific implementation of a hash table in certain programming languages. A hashmap, like a hash table, stores a key and the values associated with it by hashing the key and deciding where it should be placed.
- In Python, a `dict` is the implementation of a hashmap.

How Hash Tables Work

- A hash table is a data structure that maps keys to values using a hash function.
- A hash function converts a key into an index in an underlying array.
- Operations like insertion, deletion, and lookup are O(1) on average.

How Hash Tables Work

- A hash table is a data structure that maps keys to values using a hash function.
- A hash function converts a key into an index in an underlying array.
- Operations like insertion, deletion, and lookup are O(1) on average.

Example Hash Function:

$$\text{index} = \text{hash}(\text{key}) \mod \text{table size} \tag{1}$$

How Hash Tables Work

- A hash table is a data structure that maps keys to values using a hash function.
- A hash function converts a key into an index in an underlying array.
- Operations like insertion, deletion, and lookup are O(1) on average.

Example Hash Function:

$$\text{index} = \text{hash}(\text{key}) \mod \text{table size} \tag{1}$$

Collision Handling Strategies:

- Chaining – Use linked lists at each index.
- Linear Probing - Check the next spot sequentially after hashing.

Common Applications

- Frequency counting (e.g. counting elements in an array)

Common Applications

- Frequency counting (e.g. counting elements in an array)
- Duplicate detection (e.g. checking if an array has duplicates)

Common Applications

- Frequency counting (e.g. counting elements in an array)
- Duplicate detection (e.g. checking if an array has duplicates)
- Index mapping (e.g. storing positions of elements for quick lookup)

Common Applications

- Frequency counting (e.g. counting elements in an array)
- Duplicate detection (e.g. checking if an array has duplicates)
- Index mapping (e.g. storing positions of elements for quick lookup)
- Two sum problems (efficiently checking if two numbers sum to a target

Show Me the Code!

Of course, programmers don't want to write a hashing function and logic to handle the operations every time they want to use these very useful data structures!

Show Me the Code!

Of course, programmers don't want to write a hashing function and logic to handle the operations every time they want to use these very useful data structures! Thankfully, they are implemented in most programming languages with simple syntax and no need to get to the nitty gritty.

Show Me the Code!

Of course, programmers don't want to write a hashing function and logic to handle the operations every time they want to use these very useful data structures! Thankfully, they are implemented in most programming languages with simple syntax and no need to get to the nitty gritty.

Python

```python
hashmap = {} #declaring
hashmap["apple"] = 5 #inserting
hashmap["banana"] = 3
hashmap["cherry"] = 7
del hashmap["cherry"] #deleting
```

C++ Code

C++

```cpp
#include <iostream>
#include <unordered_map>

int main() {
 std::unordered_map<std::string, int> hashmap;

 hashmap["apple"] = 5;
 hashmap["banana"] = 3;
 hashmap["cherry"] = 7;

 hashmap.erase("cherry");
}
```

Leetcode Time!

It's now time to do a couple of Leetcode problems together

- Contains Duplicate (leetcode.com/problems/contains-duplicate/description)
- Two Sum (leetcode.com/problems/two-sum/description)

The Negatives

Of course, not everything about hashtables can be perfect. The main downside is that you cannot use them when sorted order is required.

Conclusion

Any Questions?

Other problems to try: leetcode.com/problem-list/hash-table/