UCLA Samueli Computer Science

CS32: Introduction to Computer Science II

Discussion Week 8

Yichao (Joey)

Feb. 28, 2019

Credit: Carey Nachenberg, Junheng Hao

Outline Today

UCLA Samueli Computer Science

- Hash Table

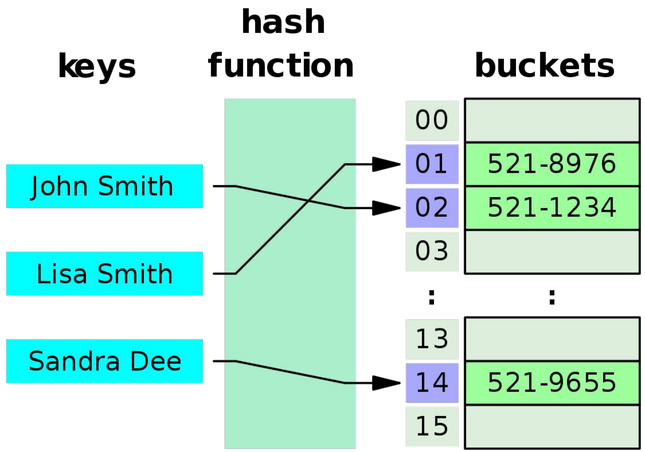
Announcements

UCLA Samueli Computer Science

- Homework 4 is due 11:00 PM Tuesday, March 3.

Hash Tables

UCLA Samueli Computer Science



In computing, a **hash table** (**hash map**) is a data structure that implements an **associative array abstract data type**, a structure that can **map keys to values.** A hash table uses a hash function to compute an *index*, also called a *hash code*, into an array of *buckets* or *slots*, from which the desired value can be found.

Hash Tables

UCLA Samueli Computer Science

The idea of hashing is to distribute the entries (key/value pairs) across an array of *buckets*. Given a key, the algorithm computes an *index* that suggests where the entry can be found:

```
index = f(key, array_size)
```

Often this is done in two steps:

```
hash = hashfunc(key)
index = hash % array_size
```

In this method, the *hash* is independent of the array size, and it is then *reduced* to an index (a number between `0` and `array_size - 1`) using the modulo operator (`%`).

Hash Tables

Open vs Closed Addressing

Open Addressing

Also known as **closed hashing**.

Collisions are dealt with by searching for **another empty buckets** within the hash table array itself.

Benefits:

- Better memory locality and cache performance. All elements laid out linearly in memory.
- Performs better than closed addressing when the number of keys is known in advance and the churn is low.

0:
1: 1
2: 2
3: 2
4: 2
5: 4
6:
7: 7
8: 7
9: 9

Closed Addressing

Also known as **open hashing**.

A key is always stored in the bucket it's hashed to. Collisions are dealt with using **separate data structures** on a per-bucket basis.

0:
1: 1
2: 2 2 2
3:
4: 4
5:
6:
7: 7 7
8:
9: 9

Benefits:

- Easier removal (no need for deleted markings)
- Typically performs better with high load factor.

Hash Tables
Operations

UCLA Samueli Computer Science

- Insert
- Remove
- Search

- The complexity depends on your hash tables.
- Closed Hashing
  - Fixed number of buckets
  - All operations are **O(n)** with a small constant of proportionality
- Open Hashing
  - Consider *#entries / #buckets*
  - Almost **O(1)** for all operations

Hash Tables

Pretty much any time you want to map keys to values with constant time for lookup/add/remove. If you find yourself **looping through a list to find an element**, think if there's a way to store the elements with keys in **a hash table** (aka hash map, aka dictionary) instead.

```
class Contacts {
 List<Person> contacts = new ArrayList<Person>();

 void addContact(Person p) {
   contacts.add(p);
 }
 // O(n) time
 Person getContact(String phoneNumber) {
   for (Person p : contacts) {
     if (p.getPhoneNumber().equals(phoneNumber)) return p;
   }
   return null;
 }
 ...
}
```

```
class Contacts {
 List<Person> contacts = new ArrayList<Person>();
 Map<String, Person> phoneNumberToPerson = new HashMap<String, Person>();

 void addContact(Person p) {
   contacts.add(p); // assuming this list is still needed somewhere else
   phoneNumberToPerson.put(p.getPhoneNumber(), p);
 }

 // O(1) time!
 Person getContact(String phoneNumber) {
   return phoneNumberToPerson.get(phoneNumber);
 }
 ...
}
```

Hash Tables

Simple question

UCLA Samueli Computer Science

Que – 1. Given the following input (4322, 1334, 1471, 9679, 1989, 6171, 6173, 4199) and the hash function x mod 10, which of the following statements are true?

i. 9679, 1989, 4199 hash to the same value
ii. 1471, 6171 has to the same value
iii. All elements hash to the same value
iv. Each element hashes to a different value
(A) i only
(B) ii only
(C) i and ii only
(D) iii or iv

Hash Tables

Simple question

Que – 2. The keys 12, 18, 13, 2, 3, 23, 5 and 15 are inserted into an initially empty hash table of length 10 using open addressing with hash function h(k) = k mod 10 and linear probing. What is the resultant hash table?

0	
1	
2	2
3	23
4	
5	15
6	
7	
8	18
9	

(A)

0	
1	
2	12
3	13
4	
5	5
6	
7	
8	18
9	

(B)

0	
1	
2	12
3	13
4	2
5	3
6	23
7	5
8	18
9	15

(C)

0	
1	
2	12, 2
3	13, 3, 23
4	
5	5, 15
6	
7	
8	18
9	

(D)

Hash Tables

Simple question

Que – 3. A hash table of length 10 uses open addressing with hash function h(k)=k mod 10, and linear probing. After inserting 6 values into an empty hash table, the table is as shown below.

Which one of the following choices gives a possible order in which the key values could have been inserted in the table?

(A) 46, 42, 34, 52, 23, 33
(B) 34, 42, 23, 52, 33, 46
(C) 46, 34, 42, 23, 52, 33
(D) 42, 46, 33, 23, 34, 52

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

Hash Tables

FNV-1

- Hash functions: Take a “key” and map it to a number
- Requirement for hash function: should return the same value for the same key
- Good hash functions:
  - Spreads out the values: two different key are likely to results in different hash values. → **Avoid confliction**
  - Compute each value quickly.
- Example: **FNV-1**


```
unsigned int FNV-1(string s) {
  unsigned int h = 2166136261U;
  for (int k = 0; k != s.size(); k++)
  {
    h += s[k];
    h *= 16777619;
  }
  return h;
}
```

offset_basis

FNV_prime

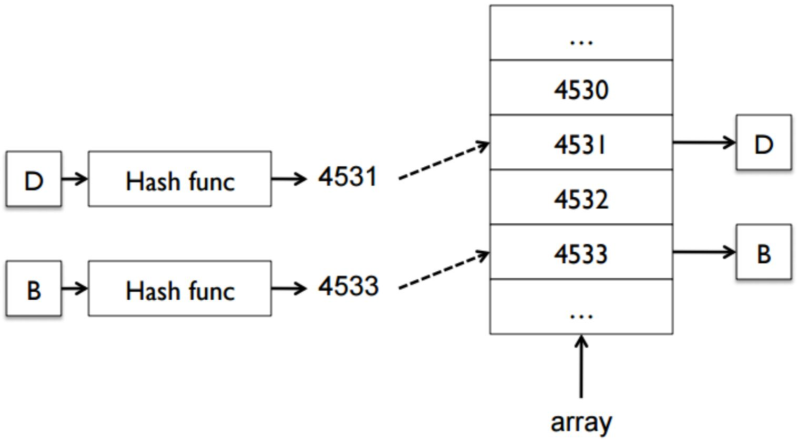
Fowler–Noll–Vo (FNV) is a non-cryptographic hash function created by Glenn Fowler, Landon Curt Noll, and Kiem-Phong Vo.
http://www.isthe.com/chongo/tech/comp/fnv/#FNV-param

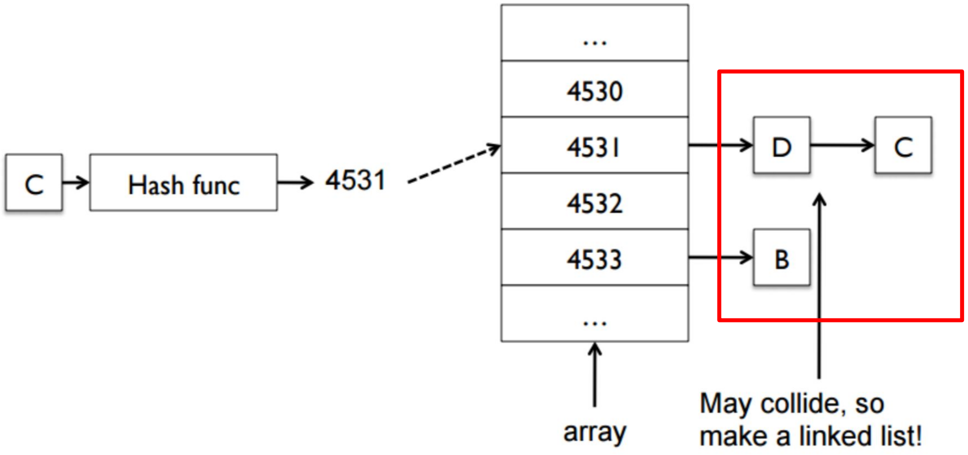
Hash Tables

Examples

- Example: Use a hash table to store people.
- Use a linked list to collision in the hash function.

*"You should almost **NEVER** assume that collisions are impossible!!!" --David Smallberg*





Hash Tables Problem

Word counting and sorting in a document

UCLA Samueli Computer Science

- Question: We have ***n*** words in a document, whose vocabulary size is ***v***. Count top k frequent words in a document.

Two steps: counting + sorting

- The most efficient way to count the frequency for all words takes **O(**n**)** time complexity.
- After getting the frequency of each word, the most efficient way to get the top k frequent words takes **O(**$v \log k$**)** time complexity.
- Totally the entire procedure takes **O(**$n + v \log k$**)**.

Hash Tables Problem

The very first question in LeetCode - TwoSum

UCLA Samueli Computer Science

- Given an array of integers, return indices of the two numbers such that they add up to a specific target.
- You may assume that each input would have exactly one solution, and you may not use the same element twice.
- Example: Given `nums = [2, 11, 7, 15]` and `target = 9`. Because `nums[0] + nums[2] = 2 + 7 = 9`, return `[0, 2]`.

1. Two Sum

Hash Table in STL

Map, multimap, unordered_map, HashMap

UCLA Samueli Computer Science

- Useful functions of STL `map`, `multimap`, `unordered_map`
  - `size()`, `begin()`, `end()`, `empty()`
  - `insert(keyvalue, mapvalue)`
  - `find()`
  - `operator[]`
- Internal implementation:
  - `map/multimap`: Red-black tree
  - **`unordered_map`**: Hash table

```cpp
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        unordered_map<int, int> m;
        for(int i = 0; i < nums.size(); i++)
        {
            if(m.find(target-nums[i]) != m.end())
                 return {m[target-nums[i]], i};
            m[nums[i]] = i;
        }
    }
 };
```

Group Exercises: Worksheet

UCLA Samueli Computer Science

- Exercise problems from **Worksheet 8** (see “LA worksheet” tab in CS32 website). Answers will be posted next week.
- Questions for today: 1, 4, 2 (simplified 2Sum), 8 (3sum)

Question 1

Given an array of distinct elements and a range [low, high], use a hash table to output all numbers in the range that are not in the array. Print the missing elements in sorted order. (F.Y.)

```
Example:

Input:  arr[] = {10, 12, 11, 15}, low = 10, high = 15
Output:     13, 14

Input:  arr[] = {1, 14, 11, 51, 15}, low = 50, high = 55
Output:     50, 52, 53, 54
```

Question 2

Given a string, find the first non-repeating character in it and return its index. If it doesn't exist, return -1. You may assume the string contains only lowercase letters. Use a hash table to solve this problem.

Examples:
s = "leetcode"
return 0
s = "loveleetcode",
return 2

Question 3

Given an array of integers and a target sum, determine if there exists two integers in the array that can be added together to equal the sum.

Examples:

Input: arr[] = [4, 8, 3, 7, 9, 2, 5], target = 15
Output: true
Explanation: 8 and 7 add up to the target sum 15

Input: arr[] = [1, 3, 5, 2, 4], target = 10
Output: false
Explanation: No combination of two numbers in the array sum to 10

Question 4

Write a function, sum3, that takes in an array of integers and determines whether there exists exactly three elements in the array that sum to 0. Return true if three such elements exist and false if not. No repeated elements are allowed. Your function must run faster than the brute force $O(N^3)$.

i.e [1,2,3,4,5,6] -> False
[1,-1,2,-2] -> False
[1,2,-3, 6, 8] -> True