|  |
| --- |
|  |

|  |
| --- |
| ACM/ICPC at Wuhan University |
| Xioumu STL(文字) |
| Created by xioumu, for OpenShield, FreeWaiting |

目录

[Graph 3](#_Toc369772389)

[混合图欧拉路径 3](#_Toc369772390)

[生成树的计数（matrix-Tree） 3](#_Toc369772391)

[关于中国邮递员问题和欧拉图应用 5](#_Toc369772392)

[差分约束 8](#_Toc369772393)

[DataStructure 8](#_Toc369772394)

[扫描线（面积并） 8](#_Toc369772395)

[Lazy函数 9](#_Toc369772396)

[离散 9](#_Toc369772397)

[Computational Geometry 9](#_Toc369772398)

[一些要注意的地方 9](#_Toc369772399)

[计算线段和圆的交点 10](#_Toc369772400)

[数论 11](#_Toc369772401)

[欧拉公式 11](#_Toc369772402)

[欧拉函数 11](#_Toc369772403)

[欧拉定论 12](#_Toc369772404)

[欧拉公式（面，点，边的关系） 12](#_Toc369772405)

[Pick公式 12](#_Toc369772406)

[卡特兰数 13](#_Toc369772407)

[二维旋转 13](#_Toc369772408)

[逆元 14](#_Toc369772409)

[求和公式 15](#_Toc369772410)

[三维旋转 15](#_Toc369772411)

[Pick公式 18](#_Toc369772412)

[MOD公式 18](#_Toc369772413)

[一些公式 18](#_Toc369772414)

[Pólya计数 18](#_Toc369772415)

[DP 19](#_Toc369772416)

[概率DP 19](#_Toc369772417)

[数位DP 19](#_Toc369772418)

# Graph

### 混合图欧拉路径

**当一个有向图所有的度为0时，这个有向图为欧拉图。**

**所以混合图的判断欧拉图只要判断原图的无向边往哪个方向即可，所以可以先以任意形态连边，然后有流量的为正向边，无流量的为反向边。**

**入度 - 出度 > 0 的，和源点连一条(入度 - 出度) / 2 的边， < 0的， 和汇点连边。**

**最后如果最大流等于所有与源点连的流量，即存在一种方法使原图变为欧拉图。**

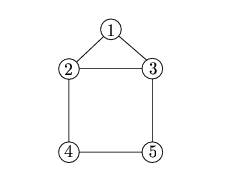
### 生成树的计数（matrix-Tree）

***Matrix-Tree*定理是解决生成树计数问题最有力的武器之一。它首先于1847年被Kirchhoff证明。在介绍定理之前，我们首先明确几个概念：**

1. ***G*的度数矩阵*D*[*G*]是一个*n*\**n*的矩阵，并且满足：当*i*≠*j*时,*dij*=0；当*i*=*j*时，*dij*等于*vi*的度数。**
2. ***G*的邻接矩阵*A*[*G*]也是一个*n*\**n*的矩阵， 并且满足：如果*vi*、*vj*之间有边直接相连，则*aij*=1，否则为0。**

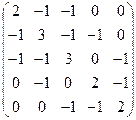
**我们定义*G*的Kirchhoff矩阵(也称为拉普拉斯算子)*C*[*G*]为C[*G*]=*D*[*G*]-*A*[*G*]，则*Matrix-Tree*定理可以描述为：*G*的所有不同的生成树的个数等于其Kirchhoff矩阵*C*[*G*]任何一个*n*-1阶主子式的行列式的绝对值。所谓*n*-1阶主子式，就是对于*r*(1≤*r*≤*n*)，将*C*[*G*]的第*r*行、第*r*列同时去掉后得到的新矩阵，用*Cr*[*G*]表示。**

**下面我们举一个例子来解释*Matrix-Tree*定理。如图*a*所示，*G*是一个由5个点组成的无向图。**

****

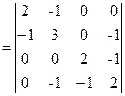
**图*a***

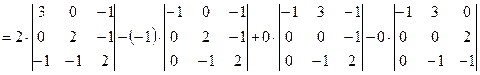
**根据定义，它的Kirchhoff矩阵*C*[*G*]为**

****

**我们取*r*=2，则有：**

**clip_image003**

****

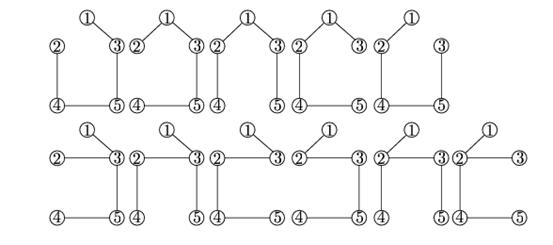
****

**clip_image006**

**clip_image007**

**clip_image008**

**这11棵生成树如图*b*所示。**

****

**图*b***

### 关于中国邮递员问题和欧拉图应用

**中国邮递员问题：**

**1962年有管梅谷先生提出中国邮递员问题（简称CPP）。一个邮递员从邮局出发，要走完他所管辖的每一条街道，可重复走一条街道，然后返回邮局。任何选择一条尽可能短的路线。**

**这个问题可以转化为：给定一个具有非负权的赋权图G，**

**（1）用添加重复边的方法求G的一个Euler赋权母图G\*，使得尽可能小。**

**（2）求G\*的Euler 环游。**

**人们也开始关注另一类似问题，旅行商问题（简称TSP）。TSP是点路优化问题，它是NPC的。而CPP是弧路优化问题，该问题有几种变形，与加权图奇点的最小完全匹配或网络流等价，有多项式算法。**[**[1]**](http://writeblog.csdn.net/Editor/FCKeditor/editor/fckeditor.html?InstanceName=ctl00_ContentPlaceHolder1_EntryEditor1_richTextEditor_richTextEditor&Toolbar=Default#_ftn1)

**欧拉图：**

**图G中经过每条边一次并且仅一次的回路称作欧拉回路。存在欧拉回路的图称为欧拉图。**

**无向图欧拉图判定：**

**无向图G为欧拉图，当且仅当G为连通图且所有顶点的度为偶数。**

**有向图欧拉图判定：**

**有向图G为欧拉图，当且仅当G的基图**[**[2]**](http://writeblog.csdn.net/Editor/FCKeditor/editor/fckeditor.html?InstanceName=ctl00_ContentPlaceHolder1_EntryEditor1_richTextEditor_richTextEditor&Toolbar=Default#_ftn2)**连通，且所有顶点的入度等于出度。**

**欧拉回路性质：**

**性质1设C是欧拉图G中的一个简单回路，将C中的边从图G中删去得到一个新的图G’，则G’的每一个极大连通子图都有一条欧拉回路。**

**性质2设C1、C2是图G的两个没有公共边，但有至少一个公共顶点的简单回路，我们可以将它们合并成一个新的简单回路C’。**

**欧拉回路算法：**

**1             在图G中任意找一个回路C；**

**2             将图G中属于回路C的边删除；**

**3             在残留图的各极大连通子图中分别寻找欧拉回路；**

**4             将各极大连通子图的欧拉回路合并到C中得到图G的欧拉回路。**

**由于该算法执行过程中每条边最多访问两次，因此该算法的时间复杂度为O(|E|)。**

**如果使用递归形式，得注意|E|的问题。使用非递归形式防止栈溢出。**

**如果图 是有向图，我们仍然可以使用以上算法。**

[**http://acm.hdu.edu.cn/showproblem.php?pid=1116**](http://acm.hdu.edu.cn/showproblem.php?pid=1116)**有向图欧拉图和半欧拉图判定**

[**http://acm.pku.edu.cn/JudgeOnline/problem?id=2337**](http://acm.pku.edu.cn/JudgeOnline/problem?id=2337)**输出路径**

**中国邮递员问题①：**

**一个邮递员从邮局出发，要走完他所管辖的每一条街道，可重复走一条街道，然后返回邮局。所有街道都是双向通行的，且每条街道都有一个长度值。任何选择一条尽可能短的路线。**

**分析：**

**双向连通，即给定无向图G。**

**如果G不连通，则无解。**

**如果G是欧拉图，则显然欧拉回路就是最优路线。**

**如果G连通，但不是欧拉图，说明图中有奇点**[**[3]**](http://writeblog.csdn.net/Editor/FCKeditor/editor/fckeditor.html?InstanceName=ctl00_ContentPlaceHolder1_EntryEditor1_richTextEditor_richTextEditor&Toolbar=Default#_ftn3)**。奇点都是成对出现的，证明从略。**

**对于最简单情况，即2个奇点，设（u，v）。我们可以在G中对（u，v）求最短路径R，构造出新图G’ = G ∪ R。此时G’就是欧拉图。**

**证明：u和v加上了一条边，度加一，改变了奇偶性。而R中其他点度加二，奇偶性不变。**

**由此可知，加一次R，能够减少两个奇点。推广到k个奇点的情况，加k/2个R就能使度全为偶数。**

**接下的问题是求一个k个奇点的配对方案，使得k/2个路径总长度最小。**

**这个就是无向完全图最小权匹配问题。有一种Edmonds算法，时间复杂度O（N^3）。**[**[4]**](http://writeblog.csdn.net/Editor/FCKeditor/editor/fckeditor.html?InstanceName=ctl00_ContentPlaceHolder1_EntryEditor1_richTextEditor_richTextEditor&Toolbar=Default#_ftn4)

**也可转换为二分图，用松弛优化的KM算法，时间复杂度也是O（N^3）。**

**完整的算法流程如下：**

**1         如果G是连通图，转2，否则返回无解并结束；**

**2         检查G中的奇点，构成图H的顶点集；**

**3         求出G中每对奇点之间的最短路径长度，作为图H对应顶点间的边权；**

**4         对H进行最小权匹配；**

**5         把最小权匹配里的每一条匹配边代表的路径，加入到图G中得到图G’；**

**6         在G’中求欧拉回路，即所求的最优路线。**

**中国邮递员问题②：**

**和①相似，只是所有街道都是单向通行的。**

**分析：**

**单向连通，即给定有向图G。**

**和①的分析一样，我们来讨论如何从G转换为欧拉图G’。**

**首先计算每个顶点v的入度与出度之差 d’（v）。如果G中所有的v都有d’（v）=0，那么G中已经存在欧拉回路。**

**d’（v）>0 说明得加上出度。d’（v）<0说明得加上入度。**

**而当d’（v）=0，则不能做任何新增路径的端点。**

**可以看出这个模型很像网络流模型。**

**顶点d’（v）>0对应于网络流模型中的源点，它发出d’（v）个单位的流；顶点d’（v）<0对应于网络流模型中的汇点，它接收-d’（v）个单位的流；而d’（v）=0的顶点，则对应于网络流模型中的中间结点，它接收的流量等于发出的流量。在原问题中还要求增加的路径总长度最小，我们可以给网络中每条边的费用值 设为图 中对应边的长度。这样，在网络中求最小费用最大流，即可使总费用最小。**

**这样构造网络N：**

**1         其顶点集为图G的所有顶点，以及附加的超级源 和超级汇 ；**

**2         对于图G中每一条边(u,v)，在N中连边(u,v)，容量为∞，费用为该边的长度；**

**3         从源点 向所有d’(v)>0的顶点v连边(s,v)，容量为d’(v)，费用为0；**

**4         从所有d’(v)<0的顶点 向汇点t连边(u,t)，容量为-d’(v)，费用为0。**

**完整的算法流程如下：**

**1         如果G的基图连通且所有顶点的入、出度均不为0，转2，否则返回无解并结束；**

**2         计算所有顶点v的d’(v)值；**

**3         构造网络N；**

**4         在网络N中求最小费用最大流；**

**5         对N中每一条流量f(u,v)的边(u,v)，在图G中增加f(u,v)次得到G’；**

**6         在G’中求欧拉回路，即为所求的最优路线。**

**NPC问题：**

**如果部分街道能够双向通行，部分街道只能单向通行。这个问题已被证明是NPC的。**[**[5]**](http://writeblog.csdn.net/Editor/FCKeditor/editor/fckeditor.html?InstanceName=ctl00_ContentPlaceHolder1_EntryEditor1_richTextEditor_richTextEditor&Toolbar=Default#_ftn5)

### 差分约束

求解差分约束系统，可以转化成[图论](https://zh.wikipedia.org/wiki/%E5%9C%96%E8%AB%96)的单源最短[路径](https://zh.wikipedia.org/wiki/%E8%B7%AF%E5%BE%91)问题。观察xj-xi<=bk，会发现它类似最短路中的三角不等式d[v] <=d[u]+w[u,v]，即d[v]-d[u]<=w[u,v]。因此，以每个变量xi为结点，对于约束条件xj-xi<=bk，连接一条边（i,j），边权为bk。再增加一个[原点](https://zh.wikipedia.org/wiki/%E5%8E%9F%E9%BB%9E)（s,s）与所有[定点](https://zh.wikipedia.org/w/index.php?title=%E5%AE%9A%E9%BB%9E&action=edit&redlink=1)相连，[边权](https://zh.wikipedia.org/w/index.php?title=%E9%82%8A%E6%AC%8A&action=edit&redlink=1)均为0。对这个图以s为原点运行[Bellman-ford算法](https://zh.wikipedia.org/wiki/Bellman-ford%E7%AE%97%E6%B3%95)（或[SPFA算法](https://zh.wikipedia.org/wiki/SPFA%E7%AE%97%E6%B3%95)），最终{d[i]}即为一组可行解。

# DataStructure

## 扫描线（面积并）

注意线段树中的每个点要代表一个左闭右开的区间！

## Lazy函数

Lazy[t] = true 表示需要向下传值，但节点t的值已经更新完。

也就是说t节点的值与lazy[t]无关。

## 离散

假如离散线段的成一个点的话，记得两边端点和左边端点左边的点和右边端点右边的点各离散一个点， 一条线段总共离散成4个点。否则会出问题。

# Computational Geometry

## 一些要注意的地方

1.判大小等于要用Sgn()

Int sgn(double x) {

Return (x > esp) - (x < -esp);

}

2.Sqrt：y = sqrt(x) x < 0的时候会出现混乱，所以:

if (d < 0)

res = 0.0;

else

res = sqrt(d);

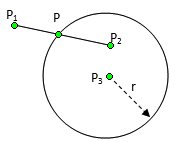
1. asin(x), acos(x) x ->[-1, 1];

Double trim(double d, double l = 1.0) {

Return d > l ? l : (d < -1 ? -1 : d);

}

## 计算线段和圆的交点



    设线段的两个端点分别是P1(x1,y1)和P2(x2,y2)，圆的圆心在P3(x3,y3)，半径为r，那么如果有交点P(x,y)的话

C:\Users\mu\AppData\Local\Temp\msohtmlclip1\02\clip_image002.gif

    其中，u在0到1之间，转换成各个坐标

C:\Users\mu\AppData\Local\Temp\msohtmlclip1\02\clip_image003.gif

    由于P也在圆上，所以

C:\Users\mu\AppData\Local\Temp\msohtmlclip1\02\clip_image004.gif

    联立上面的公式，可以得到

C:\Users\mu\AppData\Local\Temp\msohtmlclip1\02\clip_image005.gif

    其中

C:\Users\mu\AppData\Local\Temp\msohtmlclip1\02\clip_image006.gif

    解一元二次方程，可以得到

C:\Users\mu\AppData\Local\Temp\msohtmlclip1\02\clip_image007.gif

    根据

C:\Users\mu\AppData\Local\Temp\msohtmlclip1\02\clip_image008.gif

的结果，可以判断**线段所在直线**和圆的相交情况

* 如果小于0，表示没有交点
* 如果等于0，表示相切，只有一个交点
* 如果大于0，表示有两个交点

    针对P1和P2之间的线段，根据计算出的u值，有5种结果

* 如果线段和圆没有交点，而且都在圆的外面的话，则u的两个解都是小于0或者大于1的
* 如果线段和圆没有交点，而且都在圆的里面的话，u的两个解符号相反，一个小于0，一个大于1
* 如果线段和圆只有一个交点，则u值中有一个是在0和1之间，另一个不是
* 如果线段和圆有两个交点，则u值得两个解都在0和1之间
* 如果线段和圆相切，则u值只有1个解，且在0和1之间

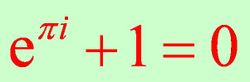
# 数论

### 欧拉公式

**（欧拉公式）如果一个连通的平面图有*n*个点，*m*条边和*f*个面，那么*f*=*m*-*n*+2**

### 欧拉函数

**e^ix=cosx+isinx**

**[](http://baike.baidu.com/albums/398/398/0/0.html#0$a6c7d717a2b9ad2dc93d6d89)**

**自然对数的底e，圆周率π，两个单位：虚数单位i和自然数的单位1，以及被称为人类伟大发现之一的0。**[**数学家**](http://baike.baidu.com/view/66878.htm)**们评价它是“上帝创造的公式”**

**计算机生成了可选文字: 御；",
沪（,,7)
1,;2，杯“，川
－……，
耳‘足耐l{J所有互异索因数，那么
k
,,7fl(I-
1
)
i=l**

**欧拉φ函数：φ(n)是所有小于n的正整数里，和n互素的整数的个数。n是一个正整数。**

|  |  |
| --- | --- |
| **欧拉证明了下面这个式子：** |  |

**如果n的标准素因子分解式是p1^a1\*p2^a2\*……\*pm^am，其中众pj(j=1,2,……,m)都是素数，而且两两不等。则有**

**φ(n)=n(1-1/p1)(1-1/p2)……(1-1/pm)**

**利用容斥原理可以证明它。**

### 欧拉定论

欧拉定理的内容是：如果a和n互质，那么aφ(n)=1(mod n)；对于任意a, n和较大的b，有ab=aφ(n)+b mod φ(n)(mod n)

### 欧拉公式（面，点，边的关系）

χ=V-E+F来计算，V-vertices表示顶点数，E-edges表示边数，F-faces表示围成的面数。于是面的个数可以这样计算：对于几个相连的圆，其内有F=E-V+1个面，其中1是平面图情况的欧拉欧拉示性数(Euler characteristic)；

### Pick公式

**定理一：Pick 定理，1899年**

**设 Γ 为平面上以格子点为顶点之单纯多边形，则其面积为**

**\begin{displaymath}
A=\frac{b}{2}+i-1\eqno{(8)}
\end{displaymath}**

**其中 *b* 为边界上的格子点数，*i* 为内部的格子点数。(8)式叫做 Pick 公式。**

**定理二：（棱线定理，Edge Theorem）**

**对于任意连通的三角形化的平面图枝，其棱线的个数恒为**

**\begin{displaymath}E=2b+3i-3\eqno{(13)}\end{displaymath}**

**其中 *b* 与 *i* 分别表示图枝边界上的顶点数与内部的顶点数。**

### 卡特兰数

**卡特兰数 ： 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420, 24466267020, 91482563640, 343059613650, 1289904147324, 4861946401452,**

**令h(1)=1,h(0)=1，catalan数满足递归式：**

**h(n)= h(0)\*h(n-1)+h(1)\*h(n-2) + ... + h(n-1)h(0) (其中n>=2)**

**另类递归式：**

**h(n)=h(n-1)\*(4\*n-2)/(n+1);**

**该递推关系的解为：**

**h(n)=C(2n,n)/(n+1) (n=1,2,3,...)**

### 二维旋转

**看下面的矩阵**

**[clip_image002[4]](http://gaoyunxiang.com/wp-content/uploads/2010/02/clip_image0024.gif)**

**这个是二维向量的旋转矩阵，它可以将一个向量逆时针旋转一个角度。**

**将其变形，变会得到二维向量的顺时针旋转的形式：**

**[clip_image002[16]](http://gaoyunxiang.com/wp-content/uploads/2010/02/clip_image00216.gif)**

**这里要注意一种特殊的情况，就是当角度为90度的时候，sin和cos的结果只有1, 0, -1三种可能性。所以这个矩阵可以改写成特殊的形式，其意义在于用这样的旋转操作，不会产生精度问题。sin和cos的运算的精度是比较低的，能少用则尽量少用。在计算几何中的向量旋转操作，大部分都可以通过变形，只用到旋转90度，从而避免精度问题。**

**===3维====**

**围绕轴u = (ux, uy, uz)来旋转的矩阵代码如下：**

1. **double a[3][3] = {**
2. **{SQR(ux) + (1 – SQR(ux)) \* c, ux \* uy \* (1 – c) – uz \* s, ux \* uz \* (1 – c) + uy \* s},**
3. **{ux \* uy \* (1 – c) + uz \* s, SQR(uy) + (1 – SQR(uy)) \* c, uy \* uz \* (1 – c) – ux \* s},**
4. **{ux \* uz \* (1 – c) – uy \* s, uy \* uz \* (1 – c) + ux \* s, SQR(uz) + (1 – SQR(uz)) \* c}**
5. **};**

**要注意这个ux \* ux + uy \* uy + uz \* uz = 1**

**有一个比较有意思的问题就是，知道旋转矩阵之后，怎么来确定向量u呢？**

**我们做如下变形之后，可以得到：**

**Ru = Iu –> (R – I)u = 0**

**也就是说我们要找到一个非0的向量，使得他和一个矩阵乘起来得到一个空矩阵。这个问题可以用高斯消元来解决。实际上我们就是要解一个线性方程组。**

### 逆元

**定义 如果*ab*≡1(mod *m*), 则称*b*是*a*的模*m*逆,记作*a*的模*m*逆是方程*ax*≡1(mod *m*)的解.**

**即gcd(a, m) == 1, 即(a, m)互质**

**例：求5的模7逆**

**做辗转相除法, 求得整数*b,k*使得 5*b*+7*k*=1, 则*b*是5的模7逆.**

**计算如下:**

**7=5+2,  5=2×2+1.**

**回代    1=5-2×2=5-2×(7-5)= 3×5-2×7,**

**得 5 -1≡3(mod7).**

**例：求21的模73逆**

**做辗转相除法, 求得整数*b,k*使得 21*b*+73*k*=1, 则*b*是21的模73逆.**

**计算如下:**

**73=21\*3+10**

**21=10\*2+1**

**回代    1=21-10\*2**

**1=21-（73-21\*3）\*2**

**=21-73\*2+6\*21**

**=7\*21-73\*2**

**得 21 -1≡7(mod73).**

### 求和公式

1^4+2^4+3^4+4^4+……+n^4 = n(n+1)(2n+1)(3n^2+3n-1)/30

1^3+2^3+3^3+4^3+……+n^3=[n(n+1)/2]^2

1^2+2^2+3^2+4^4+……+n^2=n(n+1)(2n+1)/6

1+2+3+4+……+n=n(n+1)/2

### 三维旋转

计算机生成了可选文字:
平移：
饮ty仪

计算机生成了可选文字:
拉伸：
b

C = cos(angle), S = sin(angle).

绕(0, 0, 0) - (X, Y, Z) 向量顺时针旋转angle (即从(x,y,z)向(0,0,0)点看,顺时针旋转)

计算机生成了可选文字:
旋转：
州
、．l...lee了
0001
r"，承，一。
．人州1一。＋.tS
！月：才，(，一0一凡’
戈0
通洲。（1一O一才声
c＋可（l一C)
人才，(1一0＋汉：S
0
月：.':(1一O＋人s
凡人（1一O一月：S
c＋对（l一O
0

matrix get\_rotate(double x, double y, double z, double d) {

matrix now;

now.set\_one();

d = -d / 180.0 \* pi;

double c = cos(d), s = sin(d);

double l = sqrt(x \* x + y \* y + z \* z);

x /= l, y /= l, z /= l;

now.ar[0][0] = c + x \* x \* (1 - c);

now.ar[0][1] = x \* y \* (1 - c) - z \* s;

now.ar[0][2] = x \* z \* (1 - c) + y \* s;

now.ar[1][0] = x \* y \* (1 - c) + z \* s;

now.ar[1][1] = c + y \* y \* (1 - c);

now.ar[1][2] = y \* z \* (1 - c) - x \* s;

now.ar[2][0] = x \* z \* (1 - c) - y \* s;

now.ar[2][1] = y \* z \* (1 - c) + x \* s;

now.ar[2][2] = c + z \* z \* (1 - c);

now.ar[3][3] = 1;

return now;

}

### Pick公式

**定理一：Pick 定理，1899年**

设 Γ 为平面上以格子点为顶点之单纯多边形，则其面积为

\begin{displaymath}
A=\frac{b}{2}+i-1\eqno{(8)}
\end{displaymath}

其中 *b* 为边界上的格子点数，*i* 为内部的格子点数。(8)式叫做 Pick 公式。

**定理二：（棱线定理，Edge Theorem）**

对于任意连通的三角形化的平面图枝，其棱线的个数恒为

\begin{displaymath}E=2b+3i-3\eqno{(13)}\end{displaymath}

其中 *b* 与 *i* 分别表示图枝边界上的顶点数与内部的顶点数。

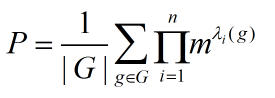
### MOD公式

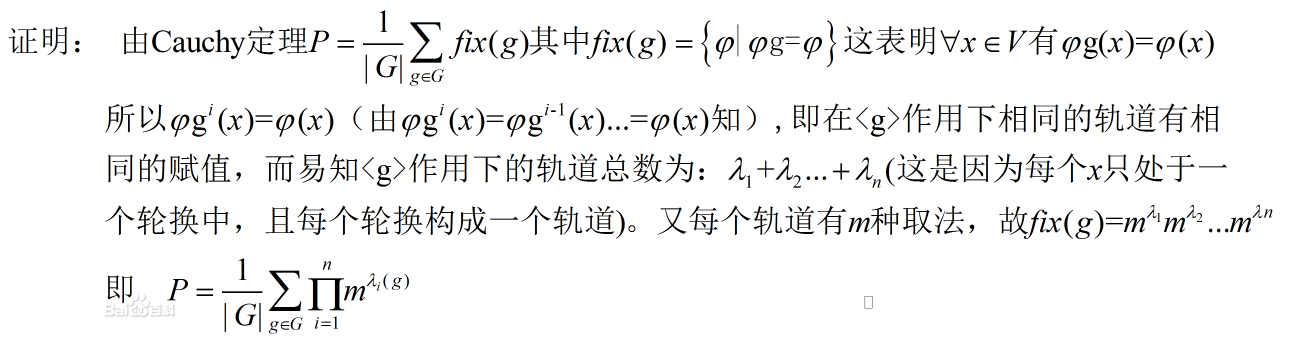
A/B %M = A % (B \* M) / B

### 一些公式

( abs(a + b) + abs(a - b) ) / 2 = max( abs(a) , abs(b) )

### Pólya计数





# DP

### 概率DP

**要倒着DP，有环可用高斯消元，或者设X直接把X给化简到一边。**

### 数位DP

两种:

1.先预处理出来一些信息（如：f(10),f(100),f(1000)的值，一般用递推预处理），然后直接递归算。

2.只需预付处理少量信息，然后直接记忆化搜索。

优点:只需要想清楚各种转移情况即可，思路较清晰

缺点:情况多的话，较容易想乱。