ACM/ICPC at Wuhan University

Xioumu STL(文字)

Created by xioumu, for OpenShield

目录

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** $i \neq j$ **时,**$d_{ij}=0$**；当** $i=j$ **时，**d_{ij} **等于** v_i **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** v_i、v_j **之间有边直接相连，则** $a_{ij}=1$**，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** $r(1 \leq r \leq n)$**，将** ***C*[*G*]的第** ***r*** **行、第** ***r*** **列同时去掉后得到的新矩阵，用** $C_r[G]$**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：**

$$|C_2[G]|$$

$$= \begin{vmatrix} 2 & -1 & 0 & 0 \\ -1 & 3 & 0 & -1 \\ 0 & 0 & 2 & -1 \\ 0 & -1 & -1 & 2 \end{vmatrix}$$

$$= 2 \cdot \begin{vmatrix} 3 & 0 & -1 \\ 0 & 2 & -1 \\ -1 & -1 & 2 \end{vmatrix} - (-1) \cdot \begin{vmatrix} -1 & 0 & -1 \\ 0 & 2 & -1 \\ 0 & -1 & 2 \end{vmatrix} + 0 \cdot \begin{vmatrix} -1 & 3 & -1 \\ 0 & 0 & -1 \\ 0 & -1 & 2 \end{vmatrix} - 0 \cdot \begin{vmatrix} -1 & 3 & 0 \\ 0 & 0 & 2 \\ 0 & -1 & -1 \end{vmatrix}$$

$$= 2\left(3 \cdot \begin{vmatrix} 2 & -1 \\ -1 & 2 \end{vmatrix} - 0 \cdot \begin{vmatrix} & \\ & \end{vmatrix} + (-1) \cdot \begin{vmatrix} 0 & 2 \\ -1 & -1 \end{vmatrix}\right) - (-1)\left((-1) \cdot \begin{vmatrix} 2 & -1 \\ -1 & 2 \end{vmatrix} - 0 \cdot \begin{vmatrix} & \\ & \end{vmatrix} + (-1) \cdot \begin{vmatrix} 0 & 2 \\ 0 & -1 \end{vmatrix}\right)$$

$$= 2(3(4-1) - 0 + (-1)(0-(-2))) - (-1)((-1)(4-1) - 0 + (-1)(0-0)) + 0 - 0$$

$$= 2(9-2) - (-1)(-1)(3) = 11$$

这 11 棵生成树如图 *b* 所示。

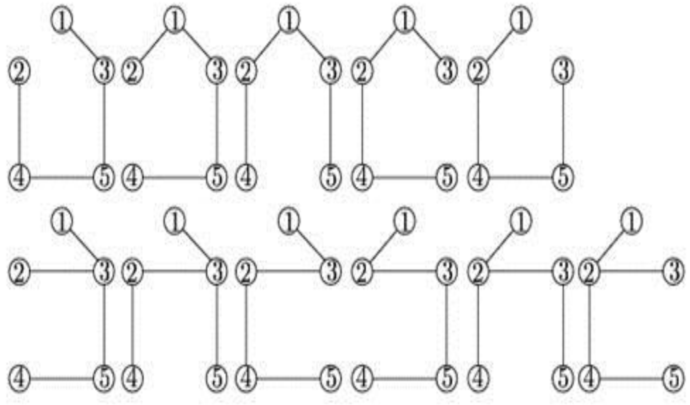

图 *b*

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]

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]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 有向图欧拉图和半欧拉图判定

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]。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]

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]

DataStructure

Computational Geometry

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

$$e^{\pi i}+1=0$$

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。数学家们评价它是"上帝创造的公式"

设 $p_1^{e_1}, p_2^{e_2}, p_3^{e_3}, p_4^{e_4}, \ldots\ldots, p_k^{e_k}$ 是 m 的所有互异素因数，那么

$$\phi(m) = m\prod_{i=1}^{k}(1-\frac{1}{p_i})$$

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。

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

$$A=\frac{b}{2}+i-1 \qquad (8)$$

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

$$E = 2b + 3i - 3 \qquad (12)$$

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

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

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：

$$\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

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：

```
double a[3][3] = {
    {SQR(ux) + (1 – SQR(ux)) * c, ux * uy * (1 – c) – uz * s, ux * uz * (1 – c) + uy * s},
    {ux * uy * (1 – c) + uz * s, SQR(uy) + (1 – SQR(uy)) * c, uy * uz * (1 – c) – ux * s},
    {ux * uz * (1 – c) – uy * s, uy * uz * (1 – c) + ux * s, SQR(uz) + (1 – SQR(uz)) * c}
  };
```

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 $ab\equiv1(\mathrm{mod}\ m)$，则称 b 是 a 的模 m 逆,记作 a 的模 m 逆是方程 $ax\equiv1(\mathrm{mod}\ m)$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 $5b+7k=1$，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 $5^{-1}\equiv3(\mathrm{mod}7)$.

例：求 21 的模 73 逆

做辗转相除法，求得整数 b,k 使得 $21b+73k=1$，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

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。