# 离散优化总结

**离散优化是一种常见的高效数据结构，它通过建立数据与存储结构（数组）之间（不一定）一一对应的映射关系来达到对复杂数据的优化。**

**离散优化最重要的一点是建立映射，对于特殊的线段、点而言，这些映射可以是对一个区间的映射，即将某段线或者某块区域映射到数组里面去，从而在计算时降低时间复杂度。**

**Hash优化：是对于字符串和数字的一种优化方式。它通过将数据映射到数组内的某个元素从而达到节省空间的效果。**

**根据hash算法的不同，可能会引起数据的碰撞，即hash(key1)==hash(key2)，会使得数据存储出现错误。有两个方法可以解决：**

1. **拉链法，将hash所对密码指向链表头，每次查找元素遍历整串链表，直到找到该元素为止（编程复杂度较高）**
2. **开地址法，当hash所对密码冲突时，将数据存入另外的位置（可以是下一个空位置，也可以是计算出的任意位置），当然如果使用线性开地址法，只要有一个数据碰撞，那么其余所有数据都很有可能进行至少一次碰撞，非常耗费时间。所以我们运用hash算出另一个位置并存储：**

**while(hashtable[ad]!=0){**

**ad+=ad%3+1;//可以是异于主hash算法的另一hash算法**

**}**

**通常Hash算法分为两个板块：**

1. **查找元素，hash(key)对应的不一定是目标元素，需要对目标进行搜索，推荐使用开地址法进行搜索**
2. **插入元素，与查找同理，在插入之前必须检查此元素是否已被插入，再用开地址法存入相应的地址**

**Hash可用的构造方法**

1. **直接定址**
2. **取模法**
3. **平均取中值**
4. **随机数**
5. **数字分析法，将最有代表特色的位置作为特征码**
6. **折叠法，将数拆分成几部分并求和**
7. **基数法，将低进制数当作高进制数转化为原进制的数，并进行分析，取特征码（两个进制之间应该是互质的关系）**

## 1640: 线段覆盖

时间限制: 1 Sec  内存限制: 128 MB  
提交: 43  解决: 27  
[[提交](http://fzoj.xndxfz.com/JudgeOnline/submitpage.php?id=1640)][[状态](http://fzoj.xndxfz.com/JudgeOnline/problemstatus.php?id=1640)][[讨论版](http://fzoj.xndxfz.com/JudgeOnline/bbs.php?pid=1640)]

## 题目描述

X轴上方有若干条平行于X轴的线段，求这些线段能覆盖到的X轴的总长度？

## 输入

第一行一个数n(n<=1000)，表示线段的个数；  
接下来n行，每行两个整数ai,bi  
（-10^8<=ai,bi<=10^8)，代表一个线段的两个端点。

## 输出

输出覆盖x轴的长度。

## 样例输入

2

10 12

2 4

## 样例输出

4

**将每个点存入数组进行排序，然后遍历所有线段，将线段所覆盖到的点全部记录，输出结果**

## 1234: 图形面积

时间限制: 0 Sec  内存限制: 128 MB  
提交: 11  解决: 2  
[[提交](http://fzoj.xndxfz.com/JudgeOnline/submitpage.php?id=1234)][[状态](http://fzoj.xndxfz.com/JudgeOnline/problemstatus.php?id=1234)][[讨论版](http://fzoj.xndxfz.com/JudgeOnline/bbs.php?pid=1234)]

## 题目描述

        桌面上放了N个矩形，这N个矩形可能有互相覆盖的部分，求它们组成的图形的面积。

## 输入

        输入第一行为一个数N（1≤N≤100），表示矩形的数量。下面N行，每行四个整数，分别表示每个矩形的左下角和右上角的坐标，坐标范围为–10^8到10^8之间的整数。

## 输出

        输出只有一行，一个整数，表示图形的面积。

## 样例输入

3

1 1 4 3

2 -1 3 2

4 0 5 2

## 样例输出

10

**二维离散优化，将图形在x轴上投影的点找出，并且分析相邻两点间的距离（即图形的宽），以及投影这段线的图形在y轴上的投影（即图形的高），ans+=d\*h即可**

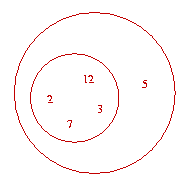
<http://noi.openjudge.cn/ch0305/1551/>

**Sumsets**

**描述**

给出一个整数集合s，找到集合中最大的d，让等式a+b+c=d成立，

其中，a,b,c,d是集合S中不同的元素。



输入

Several S, each consisting of a line containing an integer 1 <= n <= 1000 indicating the number of elements in S, followed by the elements of S, one per line. Each element of S is a distinct integer between -536870912 and +536870911 inclusive. The last line of input contains 0.

输出

For each S, a single line containing d, or a single line containing "no solution".

样例输入

5

2

3

5

7

12

5

2

16

64

256

1024

0

样例输出

12

no solution

**经过变形可得a+b=d-c，先枚举a+b用hash表存储**

**再枚举d-c，如果在hash表中有值,且该值对应的a，b异于d，c，那么将此时的d存起来，取最大值**

## 1807:正方形

<http://noi.openjudge.cn/ch0305/1807/>

描述

给出平面上一些点的坐标，统计由这些点可以组成多少个正方形。注意：正方形的边不一定平行于坐标轴。

输入

输入包括多组测试数据。每组的第一行是一个整数n (1 <= n <= 1000)，表示平面上点的数目，接下来n行，每行包括两个整数，分别给出一个点在平面上的x坐标和y坐标。输入保证：平面上点的位置是两两不同的，而且坐标的绝对值都不大于20000。最后一组输入数据中n = 0，这组数据表示输入的结束，不用进行处理。

输出

对每组输入数据，输出一行，表示这些点能够组成的正方形的数目。

样例输入

4

1 0

0 1

1 1

0 0

9

0 0

1 0

2 0

0 2

1 2

2 2

0 1

1 1

2 1

4

-2 5

3 7

0 0

5 2

0

样例输出

1

6

1

**将每个点用hash表存起来，这里有个小技巧，可以开longlong数组，从而实现一个变量存一个点的功效。然后枚举任意两个未枚举的点，将它们作为正方形的一条边，通过规律我们可以确定两个待定的正方形，寻找计算出的正方形另两个点的位置，搜索hash表，如果都有相应的元素对应，那么即为找到一个正方形。注意：每两个点都会被遍历，即每条边都会被遍历，所以正方形的个数是边数d>>2**

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**

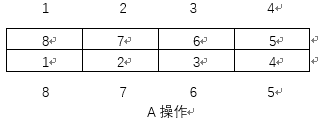
对于魔板，可以施加三种不同的操作，分别以A,B,C标识。具体操作方法如下：

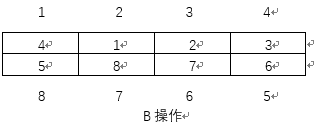
A:上下行互换。

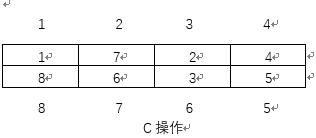
B：每一行同时循环右移一格。

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

**如果魔板数据太小，可用直接地址法，但本题魔板有10^8种可能，不现实。又考虑到在300步内可得到目标魔板，于是想到使用hash表存储已经遍历过的走法。**

**从初始状态开始，进行BFS，取出一个元素，每次考虑三种可能的变化情况，搜索hash表，如果还未考虑此情况，就加入队列，继续搜索，同时记录每种走法的上一步father[]**

**直到程序找到了目标解，那么依次遍历father[]，输出结果即可**