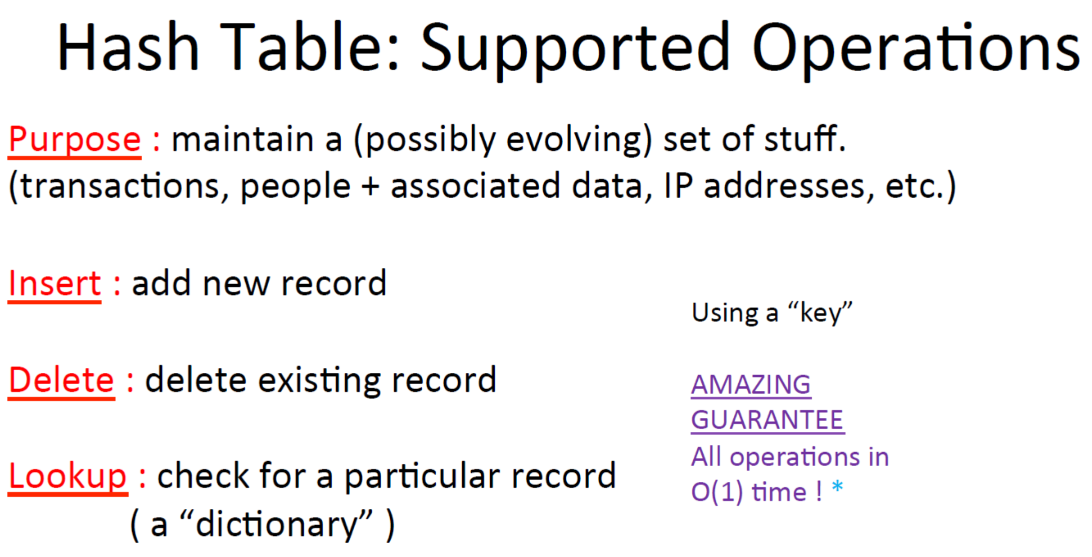
Part14-Hashing: the basics

Key videos	https://www.coursera.org/learn/algorithms-graphs-data-structures/lecture/b2Uee/hash-tables-operations-and-applications https://www.coursera.org/learn/algorithms-graphs-data-structures/lecture/Ob0K7/hash-tables-implementation-details-part-i
Note	Introduction to hash table basics --- the supported operations, the types of problems they're useful for, and an overview of implementation approaches
Period	@2020/05/03 → 2020/05/05

Note

▼ Hash Tables: Operations and Applications

- 基本操作与性质：实现Insert/Delete/Lookup仅需要O(1)的操作



Hash Table: Supported Operations

Purpose : maintain a (possibly evolving) set of stuff. (transactions, people + associated data, IP addresses, etc.)

Insert : add new record

Delete : delete existing record

Lookup : check for a particular record (a "dictionary")

Using a "key"

AMAZING GUARANTEE All operations in O(1) time ! *

- 应用
    - 去重（De-duplicate）

Given : a “stream” of objects. → Linear scan through a huge file → Or, objects arriving in real time

Goal : remove duplicates (i.e., keep track of unique objects)
-e.g., report unique visitors to web site
- avoid duplicates in search results

Solution : when new object x arrives
- lookup x in hash table H
- if not found, Insert x into H

- 2Sum问题：先进行sort复杂度O(nlogn)然后进行n次查找即可

Input : unsorted array A of n integers. Target sum t.

Goal : determine whether or not there are two numbers x,y in A with

x + y = t

Naïve Solution : $\theta(n^2)$ time via exhaustive search

Better : 1.) sort A ($\theta(n \log n)$ time) 2.) for each x in A, look for t-x in A via binary search ← $\theta(n \log n)$

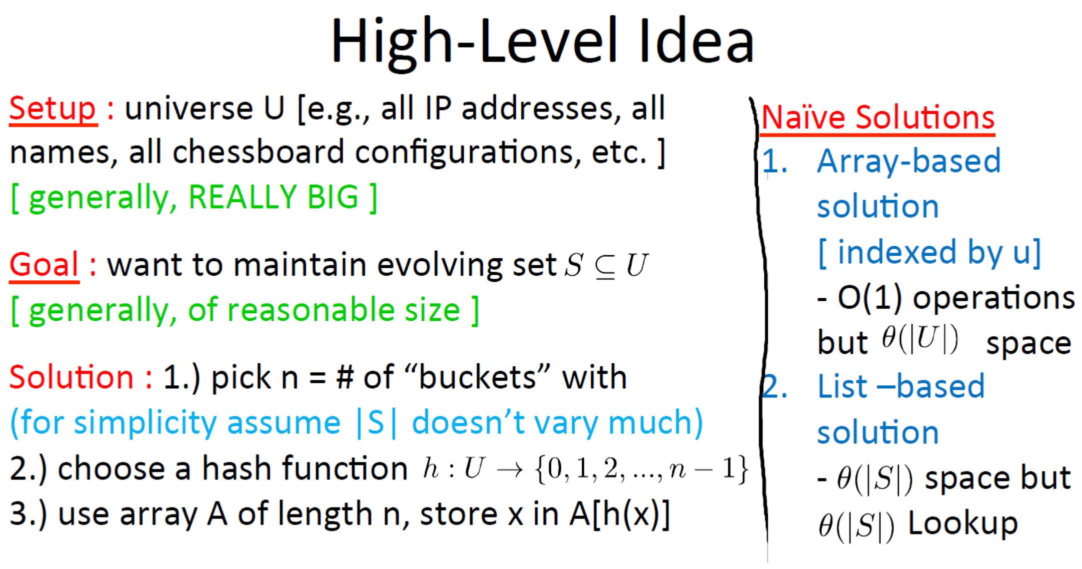
Amazing : 1.) insert elements of A into hash table H ← $\theta(n)$ time 2.) for each x in A, Lookup t-x ← $\theta(n)$ time

- 其他应用

- Historical application : symbol tables in compilers
- Blocking network traffic
- Search algorithms (e.g., game tree exploration)
  - Use hash table to avoid exploring any configuration (e.g., arrangement of chess pieces) more than once
- etc.

▼ Hash Tables: Implementation Details, Part I

- 实现方法



High-Level Idea

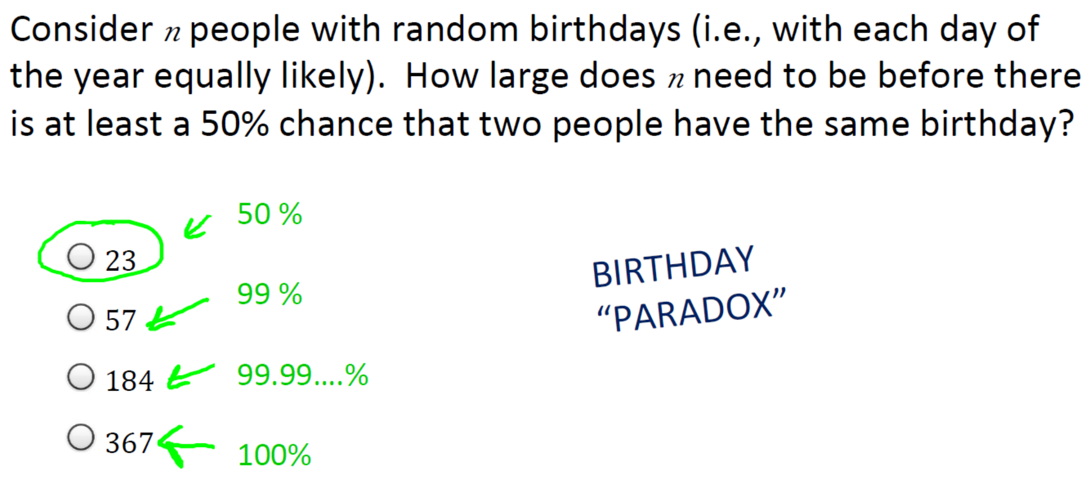
Setup : universe U [e.g., all IP addresses, all names, all chessboard configurations, etc.]
[generally, REALLY BIG]

Goal : want to maintain evolving set $S \subseteq U$
[generally, of reasonable size]

Solution : 1.) pick n = # of "buckets" with
(for simplicity assume |S| doesn't vary much)
2.) choose a hash function $h: U \rightarrow \{0,1,2,\ldots,n-1\}$
3.) use array A of length n, store x in A[h(x)]

Naïve Solutions
1. Array-based solution
[indexed by u]
- O(1) operations but $\theta(|U|)$ space
2. List –based solution
- $\theta(|S|)$ space but $\theta(|S|)$ Lookup

生日悖论，如果有超过23人，就有50%概率有两人同天生日

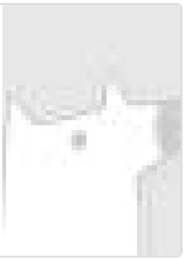


Consider n people with random birthdays (i.e., with each day of the year equally likely). How large does n need to be before there is at least a 50% chance that two people have the same birthday?

[✓ 23] 50 %
57 99 %
184 99.99....%
367 100%

BIRTHDAY "PARADOX"

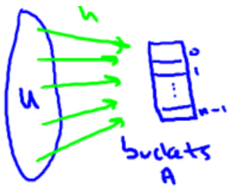
如果一个班有50个人，其中有两个人生日相同的概率是97%？为什么
方法一：1 - (364/365 * 363/365 * 362/365 * ......317/365 * 316/365)，结果约等于0.97037。含义是，2个人生日不同的概率是364/365；这时再加一个人，这与前两人生日都不同的概率是363/365；而这三人生日各不相同的概率是
知 https://www.zhihu.com/question/19691577/answer/19157843



▼ Hash Tables: Implementation Details, Part II

- 冲突解决：chaining/open addressing策略

Collision: distinct $x, y \in U$ such that $h(x) = h(y)$



Solution # 1 : (separate) chaining
-keep linked list in each bucket
- given a key/object x, perform Insert/Delete/Lookup in the list in A[h(x)]

Linked list for x

Bucket for x

Solution #2 : open addressing. (only one object per bucket)
-Hash function now specifies probe sequence $h_1(x), h_2(x), ..$
(keep trying till find open slot)
- Examples : linear probing (look consecutively), double hashing

Use 2 hash functions

- 哈希函数的好坏判断

Note : in hash table with chaining, Insert is $\theta(1)$
$\theta(list\ length)$ for Insert/Delete.

Insert new object x at front of list in A[h(x)]

Equal-length lists

could be anywhere from $\boxed{m/n}$ to m for $\boxed{m}$ objects

All objects in same bucket

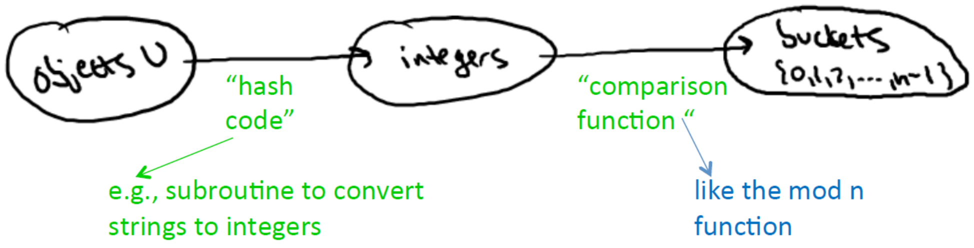
Point : performance depends on the choice of hash function!
(analogous situation with open addressing)

Properties of a "Good" Hash function
1. Should lead to good performance => i.e., should "spread data out" (gold standard – completely random hashing)
2. Should be easy to store/ very fast to evaluate.

- bucket数的选择

质数；不要过于接近2和10的次幂

Quick-and-Dirty Hash Functions



How to choose n = # of buckets

1. Choose n to be a prime (within constant factor of # of objects in table)
2. Not too close to a power of 2
3. Not too close to a power of 10

Reference

- GeeksforGeeks

Hashing Data Structure - GeeksforGeeks

Hashing is an important Data Structure which is designed to use a special function called the Hash function which is used to map a given value with a particular key for

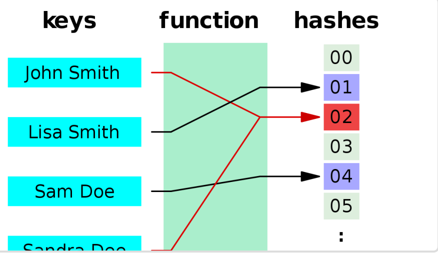
https://www.geeksforgeeks.org/hashing-data-structure/

- 哈希函数

Hash function

A hash function is any function that can be used to map data of arbitrary size to fixed-size values. The values returned by a hash function are called hash values, hash

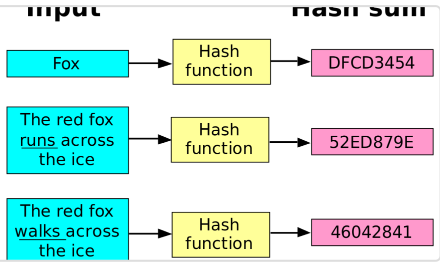
https://en.wikipedia.org/wiki/Hash_function



散列函數

（英語：Hash function）又称 散列算法、 哈希函数，是一种从任何一种数据中创建小的数字"指纹"的方法。散列函数把消息或数据压缩成摘要，使得数据量变小，将数据的格式

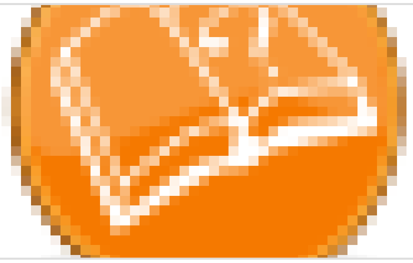
https://zh.wikipedia.org/wiki/%E6%95%A3%E5%88%97%E5%87%BD%E6%95%B8



散列

此條目需要 精通或熟悉相关主题的编者参与及协助编辑。 請邀請適合的人士改善本条目。更多的細節與詳情請參见討論頁。 雜湊（英語：Hashing）是電腦科学中一種对資料的处

W https://zh.wikipedia.org/wiki/%E6%95%A3%E5%88%97

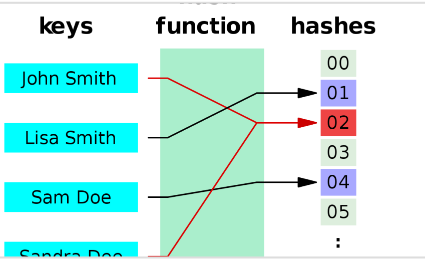


- 哈希函数

Hash function

A hash function is any function that can be used to map data of arbitrary size to fixed-size values. The values returned by a hash function are called hash values, hash

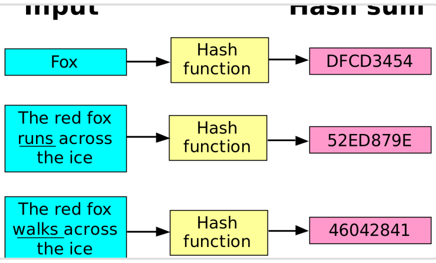
W https://en.wikipedia.org/wiki/Hash_function



散列函數

（英語：Hash function）又称 散列算法、 哈希函数，是一种从任何一种数据中创建小的数字"指纹"的方法。散列函数把消息或数据压缩成摘要，使得数据量变小，将数据的格式

W https://zh.wikipedia.org/wiki/%E6%95%A3%E5%88%97%E5%87%BD%E6%95%B8



- 视频

What is Hashing? Hash Functions Explained Simply

What is hashing? In this video we explain how hash functions work in an easy to digest way. Hashing is the process of converting an input of any length into ...

https://www.youtube.com/watch?v=2BldESGZKB8

- Universal hashing

Universal hashing

In mathematics and computing, universal hashing (in a randomized algorithm or data structure) refers to selecting a hash function at random from a family of hash functions with a certain mathematical property (see definition below). This guarantees a low number of collisions in

W https://en.wikipedia.org/wiki/Universal_hashing