



Beyond_2016

私信

关注

TA的个人主页 >

原创	粉丝	喜欢	评论
124	42	38	17

等级： 博客 5 访问： 12万+
积分： 2274 排名： 3万+
勋章：

最新文章

python assert的作用
【经典】吴恩达《机器学习》课程
2019届华为笔试题（软件卷）
Python是如何进行内存管理的
动态类型语言&&静态类型语言

分类专栏

	大数据、云计算	11篇
	机器学习	25篇
	深度学习	10篇
	Python	7篇
	Linux	2篇

展开

归档

2019年8月	1篇
2018年9月	1篇
2018年8月	38篇
2018年7月	48篇
2018年4月	17篇
2017年12月	4篇
2017年11月	4篇
2017年10月	15篇

展开

热门文章

常见的hash算法及其原理
阅读量 32708
ValueError: Input contains NaN, infinity or a value too large for dtype('float32').
阅读量 8567
2019届华为笔试题（软件卷）
阅读量 5579

常见的hash算法及其原理

2018年07月30日 15:47:13 Beyond_2016 阅读量 32813 更多

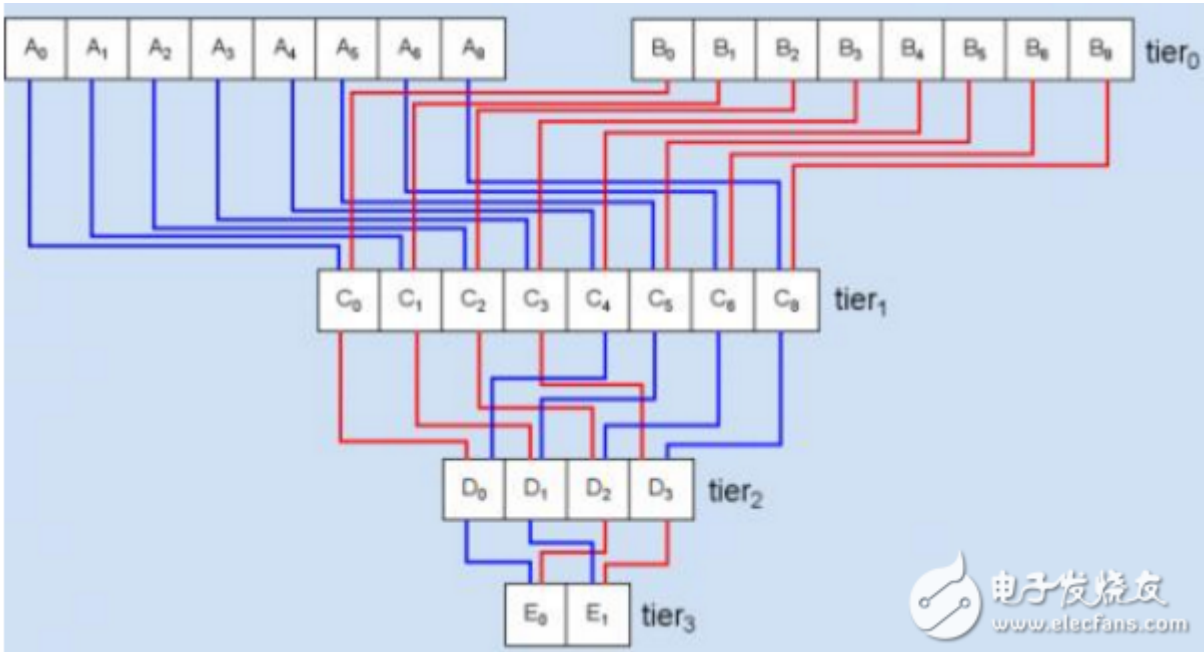
版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。
本文链接：https://blog.csdn.net/Beyond_2016/article/details/81286360

Hash，一般翻译做“散列”，也有直接音译为“哈希”的，就是把任意长度的输入（又叫做预映射， pre-image），通过散列算法，变换成固定长度的输出，该输出就是散列值。这种转换是一种压缩映射，也就是，散列值的空间通常远小于输入的空间，不同的输入可能会散列成相同的输出，而不可能从散列值来唯一的确定输入值。简单的说就是一种将任意长度的消息压缩到某一固定长度的消息摘要的函数。

哈希表是根据设定的哈希函数H（key）和处理冲突方法将一组关键字映射到一个有限的地址区间上，并以关键字在地址区间中的象作为记录在表中的存储位置，这种表称为哈希表或散列，所得存储位置称为哈希地址或散列地址。作为线性数据结构与表格和队列等相比，哈希表无疑是查找速度比较快的一种。

通过将单向数学函数（有时称为“哈希算法”）应用到任意数量的数据所得到的固定大小的结果。如果输入数据中有变化，则哈希也会发生变化。哈希可用于许多操作，包括身份验证和数字签名。也称为“消息摘要”。

简单解释：哈希（Hash）算法，即散列函数。它是一种单向密码体制，即它是一个从明文到密文的不可逆的映射，只有加密过程，没有解密过程。同时，哈希函数可以将任意长度的输入经过变化以后得到固定长度的输出。哈希函数的这种单向特征和输出数据长度固定的特征使得它可以生成消息或者数据。



常用hash算法的介绍：

(1) MD4

MD4（RFC 1320）是 MIT 的Ronald L. Rivest在 1990 年设计的，MD 是 Message Digest（消息摘要）的缩写。它适用在32位字长的处理器上用高速软件实现——它是基于 32位操作数的位操作来实现的。

(2) MD5

MD5（RFC 1321）是 Rivest 于1991年对MD4的改进版本。它对输入仍以512位分组，其输出是4个32位字的级联，与 MD4 相同。MD5比MD4来得复杂，并且速度较之要慢一点，但更安全，在抗分析和抗差分方面表现更好。

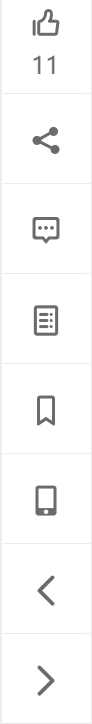
(3) SHA-1及其他

SHA1是由NIST NSA设计为同DSA一起使用的，它对长度小于264的输入，产生长度为160bit的散列值，因此抗穷举（brute-force）性更好。SHA-1 设计时基于和MD4相同原理，并且模仿了该算法。

常见hash算法的原理

散列表，它是基于快速存取的角度设计的，也是一种典型的“空间换时间”的做法。顾名思义，该数据结构可以理解为一个线性表，但是其中的元素不是紧密排列的，而是可能存在空隙。

散列表（Hash table，也叫哈希表），是根据关键码值（Key value）而直接进行访问的数据结构。也就是说，它通过把关键码值映射到表中一个位置来访问记录，以加快查找的速度。这个映射函数叫做散列函数，存放记录的数组叫做散列表。



处理数据时不进行归一化会有什么影响？归一化的作用是什么？ 什么时候需要归一化？
阅读数 5225

B树，B-树，B*树，B+和红黑树的区别
阅读数 4720

最新评论

缓冲区溢出攻击
weixin_43797605：难难难难难难难难难难难难难得一见得好好好好好好好好好好好好好文...

【算法】递归与while循环的区别
weixin_43373818：真是个小机灵鬼

【经典】吴恩达《机器学习》课程
ZZUYangShao：[reply]ZZUYangShao[/reply]为什么说网易公开课上有这门课？两个一样吗？...

【经典】吴恩达《机器学习》课程
ZZUYangShao：推荐的第二个视频，是以前的cs229的课程吗？为什么和以前看的不太一样啊？以...

B树，B-树，B*树，B+和红黑树...
OpenGao：b+tree有可能不平衡吗？



程序人生



CSDN资讯

👤 QQ客服 ✉ kefu@csdn.net
🗣 客服论坛 ☎ 400-660-0108
 工作时间 8:30-22:00

关于我们 | 招聘 | 广告服务 | 网站地图
🐾 百度提供站内搜索 京ICP备19004658号
© 1999-2019 北京创新乐知网络技术有限公司

网络110报警服务 经营性网站备案信息
北京互联网违法和不良信息举报中心
中国互联网举报中心 家长监护 版权申诉

比如我们存储70个元素，但我们可能为这70个元素申请了100个元素的空间。70/100=0.7，这个数字称为负载因子。我们之所以这样做，也是为了“快速存取”的目的。我们基于一种结果尽可能随机平均分布的固定函数H为每个元素安排存储位置，这样就可以避免遍历性质的线性搜索，以达到快速存取。但是由于此随机性，也必然导致一个问题就是冲突。所谓冲突，即两个元素通过散列函数H得到的地址相同，那么这两个元素称为“同义词”。这类似于70个人去一个有100个椅子的饭店吃饭。散列函数的计算结果是一个存储单位地址，每个存储单位称为“桶”。设一个散列表有m个桶，则散列函数的值域应为 [0，m-1] 。

解决冲突是一个复杂问题。

冲突主要取决于：

- (1) 散列函数，一个好的散列函数的值应尽可能平均分布。
- (2) 处理冲突方法。
- (3) 负载因子的大小。太大不一定就好，而且浪费空间严重，负载因子和散列函数是联动的。

解决冲突的办法：

- (1) 线性探查法：冲突后，线性向前试探，找到最近的一个空位置。缺点是会出现堆积现象。存取时，可能不是同义词的词也位于探查序列，影响效率。
- (2) 双散列函数法：在位置d冲突后，再次使用另一个散列函数产生一个与散列表桶容量m互质的数c，依次试探 (d+n*c) %m，使探查序列跳跃式分布。

常用的构造散列函数的方法

散列函数能使对一个数据序列的访问过程更加迅速有效，通过散列函数，数据元素将被更快地定位：

- 1. 直接寻址法：取关键字或关键字的某个线性函数值为散列地址。即H (key) =key或H (key) = a? key + b，其中a和b为常数 (这种散列函数叫做自身函数)
- 2. 数字分析法：分析一组数据，比如一组员工的出生年月日，这时我们发现出生年月日的前几位数字大体相同，这样的话，出现冲突的几率就会很大，但是我们发现年月日的后几位表示月份和具体日期的数字差别很大，如果用后面的数字来构成散列地址，则冲突的几率会明显降低。因此数字分析法就是找出数字的规律，尽可能利用这些数据来构造冲突几率较低的散列地址。
- 3. 平方取中法：取关键字平方后的中间几位作为散列地址。
- 4. 折叠法：将关键字分割成位数相同的几部分，最后一部分位数可以不同，然后取这几部分的叠加和（去除进位）作为散列地址。
- 5. 随机数法：选择一随机函数，取关键字的随机值作为散列地址，通常用于关键字长度不同的场合。
- 6. 除留余数法：取关键字被某个不大于散列表表长m的数p除后所得的余数为散列地址。即 H (key) = key MOD p，p ≦m。不仅可以对关键字直接取模，也可在折叠、平方取中等运算之后取模。对p的选择很重要，一般取素数或m，若p选的不好，容易产生同义词。

查找的性能分析

散列表的查找过程基本上和造表过程相同。一些关键码可通过散列函数转换的地址直接找到，另一些关键码在散列函数得到的地址上产生了冲突，需要按处理冲突的方法进行查找。在介绍的三种处理冲突的方法中，产生冲突后的查找仍然是给定值与关键码进行比较的过程。所以，对散列表查找效率的度量，依然用平均查找长度来衡量。

查找过程中，关键码的比较次数，取决于产生冲突的多少，产生的冲突少，查找效率就高，产生的冲突多，查找效率就低。因此，影响产生冲突多少的因素，也就是影响查找效率的因素。影响产生冲突多少有以下三个因素：

- 1. 散列函数是否均匀；
- 2. 处理冲突的方法；
- 3. 散列表的装填因子。

散列表的装填因子定义为：α= 填入表中的元素个数 / 散列表的长度

α是散列表装满程度的标志因子。由于表长是定值，α与“填入表中的元素个数”成正比，所以，α越大，填入表中的元素较多，产生冲突的可能性就越大；α越小，填入表中的元素较少，产生冲突的可能性就越小。

实际上，散列表的平均查找长度是装填因子α的函数，只是不同处理冲突的方法有不同的函数。

👍
11

👤

💬

📖

🔖

📱

<

>

👑
vip

🧩

🔄

🛡

了解了hash基本定义，就不能不提到一些著名的hash算法，MD5 和 SHA-1 可以说是目前应用最广泛的Hash算法，而它们都是以MD4 为基础设计的。那么他们都是什么意思呢？

这里简单说一下：

(1) MD4

MD4 (RFC 1320) 是 MIT 的 Ronald L. Rivest 在 1990 年设计的，MD 是 Message Digest 的缩写。它适用在32位字长的处理器上用高速软件实现--它是基于 32 位操作数的位操作来实现的。

(2) MD5

MD5 (RFC 1321) 是 Rivest 于1991年对MD4的改进版本。它对输入仍以512位分组，其输出是4个32位字的级联，与 MD4 相同。MD5比MD4来得复杂，并且速度较之要慢一点，但更安全，在抗分析和抗差分方面表现更好

(3) SHA-1 及其他

SHA1是由NIST NSA设计为同DSA一起使用的，它对长度小于264的输入，产生长度为160bit的散列值，因此抗穷举（brute-force）性更好。SHA-1 设计时基于和MD4相同原理，并且模仿了该算法。

哈希表不可避免冲突（collision）现象：对不同的关键字可能得到同一哈希地址 即key1≠key2，而hash（key1）=hash（key2）。因此，在建造哈希表时不仅要设定一个好的哈希函数，而且要设定一种处理冲突的方法。可如下描述哈希表：根据设定的哈希函数 H（key）和所选中的处理冲突的方法，将一组关键字映象到一个有限的、地址连续的地址集（区间）上并以关键字在地址集中的“象”作为相应记录在表中的存储位置，这种表被称为哈希表。

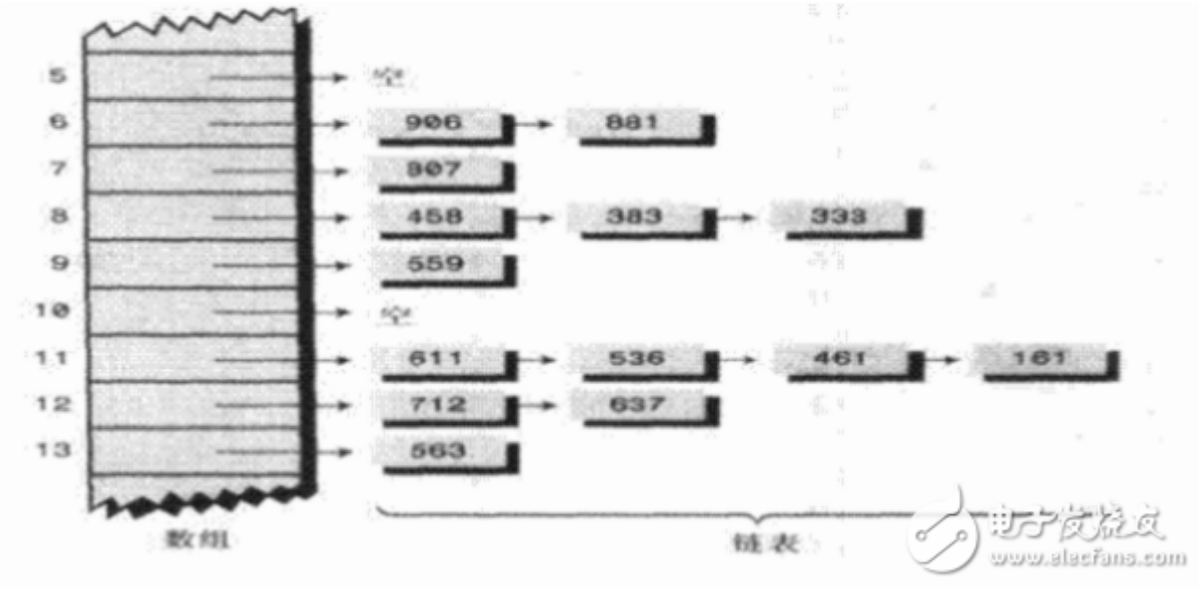
对于动态查找表而言，1）表长不确定；2）在设计查找表时，只知道关键字所属范围，而不知道确切的关键字。因此，一般情况需建立一个函数关系，以f（key）作为关键字为key的录在表中的位置，通常称这个函数f（key）为哈希函数。（注意：这个函数并不一定是数学函数）

哈希函数是一个映象，即：将关键字的集合映射到某个地址集合上，它的设置很灵活，只要这个地址集合的大小不超出允许范围即可。

现实中哈希函数是需要构造的，并且构造的好才能使用的好。

那么这些Hash算法到底有什么用呢？

Hash算法在信息安全方面的应用主要体现在以下的3个方面：



(1) 文件校验

我们比较熟悉的校验算法有奇偶校验和CRC校验，这2种校验并没有抗数据篡改的能力，它们一定程度上能检测并纠正数据传输中的信道误码，但却不能防止对数据的恶意破坏。

MD5 Hash算法的“数字指纹”特性，使它成为目前应用最广泛的一种文件完整性校验和（Checksum）算法，不少Unix系统有提供计算md5 checksum的命令。

(2) 数字签名

Hash 算法也是现代密码体系中的一个重要组成部分。由于非对称算法的运算速度较慢，所以在数字签名协议中，单向散列函数扮演了一个重要的角色。对 Hash 值，又称“数字摘要”进行数字签名，在统计上可以认为与对文件本身进行数字签名是等效的。而且这样的协议还有其他的优点。

👍
11

👤

💬

📖

🔖

📱

<

>

👑
vip

📺

🗣️

🛡️

(3) 鉴权协议

如下的鉴权协议又被称作挑战--认证模式：在传输信道是可被侦听，但不可被篡改的情况下，这是一种简单而安全的方法。

文件hash值

MD5-Hash-文件的数字文摘通过Hash函数计算得到。不管文件长度如何，它的Hash函数计算结果是一个固定长度的数字。与加密算法不同，这一个Hash算法是一个不可逆的单向函数。采用安全性高的Hash算法，如MD5、SHA时，两个不同的文件几乎不可能得到相同的Hash结果。因此，一旦文件被修改，就可检测出来。

Hash函数还有另外的含义。实际中的Hash函数是指把一个大范围映射到一个小范围。把大范围映射到一个小范围的目的往往是为了节省空间，使得数据容易保存。除此以外，Hash函数往往应用于查找上。所以，在考虑使用Hash函数之前，需要明白它的几个限制：

- 1. Hash的主要原理就是把大范围映射到小范围；所以，你输入的实际值的个数必须和小范围相当或者比它更小。不然冲突就会很多。
- 2. 由于Hash逼近单向函数；所以，你可以用它来对数据进行加密。
- 3. 不同的应用对Hash函数有着不同的要求；比如，用于加密的Hash函数主要考虑它和单项函数的差距，而用于查找的Hash函数主要考虑它映射到小范围的冲突率。

应用于加密的Hash函数已经探讨过太多了，在作者的博客里面有更详细的介绍。所以，本文只探讨用于查找的Hash函数。

Hash函数应用的主要对象是数组（比如，字符串），而其目标一般是一个int类型。以下我们都按照这种方式来说明。

一般的说，Hash函数可以简单的划分为如下几类：

- 1. 加法Hash；
- 2. 位运算Hash；
- 3. 乘法Hash；
- 4. 除法Hash；
- 5. 查表Hash；
- 6. 混合Hash；

下面详细的介绍以上各种方式在实际中的运用。

一 加法Hash

所谓的加法Hash就是把输入元素一个一个的加起来构成最后的结果。标准的加法Hash的构造如下：

```
static int additiveHash (String key, int prime)
{
    int hash, i;

    for (hash = key.length () , i = 0; i <= key.length () ; i++)

        hash += key.charAt (i) ;

    return (hash % prime) ;
}
```

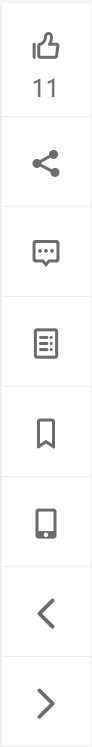
这里的prime是任意的质数，看得出，结果的值域为 [0, prime-1] 。

二 位运算Hash

这类型Hash函数通过利用各种位运算（常见的是移位和异或）来充分的混合输入元素。比如，标准的旋转Hash的构造如下：

```
static int rotatingHash (String key, int prime)
{

```



```
int hash, i;

for (hash=key.length () , i=0; i<key.length (); i++)
    hash = (hash « «4» » 28) ^key.charAt (i) ;

return (hash % prime) ;

}
```

先移位，然后再进行各种位运算是这种类型Hash函数的主要特点。比如，以上的那段计算hash的代码还可以有如下几种变形：

```
hash = (hash << 5) >> 27) ^key.charAt (i) ;
```

```
hash += key.charAt (i) ;
```

```
hash += (hash << 10) ;
```

```
hash ^= (hash >> 6) ;
```

```
if ( (i&1) == 0)
```

```
hash ^= (hash << 7) >> 3) ;
```

else

```
hash ^= ~ ( (hash << 11) >> 5) ;
```

```
hash += (hash << 5)
```

```
hash = key.charAt (i) + (hash « «6» » 16) ? hash;
```

```
hash ^= ( (hash << 5) >> 2) ;
```

三 乘法Hash

这种类型的Hash函数利用了乘法的不相关性（乘法的这种性质，最有名的莫过于平方取头尾的随机数生成算法，虽然这种算法效果不好）。比如，

```
static int bernstein (String key)
```

```
int hash = 0;
```

```
int i;
```

```
for (i=0; i
```

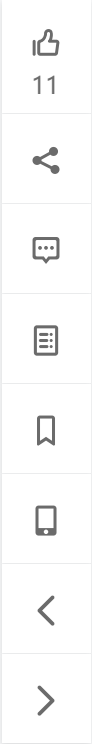
```
return hash;
```

jdk5.0里面的String类的hashCode () 方法也使用乘法Hash。不过，它使用的乘数是31。推荐的乘数还有：131， 1313， 31， 131313等等。

使用这种方式的著名Hash函数还有：

```
// 32位FNV算法
```

```
int M_SHIFT = 0;
```



```
public int FNVHash (byte [] data)

{

int hash = (int) 2166136261L;

for (byte b : data)

hash = (hash * 16777619) ^ b;

if (M_SHIFT == 0)

return hash;

return (hash ^ (hash 》》 M_SHIFT) ) & M_MASK;

}
```

以及改进的FNV算法：

```
public static int FNVHash1 (String data)

{

final int p = 16777619;

int hash = (int) 2166136261L;

for (int i=0;i

hash = (hash ^ data.charAt (i) ) * p;

hash += hash 《《 13;

hash ^= hash 》》 7;

hash += hash 《《 3;

hash ^= hash 》》 17;

hash += hash 《《 5;

return hash;

}
```

除了乘以一个固定的数，常见的还有乘以一个不断改变的数，比如：

```
static int RSHash (String str)

{

int b = 378551;

int a = 63689;

int hash = 0;

for (int i = 0; i 《 str.length () ; i++)

{


hash = hash * a + str.charAt (i) ;


a = a * b;

}


return (hash & 0x7FFFFFFF) ;


}
```


11
























虽然Adler32算法的应用没有CRC32广泛，不过，它可能是乘法Hash里面最有名的一个了。关于它的介绍，大家可以去看RFC 1950 规范。

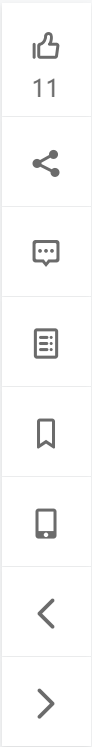
四 除法Hash

除法和乘法一样，同样具有表面上看起来的不相关性。不过，因为除法太慢，这种方式几乎找不到真正的应用。需要注意的是，我们在前面看到的hash的 结果除以一个prime的目的只是为了保证结果的范围。如果你不需要它限制一个范围的话，可以使用如下的代码替代“ hash%prime”：
hash = hash ^ (hash》》10) ^ (hash》》20) 。

五 查表Hash

查表Hash最有名的例子莫过于CRC系列算法。虽然CRC系列算法本身并不是查表，但是，查表是它的一种最快的实现方式。下面是CRC32的实现：

```
static int crctab [256] = {  
  
    0x00000000, 0x77073096, 0xee0e612c, 0x990951ba, 0x076dc419, 0x706af48f, 0xe963a535, 0x9e6495a3,  
    0x0edb8832,  
  
    0x79dcb8a4, 0xe0d5e91e, 0x97d2d988, 0x09b64c2b, 0x7eb17cbd, 0xe7b82d07, 0x90bf1d91, 0x1db71064,  
    0x6ab020f2,  
  
    0xf3b97148, 0x84be41de, 0x1adad47d, 0x6ddde4eb, 0xf4d4b551, 0x83d385c7, 0x136c9856, 0x646ba8c0,  
    0xfd62f97a,  
  
    0x8a65c9ec, 0x14015c4f, 0x63066cd9, 0xfa0f3d63, 0x8d080df5, 0x3b6e20c8, 0x4c69105e, 0xd56041e4,  
    0xa2677172,  
  
    0x3c03e4d1, 0x4b04d447, 0xd20d85fd, 0xa50ab56b, 0x35b5a8fa, 0x42b2986c, 0xdbbbc9d6, 0xacbcf940,  
    0x32d86ce3,  
  
    0x45df5c75, 0xdcd60dcf, 0xabd13d59, 0x26d930ac, 0x51de003a, 0xc8d75180, 0xbf061116, 0x21b4f4b5,  
    0x56b3c423,  
  
    0xcfba9599, 0xb8bda50f, 0x2802b89e, 0x5f058808, 0xc60cd9b2, 0xb10be924, 0x2f6f7c87, 0x58684c11,  
    0xc1611dab,  
  
    0xb6662d3d, 0x76dc4190, 0x01db7106, 0x98d220bc, 0xefd5102a, 0x71b18589, 0x06b6b51f, 0x9fbfe4a5,  
    0xe8b8d433,  
  
    0x7807c9a2, 0x0f00f934, 0x9609a88e, 0xe10e9818, 0x7f6a0dbb, 0x086d3d2d, 0x91646c97, 0xe6635c01,  
    0x6b6b51f4,  
  
    0x1c6c6162, 0x856530d8, 0xf262004e, 0x6c0695ed, 0x1b01a57b, 0x8208f4c1, 0xf50fc457, 0x65b0d9c6,  
    0x12b7e950,  
  
    0x8bbeb8ea, 0xfcb9887c, 0x62dd1ddf, 0x15da2d49, 0x8cd37cf3, 0xfbd44c65, 0x4db26158, 0x3ab551ce,  
    0xa3bc0074,  
  
    0xd4bb30e2, 0x4adfa541, 0x3dd895d7, 0xa4d1c46d, 0xd3d6f4fb, 0x4369e96a, 0x346ed9fc, 0xad678846,  
    0xda60b8d0,  
  
    0x44042d73, 0x33031de5, 0xaa0a4c5f, 0xdd0d7cc9, 0x5005713c, 0x270241aa, 0xbe0b1010, 0xc90c2086,  
    0x5768b525,  
  
    0x206f85b3, 0xb966d409, 0xce61e49f, 0x5edef90e, 0x29d9c998, 0xb0d09822, 0xc7d7a8b4, 0x59b33d17,  
    0x2eb40d81,  
  
    0xb7bd5c3b, 0xc0ba6cad, 0xedb88320, 0x9abfb3b6, 0x03b6e20c, 0x74b1d29a, 0xead54739, 0x9dd277af,  
    0x04db2615,  
  
    0x73dc1683, 0xe3630b12, 0x94643b84, 0x0d6d6a3e, 0x7a6a5aa8, 0xe40ecf0b, 0x9309ff9d, 0x0a00ae27,  
    0x7d079eb1,  
  
    0xf00f9344, 0x8708a3d2, 0x1e01f268, 0x6906c2fe, 0xf762575d, 0x806567cb,
```



0x196c3671, 0x6e6b06e7, 0xfed41b76, 0x89d32be0, 0x10da7a5a, 0x67dd4acc, 0xf9b9df6f, 0x8ebeeff9, 0x17b7be43,

0x60b08ed5, 0xd6d6a3e8, 0xa1d1937e, 0x38d8c2c4, 0x4fdff252, 0xd1bb67f1, 0xa6bc5767, 0x3fb506dd, 0x48b2364b,

0xd80d2bda, 0xaf0a1b4c, 0x36034af6, 0x41047a60, 0xdf60efc3, 0xa867df55, 0x316e8eef, 0x4669be79, 0xcb61b38c,

0xbc66831a, 0x256fd2a0, 0x5268e236, 0xcc0c7795, 0xbb0b4703, 0x220216b9, 0x5505262f, 0xc5ba3bbe, 0xb2bd0b28,

0x2bb45a92, 0x5cb36a04, 0xc2d7ffa7, 0xb5d0cf31, 0x2cd99e8b, 0x5bdeae1d, 0x9b64c2b0, 0xec63f226, 0x756aa39c,

0x026d930a, 0x9c0906a9, 0xeb0e363f, 0x72076785, 0x05005713, 0x95bf4a82, 0xe2b87a14, 0x7bb12bae, 0x0cb61b38,

0x92d28e9b, 0xe5d5be0d, 0x7cdcefb7, 0x0bdbbdf21, 0x86d3d2d4, 0xf1d4e242, 0x68ddb3f8, 0x1fda836e, 0x81be16cd,

0xf6b9265b, 0x6fb077e1, 0x18b74777, 0x88085ae6, 0xff0f6a70, 0x66063bca, 0x11010b5c, 0x8f659eff, 0xf862ae69,

0x616bffd3, 0x166ccf45, 0xa00ae278, 0xd70dd2ee, 0x4e048354, 0x3903b3c2, 0xa7672661, 0xd06016f7, 0x4969474d,

0x3e6e77db, 0xaed16a4a, 0xd9d65adc, 0x40df0b66, 0x37d83bf0, 0xa9bcae53, 0xdebb9ec5, 0x47b2cf7f, 0x30b5ffe9,

0xbdbdf21c, 0xcabac28a, 0x53b39330, 0x24b4a3a6, 0xbad03605, 0xcd70693, 0x54de5729, 0x23d967bf, 0xb3667a2e,

0xc4614ab8, 0x5d681b02, 0x2a6f2b94, 0xb40bbe37, 0xc30c8ea1, 0x5a05df1b, 0x2d02ef8d

```
};

int crc32 (String key, int hash)

{

    int i;

    for (hash=key.length () , i=0; i

        hash = (hash 》 》 8) ^ crctab [ (hash & 0xff) ^ k.charAt (i) ] ;

    return hash;

}
```

查表Hash中有名的例子有：Universal Hashing和Zobrist Hashing。他们的表格都是随机生成的。

六 混合Hash


混合Hash算法利用了以上各种方式。各种常见的Hash算法，比如MD5、Tiger都属于这个范围。它们一般很少在面向查找的Hash函数里面使用。


hash算法原理详解


阅读数 11万 +


一.概念哈希表就是一种以键-值(key-indexed)存储数据的结构，我们只要输入待查找的值即key，即可查找到其对应... 博文 | 来自： 至道


 想对作者说点什么


11

























算法：常见hash算法的原理

阅读数 194

算法：常见hash算法的原理 散列表（Hashtable，也叫哈希表），它是基于高速存取的角度设计的，也是一种典型的...

博文 | 来自： 王晓(Java)

一致性哈希算法的原理与实现

阅读数 6276

分布式系统中对象与节点的映射关系，传统方案是使用对象的哈希值，对节点个数取模，再映射到相应编号的节点，...

博文 | 来自： kefeng.wang 的博客

HASH哈希算法的简单理解

阅读数 292

基本概念：哈希算法就是将不同长度的输入值，计算成为小于输入长度的固定值，当两个不同的输入值x!=y，可能会...

博文 | 来自： zwf888的专栏

数据结构 Hash表（哈希表）

阅读数 2万+

参考链接：数据结构（严蔚敏）什么是Hash表要想知道什么是哈希表，那得先了解哈希函数哈希函数对比之前博客讨...

博文 | 来自： 积跬步 至千里

Hash详解

阅读数 7471

Hash(哈希)Hash：散列，通过关于键值(key)的函数，将数据映射到内存存储中一个位置来访问。这个过程叫做Hash...

博文 | 来自： yt618121的专栏

通俗易懂的哈希算法讲解

阅读数 1万+

哈希是一种加密算法哈希函数（HashFunction），也称为散列函数或杂凑函数。哈希函数是一个公开函数，可以将...

博文 | 来自： zongyue_wang的...

系统学习hash算法（哈希算法）

阅读数 2万+

本篇文章仅仅是关于hash一个开始。在学习了

博文 | 来自： 一座青山的专栏

几种常用hash算法及原理

阅读数 5

计算理论中，没有Hash函数的说法，只有单向函数的说法。所谓的单向函数，是一个复杂的定义，大家可以去看计算...

博文 | 来自： weixin_30361753...

常见的hash算法及其原理 - Beyond_2016的博客 - CSDN博客

9-6

几种常用hash算法及原理 - weixin_33686714的博客 - CSDN博客

1-26

哈希表（散列表） 原理详解

阅读数 102

什么是哈希表？ 哈希表（Hashtable，也叫散列表），是根据关键码值(Keyvalue)而直接进行访问的数据结构。也...

博文 | 来自： u011514729的专栏

常见hash算法的原理 - weixin_34388207的博客 - CSDN博客

1-24

【整理】hash算法原理及常见函数 - 土著部落 - CSDN博客

12-15

五分钟带你了解哈希算法究竟是什么！

阅读数 3万+

大家好呀，我是你们的贝尔同学。经过一段时间的认知学习，大家应该对数字货币有了一定的了解。今天呢，我们要...

博文 | 来自： xinshengdaxue00...



汤高

197篇文章
排名:4000+

关注



王晓(Java)

294篇文章
排名:8000+

关注



kefeng-wang

55篇文章
排名:千里之外

关注



乘简

28篇文章
排名:千里之外

关注

HashMap底层实现原理 - Beyond_2016的博客 - CSDN博客

11-3

【hash】理解哈希算法和常见应用 - scxyz的博客 - CSDN博客

3-19

Hashmap and Java.io PrintWriter

阅读数 182

这几天学习了HashMap的底层实现,但是发现好几个版本的，代码不一，而且看了Android包的HashMap和JDK中的...

博文 | 来自： qq_36864672的博客

解决hash冲突的四种办法

阅读数 720

目录开放定址法 线性探测再散列 二次探测再散列 伪随机探测再散列 再哈希法 链地址法 建立公共溢出区 优缺点 开放...

博文 | 来自： Beyond_2016的博客

hash算法原理详解 - chenshuangma的博客 - CSDN博客

11-20

11

<

>

hash算法原理详解 - Chen Rong的博客 - CSDN博客

11-20

Hash算法总结

阅读数 5万+

Hash是什么，它的作用先举个例子。我们每个活在世上的人，为了能够参与各种社会活动，都需要一个用于识别自己... 博文 | 来自: asdzheng的专栏

...两个数组之交集算法与hash - Beyond_2016的博客 - CSDN博客...

6-28

各种经典的hash算法 - kingj126的专栏 - CSDN博客

11-21

Hash算法详解

阅读数 1万+

Hash算法，简称散列算法，也成哈希算法（英译）。是将一个大文件映射成一个小串字符。... 博文 | 来自: 时光钟摆

嵌入式驱动那年的笔试面试-有干货

阅读数 1万+

那年的笔试面试题，面试经验总结和干货发放

博文 | 来自: Mingrenjiuwei的...

最通俗易懂的一致性哈希算法原理

阅读数 512

再讨论一致性哈希之前，我们先来回顾一下缓存的演化历史：当我们的系统还是一个非常小的时候，对于用户的请求... 博文 | 来自: 不爱学习

哈希算法

阅读数 664

对于哈希算法，在我们平时的开发中，都是基本上拿来就用就行了，所以这节我们将重点放在如何使用，并不进行哈... 博文 | 来自: Tattoo的博客

哈希表&一致性哈希&超有爱的并查集

阅读数 442

文章目录1.哈希函数，哈希表与布隆过滤器2.设计RandomPool结构3.一致性哈希4.并查集结构5.并查集结构应用【岛... 博文 | 来自: 李滚滚的博客

最快速度求两个数组之交集算法与hash

阅读数 110

一个题目该题目来自58同城的二面，用最快速度求两个数组之交集算法。比如A={6，2，4，1},B={2，9，4，3}，那... 博文 | 来自: Beyond_2016的博客

【算法】理解哈希算法 hash 和常见应用

阅读数 201

概念将任意长度的二进制值串映射为固定长度的二进制值串，这个映射的规则就是哈希算法。通过原始数据映射之后... 博文 | 来自: scxyz的博客

常用字符串hash算法及评测

阅读数 289

RShash算法unsignedintRShash(char*str,unsignedintlen){ unsignedintb =378551; unsignedinta =63689;... 博文 | 来自: netqvq|求知

perl 哈希(hash)学习笔记（一）

阅读数 6347

转载请注明出处：http://www.cnblogs.com/tobecrazy/1.什么是哈希 哈希是perl的一种数据类型，比较类似数组， ... 博文 | 来自: u012467492的专栏

史上最全设计模式导学目录（完整版）

阅读数 28万+

圣诞献礼！ 2012年-2013年，Sunny在CSDN技术博客中陆续发表了100多篇与设计模式相关的文章，涵盖了七... 博文 | 来自: 刘伟技术博客

哈希算法详解

阅读数 734

一维字符串哈希功能：在O(1)时间内查询某个区间的子串是什么(该串的哈希值)等等实现方法：类似于前缀合，对字... 博文 | 来自: l_believe_CWJ的博...

Object类的hashCode()方法

阅读数 905

publicclassday11{publicstaticvoidmain(String[]args){Objectobj1=newObject();inthashCode=obj1.hashCode... 博文 | 来自: beyond谚语的博客

常见hash算法的原理

阅读数 3

散列表,它是基于快速存取的角度设计的，也是一种典型的“空间换时间”的做法。顾名思义，该数据结构可以理解为... 博文 | 来自: weixin_30700977...

什么是哈希算法？

阅读数 952

哈希算法的基本含义 哈希是密码学的基础，理解哈希是理解数字签名和加密通信等技术的必要前提。 哈希，英文... 博文 | 来自: 每天做一点

hash算法原理

阅读数 1712

一.概念哈希表就是一种以键-值(key-indexed)存储数据的结构，我们只要输入待查找的值即key，即可查找到其对应... 博文 | 来自: liyanan21的博客

11

VIP

Murmurhash介绍与实现

阅读数 2万+

MurmurHash 是一种非加密型哈希函数，适用于一般的哈希检索操作。[1][2][3] 由AustinAppleby在2008年发明，[... 博文 | 来自： ThinkMo的专栏

区块链100讲：据说，80%的人都搞不懂哈希算法

阅读数 109

前面的《区块链100讲》介绍了区块链、算力、挖矿等，几乎每一讲都会提到一个词哈希(Hashing)。聊到区块链的时... 博文 | 来自： weixin_33850015...

一致性hash算法 - consistent hashing

阅读数 17万+

一致性hash算法（consistenthashing）张亮consistenthashing算法早在1997年就在论文Consistenthashingandr... 博文 | 来自： sparkliang的专栏

【整理】hash算法原理及常见函数

阅读数 6

简介Hash，一般翻译做“散列”，也有直接音译为“哈希”的，就是把任意长度的输入，通过散列算法，变换成固定... 博文 | 来自： weixin_30740581...

机器学习常见的问题

阅读数 111

过拟合原因数据：数据不规范，数据量少，数据穿越，统计特征用到了未来的信息或者标签信息算法：算法过于复杂... 博文 | 来自： Beyond_2016的博客

【数据结构与算法】之哈希算法 --- 第十一篇

阅读数 517

博主秋招提前批已拿百度、字节跳动、拼多多、顺丰等公司的offer，可加微信：pcwl_Java一起交流秋招面试经验，... 博文 | 来自： 微信公众号：码农...

浅显理解 hashcode 和 hash 算法

阅读数 3万+

摘要二进制计算的一些基础知识为什么使用hashCodeString类型的hashCode方法为什么大部分hashCode方法使用3... 博文 | 来自： ignore

JAVA中hashCode的编写

阅读数 162

1、把某个非零常数值，比如17，保存在一个叫result的int类型的变量中。2、对于对象中的关键域f(指equals方法... 博文 | 来自： beyondqinghua的...

一致性hash算法释义

阅读数 29

一致性Hash算法背景 一致性哈希算法在1997年由麻省理工学院的Karger等人在解决分布式Cache中提出的，设... 博文 | 来自： weixin_33964094...

哈希表算法原理

阅读数 1944

原文地址：https://blog.mimvp.com/article/5724.html今天看到这篇文章重新对哈希表和字典有了新的认识，希望... 博文 | 来自： u012371712的博客

hash算法 (hashmap 实现原理)

阅读数 254

Hash，一般翻译做“散列”，也有直接音译为“哈希”的，就是把任意长度的输入（又叫做预映射，pre-image... 博文 | 来自： zha_zi的专栏

HashMap的实现原理

阅读数 767

Hash，一般翻译做“散列”，也有直接音译为“哈希”的，就是把任意长度的输入（又叫做预映射，pre-image），... 博文 | 来自： 春夏秋冬过

看完此文，必须明白一致性Hash算法

阅读数 7377

一致性Hash算法在1997年由麻省理工学院提出的一种分布式哈希（DHT）实现算法，设计目标是为了解决因特网... 博文 | 来自： cb_lcl的博客

哈希表冲突及处理冲突的方法

阅读数 2702

一、哈希函数和哈希冲突的基本概念1.哈希函数：哈希法又称散列法、杂凑法以及关键字地址计算法等，相应的表成... 博文 | 来自： Hello_GY 的博客

哈希Hash 算法

阅读数 1190

注意文章详解（http://blog.csdn.net/v_july_v/article/details/6256463）哈希由来：数组的特点是：寻址容易，插... 博文 | 来自： 《我是你的正能量》

大数据量中的模糊查询优化方案

阅读数 46

-----[版权声明： ... 博文 | 来自： weixin_34138521...

哈希（Hash）算法


阅读数 3163


一、什么是Hash算法散列算法（HashAlgorithm），又称哈希算法，杂凑算法，是一种从任意文件中创造小的数字... 博文 | 来自： 码墨


到底什么是哈希Hash？


阅读数 9406


但凡是从事过计算机行业的人，多多少少都会听说过这个概念，但是又对其很模糊，那么到底什么是Hash呢？定义H... 博文 | 来自： Toby的博客


11






















Java基础-Hash算法

阅读数 409

文章转载自：全网把Map中的hash()分析的最透彻的文章，别无二家。你知道HashMap中hash方法的具体实现吗? ... 博文 | 来自： 数数1234

常用Hash算法(C语言实现)

阅读数 970

转载自：http://blog.csdn.net/huangkangying/article/details/8748537以下代码来自：http://www.partow.net/... 博文 | 来自： 收集学习过程中对...


最常用的三种哈希算法

阅读数 251

散列算法（HashAlgorithm），又称哈希算法，Hash算法能将将任意长度的二进制明文映射为较短的二进制串的算... 博文 | 来自： 上道的程序员

c# linq原理 c# 装箱有什么用 c#集合 复制 c# 一个字符串分组 c++和c#哪个就业率高 c# 批量动态创建控件 c# 模块和程序集的区别 c# gmap 截图 c# 验证码图片生成类 c# 再次尝试 连接失败

没有更多推荐了，[返回首页](#)


11

