Hash在ACM竞赛中的应用

瞿绍军
湖南师范大学

排序

- “基于比较的”排序复杂度下界是O(nlogn)
- 但对于某些情况可以更快
- 现有N个整数，范围在0至10000，如何排序？
- 建立数组int num[10001]，初始化为0，num[i]表示有多少个数等于i
- 读入一个数a，则num[a]++
- 可以达到O(n)复杂度，这个思想就是hash

Hash的思想

- 哈希表的基本原理：
- 使用一个下标范围比较大的数组来存储元素，一般通过设计一个函数（哈希函数，即散列函数），使得每个元素的关键字都与一个函数值（即数组下标）相对应，然后用该数组单元来存储对应元素。
- Hash可以用来判重和统计数目

函数构造

- 无定式，方法很多
- 最常见的方法：除余法

  $$H(k) = k \bmod p$$

  (p一般选取适当大的素数)
- 常用的经典字符串Hash后面介绍

Hash中的冲突问题

- 由于不能够保证每个元素的关键字与函数值是一一对应的，因此很有可能出现如下情况：
- “对于不同的元素关键字，Hash函数计算出了相同的函数值”，这就是产生了所谓的“冲突”。
- 换句话说，就是Hash函数把不同的元素分在了相同的下标单元。

冲突解决

方法很多～

常用方法：线性探测再散列技术

即：当 h(k)位置已经存储有元素的时候，依次探查 (h(k)+i) mod S, i=1,2,3…，直到找到空的存储单元为止。其中， S为 数组长度。

特别地，如果将数组扫描一圈仍未发现空单元，则说明哈希表已满，这会带来麻烦，但是，该情况完全可以通过扩大数组范围来避免。

- Hash函数评价标准：
  - 低冲突率
  - 易于编码
- Hash函数特点：
  - 优点：数据存储和查找效率高
    （几乎是常数时间）
  - 缺点：消耗较多内存（内存很便宜哪～）
- Hash主要应用：
  - 查找元素是否属于集合
  - 搜索中的状态表示

HDU-1425 sort

- **Problem Description**
- 给你n个整数，请按从大到小的顺序输出其中前m大的数。
- **Input**
- 每组测试数据有两行，第一行有两个数n,m (0<n,m<1000000)，第二行包含n个各不相同，且都处于区间[-500000,500000]的整数。
- **Output**
- 对每组测试数据按从大到小的顺序输出前m大的数。

初步分析

题目特点：

- 数据量大
- 数据在一定范围范围

- 常规算法的缺陷？
- 是否可以将“数据值”和“存储位置”做某种对应？

- 经典所在：
- 存储完毕，则排序完毕！

再思考（1425加强版）

- 原题描述：

  第二行包含n个各不相同，且都处于区间[-500000,500000]的整数

- 加强版(思考)：

  如果整数允许相同怎么处理？

前m大的数-hdu1280

- Problem Description
- 还记得Gardon给小希布置的那个作业么？（上次比赛的1005）其实小希已经找回了原来的那张数表，现在她想确认一下她的答案是否正确，但是整个的答案是很庞大的表，小希只想让你把答案中最大的M个数告诉她就可以了。
给定一个包含N(N<=3000)个正整数的序列，每个数不超过5000，对它们两两相加得到的N*(N-1)/2个和，求出其中前M大的数(M<=1000)并按从大到小的顺序排列。
- Input
- 输入可能包含多组数据，其中每组数据包括两行：
第一行两个数N和M，
第二行N个数，表示该序列。
- Output
- 对于输入的每组数据，输出M个数，表示结果。输出应当按照从大到小的顺序排列。

整数Hash-hdu1496

- Consider equations having the following form:
- $a*x_1^2+b*x_2^2+c*x_3^2+d*x_4^2=0$
  a, b, c, d are integers from the interval [-50,50] and any of them cannot be 0.
- It is consider a solution a system (x_1,x_2,x_3,x_4) that verifies the equation, x_i is an integer from [-100,100] and x_i != 0, any $i \in \{1,2,3,4\}$.
- Determine how many solutions satisfy the given equation.

- 题意：给a,b,c,d。计算有多少组(x1,x2,x3,x4)满足方程。
- 首先将等式分成两边，写成a*x1^2+b*x2^2 = -c*x3^2-d*x4^2，然后枚举其中一边的所有值存起来(hash存)，再枚举另一边求解。

字符串Hash

- Hash中最常用的是字符串Hash
- 将一个字符串对应到一个整型数值，插入到哈希表
- 对应方法有很多种，甚至可以根据问题的特殊性自己构造，常用的有Rabin-Karp，ELFHash

Rabin-Karp

- 如果字符串中可能出现的字符有k个，则可以将字符串对应到k进制数
- 例如，如果字符串只可能为小写字母组成，则acm就对应到0×26^2+2*26+12
- log(2^63)/log(26) =13.403001373861878677 19
- 当字符串长度不超过13的时候，用long long作关键值类型，加上字符串长度作为限制，每个字符串唯一对应关键值
- 当字符串长度超过13的时候，就要进一步验证

Hash冲突的判断

- 如果链表中有值，不代表一定是你要找的字符串
- 两种方法可以进一步判定
  1,逐个字符比较字符串
  2,建立另外一个或者多个hash函数，比较他们的其他hash关键值是否相同

Flying to the Mars-hdu1800

- 意思是有若干个飞行员，需要在扫帚上练习飞行，每个飞行员具有不同的等级，且等级高的飞行员可以当等级低的飞行员的老师，且每个飞行员至多有且只有一个老师和学生。具有老师和学生关系的飞行员可以在同一把扫帚上练习，并且这个性质具有传递性。即比如有A，B，C，D，E五个飞行员，且等级是A>B>C>D>E，那么可以使A当B的老师，B当C的老师， D 当E的老师，那么A，B，C可以在同一扫帚上练习，D，E在同一把扫帚上练习，这样需要2把扫帚，而如果是A当B的老师，B当C的老师，C当D的老师，D当E的老师，那么只需要一把扫帚。题目所求即所需最少的扫帚数目。

- 假设有若干个飞行员，{{A1,A2,A3...AK},{B1,B2,B3,...Bm}.....{F1,F2,F3.....Fn}}。其已经按照等级由低到高排好序，在同一个集合里的飞行员等级相同。若需要最少数目的扫帚，则只能是{A1,B1....F1},{A2,B2....F2}..这样进行组合，扫帚数目最少。因此决定所需最少扫帚数目的集合是含有飞行员最多的集合，即同一等级数目最多的飞行员集合。因此可以采用STL中的map直接实现。
- 本题的本质是——求相同级别（level）的人最多是几个。

- 此外还可以用字典树、 ELFhash等。

ELFHash算法基本思想

- 算法：
  1.把字符串的每个字符依次相加，每次将加的结果向左移动4位。
  2.如果加的结果大于28位，对结果向右移动24位与原值取异或。

ELFhash

```
inline int ELFhash(char *key)
{
    unsigned long h = 0;
    unsigned long g;
    while( *key )
    {
    h =( h<< 4) + *key++;  //hash左移4位，当前字符ASCII存入hash低四位。
    g = h & 0xf0000000L;  //取最高四位
    if( g ) h ^= g >> 24;
    h &= ~g;
    }
    return h;
}
```

其他字符串Hash方式

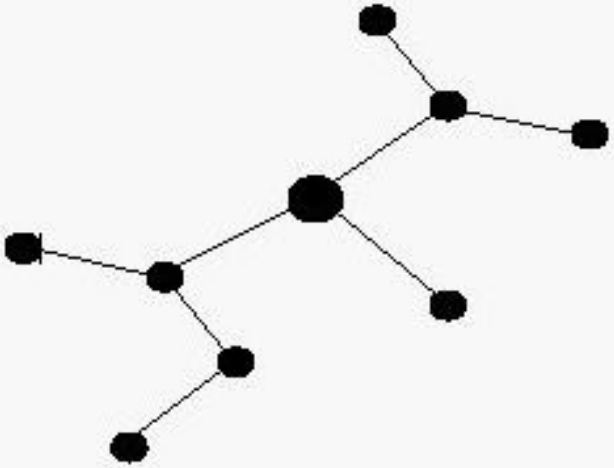
- 加法
  acm -> 'a' + 'c' + 'm'
  优点：计算方便，快
  缺点：容易冲突，acm = cam = bbm
- 乘法
  acm -> 'a' * 'c' * 'm'
  比加法冲突略少
- 上述两种方法可以用于忽略顺序的hash，如判定集合相同。
- 其中乘法可以利用优化将各字母对应到素数

数同构的hash

- 给定两个树，兄弟节点之间无顺序关系，要求判定两树是否同构
- 可以给每个叶子节点赋相同的值，并给出一个公式，用于通过叶子节点的值计算父亲节点的值
- 公式需要满足可交换性，如乘法，但建议使用更复杂的公式。如果公式不满足交换性，也可考虑对所有叶子节点排序后使用公式。

poj-1635 Subway tree systems

- 题意：有不同种地铁路线，每种地铁路线都有一个中心站，从中心站出发，0表示远离中心站，1表示原路返回，有n种测试数据，每组有两行01代码，表示两个路线，问这两个路线是不是一样



```
0010011101001011
0100011011001011
0100101100100111
...
```

Figure 1. To the left: A subway tree system. The larger dot is the central station. To the right: Three out of several possible encodings of exploration tours for the subway system.

poj-1635 Subway tree systems

- 哈希的策略：先随机产生一系列随机数存到数组，接着从根节点出发，递归计算每个子树的哈希值，将子树的哈希值相加然后和父节点自己对应的数组上的随机数相加得到父节点的哈希值。这个计算结果和子树的顺序是没有关系的，所以同构的树一哈希值一定是一样的。对于异构的树，必然在某些节点计算的哈希值不同，由于都是随机产生的一些数字，所以他们相加值和另外一棵树哈希值相同的概率也会非常低。（应该不能完全保证的）

其他结构的hash

- 给定N个点的坐标，求可以构成多少个正方形？长方形？平行四边形？

POJ 2002 Squares

- 思路分析
- 将顶点按x坐标递增排序，若x相同，按y坐标递增排序，然后枚举所有边，对每一条由点p1和p2(根据排序p1 < p2)组成的边按照如下方式可唯一确定一个正方形：
- 1) 将边绕p1逆时针旋转90度得到点p3
- 2) 将边绕p2顺时针旋转90度得到点p4

POJ 2002 Squares

- 则p1 p2 p3 p4组成一个正方形，设p1 = (x1,y1), p2 = (x2, y2)，根据向量的旋转公式可以求出p3, p4的坐标为
- p3 = (y1 - y2 + x1, x2 - x1 + y1)
- p4 = (y1 - y2 + x2, x2 - x1 + y2)

POJ 2002 Squares

- 然后搜索点p3和p4是否存在，若存在则找到一个正方形，计数加1，可以发现总是存在两条边确定的正方形是一样的，也就是说每个正方形会被发现2次，所以要将最后的计数结果除以2.

Poj1971 Parallelogram Counting

- 题目大意：给定n个点的坐标，问组成平行四边形的个数。

Poj1971 Parallelogram Counting

- 解题思路： 若两条线段相互平分，那么他们可以做为平行四边形的对角线。既然是相互平分，也就是说两条线段的中点相同，他们可以确定一个平行四边形。这样本题就有两种解法：
- 一、将每个中点排序，然后统计相同中点的个数N,答案Ans += C(N,2),因为会有重复，所以要用组合公式。

Poj1971 Parallelogram Counting

- 二、用Hash保存每个中点，同一个中点映射到同一个地方。这里可以全部求出来以后按方法一那样求，也可以在发现和前面的某些中点一样时加上前面出现的中点数量，因为它可以和前面那么多条线段组成平行四边形。

推荐题目

- POJ 1200 Crazy Search（Rabin-Karp）
- POJ 1635 Subway tree systems（树同构）
- POJ 1971 Parallelogram Counting（统计平行四边形）
- POJ 2002 Squares（统计正方形）
- POJ 3504 Obfuscation（忽略顺序的字符串hash）
- POJ 1690 (Your)((Term)((Project))) (公式Hash）
- POJ 2549 Sumsets
- Hdoj 1043 Eight