# 字符串hash，康托展开总结

**各种字符串hash函数：**

**ELF Hash BKDRHash APHash DJBHash JSHash RSHash SDBMHash PJWHash ELFHash**

**字符串hash之BKDRhash函数**

**有研究表明BKDRhash函数冲突率较低，且变成复杂度低，所以可以用来作为常用hash函数**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Hash函数** | **数据1** | **数据2** | **数据3** | **数据4** | **数据1得分** | **数据2得分** | **数据3得分** | **数据4得分** | **平均分** |
| **BKDRHash** | **2** | **0** | **4774** | **481** | **96.55** | **100** | **90.95** | **82.05** | **92.64** |
| **APHash** | **2** | **3** | **4754** | **493** | **96.55** | **88.46** | **100** | **51.28** | **86.28** |
| **DJBHash** | **2** | **2** | **4975** | **474** | **96.55** | **92.31** | **0** | **100** | **83.43** |
| **JSHash** | **1** | **4** | **4761** | **506** | **100** | **84.62** | **96.83** | **17.95** | **81.94** |
| **RSHash** | **1** | **0** | **4861** | **505** | **100** | **100** | **51.58** | **20.51** | **75.96** |
| **SDBMHash** | **3** | **2** | **4849** | **504** | **93.1** | **92.31** | **57.01** | **23.08** | **72.41** |
| **PJWHash** | **30** | **26** | **4878** | **513** | **0** | **0** | **43.89** | **0** | **21.95** |
| **ELFHash** | **30** | **26** | **4878** | **513** | **0** | **0** | **43.89** | **0** | **21.95** |

**unsigned int bkdr\_hash(const char\* str)**

**{**

**unsigned int seed = 31; // 31 131 1313 13131 131313 etc.. 37（最好是质数）**

**unsigned int hash = 0;**

**while (\*str) hash = (hash \* seed + (\*str++))%P;//P是一个较大质数**

**return hash;**

**}**

**各种证明：**

[**http://www.it165.net/pro/html/201410/24949.html**](http://www.it165.net/pro/html/201410/24949.html)

**双hash优化：**

**在hash表中想要用线性探查的方式处理hash表的冲突，那么每次比对要查找的元素与当前元素是否相等就显得十分麻烦，特别是两个数据是字符串的时候，strcmp超级耗时，此时我们就可以在存储新元素时通过另一hash算法算出两者的hash值，将这两个hash存入结构体，并且排序，用二分查找第一个hash值，并比较第二个hash值，若符合则相同**

**康托展开**

**叙述：有一个数字序列，所有数都是[1,n]的，且任意两者互异，此时我们用一一对应的方式存储这些数据就有n！种可能，康托算法对处理这种序列提供了完美的解决方法。称为康托展开。我们把这个数列看作全排列，数字所对hash值就为全排列的大小（第几大）。**

**公式：hash(key)=a[n]\*(n-1)!+a[n-1]\*(n-2)!+…+a[2]\*(2-1)!+a[1]\*(1-1)!**

**其中，a[n]所存的值为在第n个数（从小到大的顺序）之前比n小的数的个数**

**Hash值完全可以通过循环在n次内算出，且数列与hash完全一一对应，可谓完美算法。**

**康托逆展开：**

**由于康托展开的一一对应性，我们同样可以算出原来的全排列。**

**n为康托展开式，k为总阶乘**

**第一位：(n-1)/(k-1) 余n’**

**第二位：n’/(k-2) 余n’’**

**以此类推。。。**

**Poj 1200**

**描述**

许多人喜欢解决难题。一个这样的难题是在一个给定的文本中找到一个隐藏的素数。这个数字是所给字符串中不同子串的个数。当你很快会发现，你真的需要一台计算机和一个很好的算法来解决这样的难题。

你的任务是编写一个程序，文本中不同字符的数量不超过NC，给出这样的字符串，求长度为n的不同的子串有多少个。

举一个例子，当n = 3，NC = 4，字符串为“daababac”。可以在字符串中到到符合条件的子串为："daa"; "aab"; "aba"; "bab"; "bac"。因此，答案应该是5。

**输入：**

第一行：两个数：n，nc

第二行：要搜索的字符串；

你可以假设，最后你所搜索出来符合要求的子串个数不大于16000000；

**输出**

不同子串的个数

**样例输入**

3 4

daababac

**样例输出**

5

**提示**

输入数据巨大，不用cin

**数据巨大，seed不宜太大，看了poj题解，有一个很棒的方法，seed就是当前字符在整个字符串出现过的个数，刚好可以做到一一对应**

**POJ2503-** **Babelfish**

**描述：**

你刚从滑铁卢搬到一个大城市，但是你却不懂他们的语言，还好你有一本词典可以帮助你。

**输入格式**

输入包括高达10万词条，之后是一个空行，然后是长达10万个单词查询。每个字典条目是包含一个英文单词，后面跟一个空格和一个当地语言。没有当地语言在字典中出现超过一次。每行一个词，输入中的每个词至多10小写字母。

**输出格式**

输出翻译成英语的消息，每行一个单词。没有在字典中的外来词应该被译为“en”。

**样例输入**

dog ogday

cat atcay

pig igpay

froot ootfray

loops oopslay

atcay

ittenkay

oopslay

**样例输出**

cat

eh

loops

**这里读入就是个问题，我是这样的：先读入s1，在getchar()，如果get到空格就继续读s2**

**否则就进行翻译步骤**

**这里我使用的方式是双哈希，将字符串的第一个hash作为地址，第二个hash作为数据，线性探查相应单词就会省时省力**

**POJ1002- 487-3279**

**描述：**

企业喜欢用容易被记住的电话号码。让电话号码容易被记住的一个办法是将它写成一个容易记住的单词或者短语。例如，你需要给滑铁卢大学打电话时，可以拨打TUT-GLOP。有时，只将电话号码中部分数字拼写成单词。当你晚上回到酒店，可以通过拨打310-GINO来向Gino's订一份pizza。让电话号码容易被记住的另一个办法是以一种好记的方式对号码的数字进行分组。通过拨打必胜客的“三个十”号码3-10-10-10，你可以从他们那里订pizza。   
  
电话号码的标准格式是七位十进制数，并在第三、第四位数字之间有一个连接符。电话拨号盘提供了从字母到数字的映射，映射关系如下：   
A, B, 和C 映射到 2   
D, E, 和F 映射到 3   
G, H, 和I 映射到 4   
J, K, 和L 映射到 5   
M, N, 和O 映射到 6   
P, R, 和S 映射到 7   
T, U, 和V 映射到 8   
W, X, 和Y 映射到 9

ITS-EASY

4873279  
  
Q和Z没有映射到任何数字，连字符不需要拨号，可以任意添加和删除。 TUT-GLOP的标准格式是888-4567，310-GINO的标准格式是310-4466，3-10-10-10的标准格式是310-1010。   
  
如果两个号码有相同的标准格式，那么他们就是等同的（相同的拨号）   
  
你的公司正在为本地的公司编写一个电话号码薄。作为质量控制的一部分，你想要检查是否有两个和多个公司拥有相同的电话号码。

**输入格式：**

输入的格式是，第一行是一个正整数，指定电话号码薄中号码的数量（最多100000）。余下的每行是一个电话号码。每个电话号码由数字，大写字母（除了Q和Z）以及连接符组成。每个电话号码中只会刚好有7个数字或者字母。

**输出格式：**

对于每个出现重复的号码产生一行输出，输出是号码的标准格式紧跟一个空格然后是它的重复次数。如果存在多个重复的号码，则按照号码的字典升序输出。如果输入数据中没有重复的号码，输出一行：   
No duplicates.

**样例输入：**

12

4873279

ITS-EASY

888-4567

3-10-10-10

888-GLOP

TUT-GLOP

967-11-11

310-GINO

F101010

888-1200

-4-8-7-3-2-7-9-

487-3279

**样例输出：**

310-1010 2

487-3279 4

888-4567 3

**个人认为这道题不用hash，直接开一个足够大的数组即可，只是在读入数据方面要小心**

FZOJ1639魔板

**题目描述**

在魔方风靡全球之后，小Y发明了它的简化版——魔板，如图1所示，魔板由8个同样大小的方块组成，每个方块的颜色均不相同，本题中分别用数字1~8表示，它们可能出现在魔板的任一位置。任一时刻魔板的状态可以用方块的颜色序列表示：从魔板的左上角开始，按顺时针方向依次写下各个颜色块的颜色代号，得到数字序列即可表示此时魔板的状态。例如，序列(1,2,3,4,5,6,7,8)表示如图1所示魔板的状态，这也是本题中魔板的初始状态。

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 8 | 7 | 6 | 5 |

**图 1 魔板的初始状态**

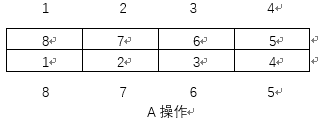
对于魔板，可以施加三种不同的操作，分别以A,B,C标识。具体操作方法如下：

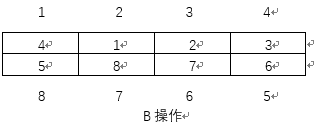
A:上下行互换。

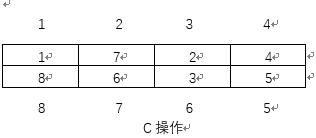
B：每一行同时循环右移一格。

C：中间4个方块顺时针旋转一格。

应用这三种基本操作，可以由任一种状态达到任意另外一种状态。







**图 2 魔板的操作方法**

图2描述了上述3种操作的具体含义，图中方格外面的数字标识魔板的8个方块位置，方格内数字表示此次操作前该小方块所在位置，即：如果位置P对应的方格中数字为I，则表示此次操作前该方块在位置I。

任务一：请编一程序，对于输入的一个目标状态寻找一种操作的序列，使得从初始状态开始，经过此操作序列后使该魔板变为目标状态。

任务二：如果你的程序寻找到的操作序列在300步以内，会得到任务二的分数。

输入数据只有一行，内容是8个以一个空隔分隔的正整数，表示目标状态。输入样例对应的状态如图3所示。输出数据要求第一行输出你的程序寻找到的操作序列的步数L，随后L行是相应的操作序列，每行的行首输出一个字符，代表相应的操作。

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | 6 | 8 | 4 |
| 1 | 3 | 7 | 5 |

**图 3 魔板的输入样例的状态**

**输入**

**输出**

第一行输出你的程序寻找到的操作序列的步数L，随后L行是相应的操作序列，每行的行首输出一个字符，代表相应的操作。

**样例输入**

2 6 8 4 5 7 3 1

**样例输出**

7

B

C

A

B

C

C

B

**学过康托展开，此题的算法就可以大大简化，正好，魔板是标准的全排列，用康托展开的方法算出hash值，并且在这个hash地址上存储相应的步数、是否被遍历等数据，十分方便。**