**一笔画问题或欧拉路径**

**1、问题概述**

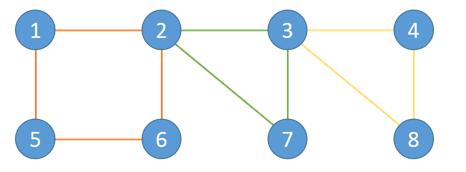
欧拉路是指从图中任意一个点开始到图中任意一个点结束的路径，并且图中每条边通过的且只通过一次。

**2、求解算法思想**

首先，确定这是一个连通图

若是无向图，则这个图的度数为奇数的点的个数必须是0或2；若是有向图，则要么所有点的入度和出度相等，要么有且只有两个点的入度分别比出度大1和少1。

**3、举例说明求解过程**



在这个例子中：

L1: 1-2-6-5-1

L2: 2-3-7-2

L3: 3-4-8-3

第一步时我们将L1压入栈S，同时我们用一个数组Path来记录我们出栈的顺序：

S: [1 2 6 5 1]

Path:

然后出栈到节点2时我们发现了2有其他路径，于是我们把2的另一条路径加入：

S: 1 [2 3 7 2]

Path: 1 5 6

此时L2已经走完，然后再开始弹出元素，直到我们发现3有其他路径，同样压入栈：

S: 1 2 [3 4 8 3]

Path: 1 5 6 2 7

之后依次弹出剩下的元素：

S:

Path: 1 5 6 2 7 3 8 4 3 2 1

此时的Path就正好是我们需要的欧拉路径

**4、算法具体步骤**

一、对于无向图，判断度数为奇数的点的个数，若为0，则设任意一点为起点，若为2，则从这2个点中任取一个作为起点；对于有向图，判断入度和出度不同的点的个数，若为0，则设任意一点为起点，若为2，则设入度比出度小1的点为起点，另一点为终点。具体起点的选择要视题目要求而定。

二、从起点开始进行递归：对于当前节点x，扫描与x相连的所有边，当扫描到一条(x,y)时，删除该边，并递归y。扫描完所有边后，将x加入答案队列。

三、倒序输出答案队列。（因为这里是倒序输出，我们可以用栈来存储答案，当然用双端队列也可以）

**5、性能分析**

这个算法会遍历所有边，因此整个算法需要线性时间为O(|E|)。

**哈密尔顿问题**

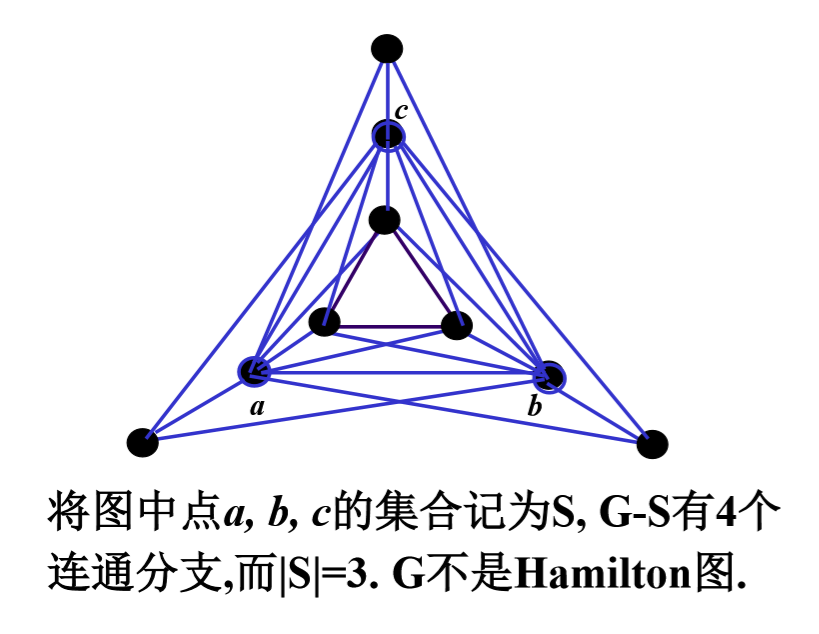
**1、问题概述**

由指定的起点前往指定的终点，途中经过所有其他节点且只经过一次。

**2、求解算法思想**

设一个无向图中有N个顶点,若所有顶点的度数大于等于N/2,则哈密顿回路一定存在。

**3、举例说明求解过程**



**4、算法具体步骤**

1、初始化,令s = 1,t为s的任意一个邻接点.

2、如果ans[]中元素的个数小于n,则从t开始向外扩展,如果有可扩展点v,放入ans[]的尾部,并且t=v,并继续扩展,如无法扩展进入步骤3.

3、将当前得到的ans[]倒置,s和t互换,从t开始向外扩展,如果有可扩展点v,放入ans[]尾部,并且t=v,并继续扩展.如无法扩展进入步骤4.

4、如果当前s和t相邻,进入步骤5.否则,遍历ans[],寻找点ans[i],使得ans[i]与t相连并且ans[i +１]与ｓ相连,将从ans[i + 1]到t部分的ans[]倒置,t=ans[i +１],进如步骤5.

5、如果当前ans[]中元素的个数等于n,算法结束,ans[]中保存了哈密顿回路(可看情况是否加入点s).否则,如果s与t连通,但是ans[]中的元素的个数小于n,则遍历ans[],寻找点ans[i],使得ans[i]与ans[]外的一点(j)相连,则令s=ans[i - 1],t = j,将ans[]中s到ans[i - 1]部分的ans[]倒置,将ans[]中的ans[i]到t的部分倒置,将点j加入到ans[]的尾部,转步骤2.

**5、性能分析**

如果说每次到步骤5算一轮的话,那么由于每一轮当中至少有一个节点被加入到路径S -> T中,所以总的轮数肯定不超过n轮,所以时间复杂度为O(n^2)。

**中国邮递员问题**

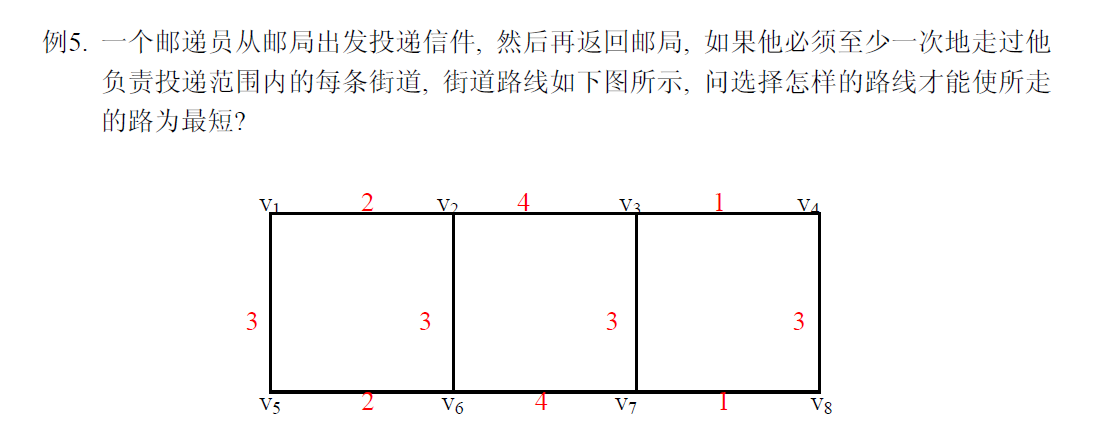
**1、问题概述**

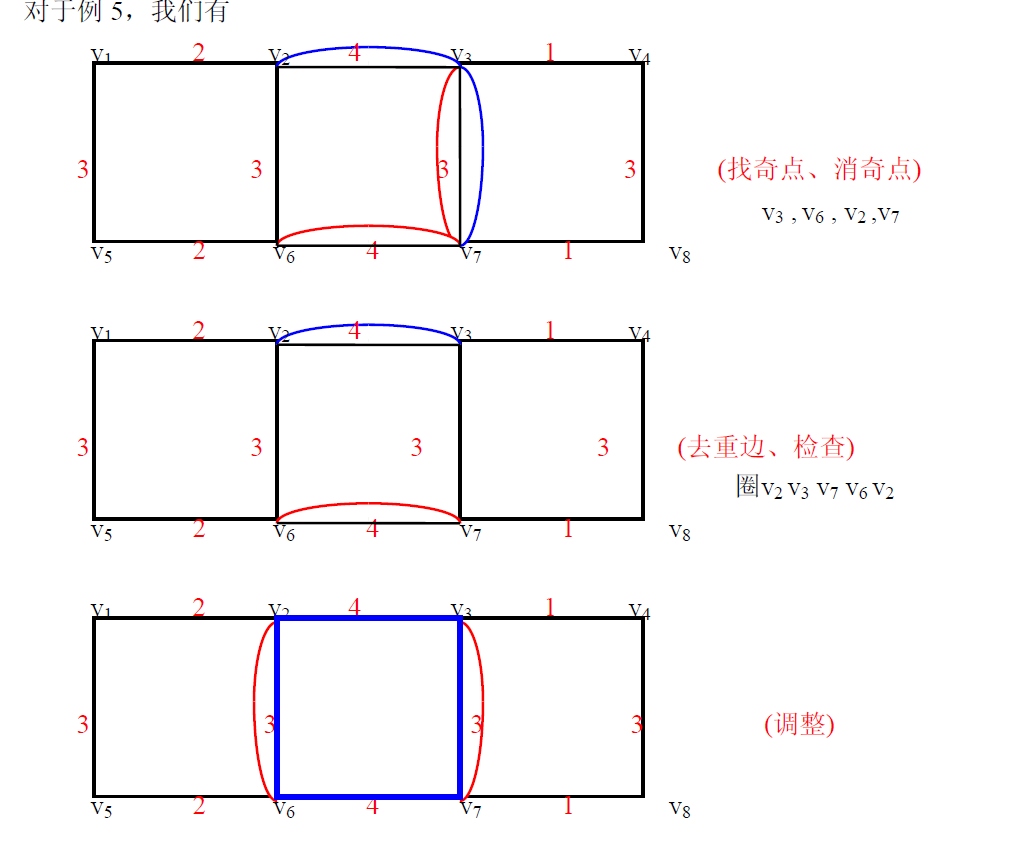
邮递员每天从邮局出发，走遍该地区所有街道再返回邮局，问题是他应如何安排送信的路线可以使所走的总路程最短。

**2、求解算法思想**

由命题1，简单图G的奇点个数为偶数，可设为v1 , v2 , …, v2k, 对每个1≤ i ≤k,找v2i − 1 至v2i的链pi，将pi的边重复一次。对于每一个pi而且除两端点外，其它点保持原奇偶性，即此时图中无奇点。再将添加边多于1条的边, 成对删去, 仍保持点的奇偶性。由命题2，存在伪Euler圈。将添加的边组成一个可行集，由命题3检验是否为最优，如果非最优的，则存在一圈不满足命题3, 将该圈中非重复边重复一次，重复边删去一次，图的各点奇偶性不变。

**3、举例说明求解过程**





**4、算法具体步骤**

（1） 建立街区无向网的邻接矩阵；

（2） 求各顶点的度数；

（3） 求出所有奇度点；

（4） 求出每一个奇度点到其它奇度点的最短路径；

（5） 找出距离最短的添加方案；

（6） 根据最佳方案添加边，对图进行修改，使之满足一笔画的条件；

（7） 对图进行一笔画，并输出；

**5、性能分析等**



**旅行推销员问题**

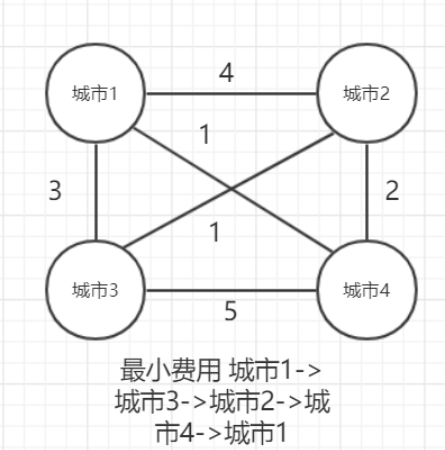
**1、问题概述**

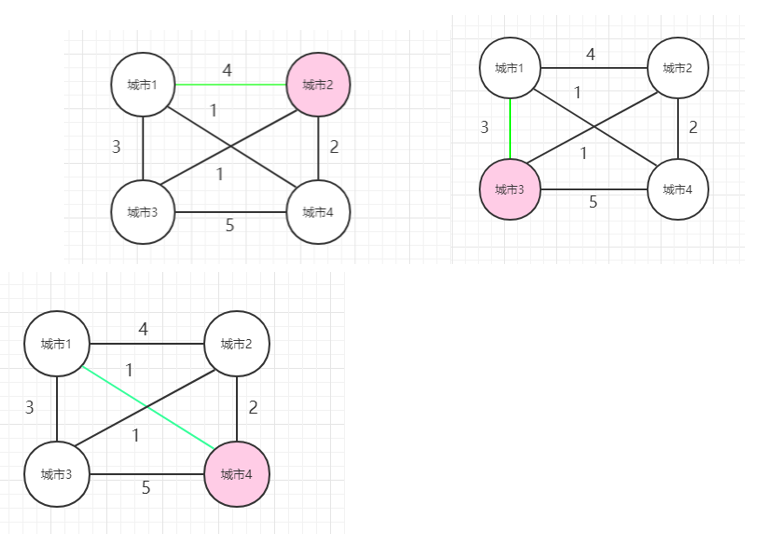
一个商品推销员要去若干个城市推销商品，该推销员从一个城市出发，需要经过所有城市后，回到出发地。应如何选择行进路线，以使总的行程最短。

**2、求解算法思想**

贪心寻找最近的未访问的邻点

**3、举例说明求解过程**

****



1、 从1出发，到2，然后再从2出发，经过[3,4]这几个城市，然后回到1，使得花费最少。

2、 从1出发，到3，然后再从3出发，经过[2,4]这几个城市，然后回到1，使得花费最少。

3、 从1出发，到4，然后再从4出发，经过[2,3]这几个城市，然后回到1，使得花费最少。

上面也提到了最优结果通过表来保留：设置一个二维的动态规划表dp , dp[1]{2，3，4}表示从1号城市出发，经过2，3，4 回到1花费最少。

要求上面三个方案的最小值意味：（D12表示1到2的距离，其他同理）

dp[1] [{2,3,4}] =  min{ D12+dp[2]{3,4} ，D13+dp[3]{2,4} ， D14+dp[4]{2,3}}

由于D12，D13，D14是已知的，那么我们现在的目的就是求dp[2]{3,4}，dp[3]{2,4}，dp[4]{2,3}，

我们可以列出：（这里只列出dp[2]{3,4} ，其他两个类似）

　　　　dp[2]{3,4} = min{ D23+dp[3]{4} ，D24+dp[4][3}}

　　　　dp[3]{4}]= D43+dp[4]{}

　　　　dp[4]{}=D41

那么经过慢慢的分解，我们知道了我们已知了从4到1的最小花费，那么就可以推出从3出发经过4回到1的花费。

**4、算法具体步骤**

1、易知从哪个城市出发其最短路径都是一样的，故假设从城市1出发。假设已经经过了几个城市，我们需要记录此时位于的城市，以及未访问的城市的集合。

2、以dp[{V}][init]表示从init点开始，要经过集合V中所有点的距离。dis[init][i]表示城市init到城市i的距离。

3、其状态转移方程为dp[{V}][init]=min(dp[{V}][init], dp[{V-i}][i]+dis[init][i])。即假设处于城市init，欲前往下一个城市i，如果在城市i的状态dp[{V-i}][i]加上城市init到城市i的距离小于当前最小值，则前往城市i。

4、我们怎么存储未访问的城市的集合？一个方法是以二进制01表示该城市是否被访问过，如s=111110，城市1对应的位为0，其他城市对应的位为1，则表示城市6到城市2都还未访问，城市1已访问过。例如在出发点1时，要判断城市2是否访问过，若s&(1<<1)为1则表示城市2未被访问过，若去城市2，则集合V变为s&(~(1<<1))。

5、以数组path[s][init]记录在城市init，未访问城市集合为s时，下一个城市的最优选择，以存储最优路径。

**5、性能分析等**

由于每个k规模的子问题需要进行k-1次比较运算求最小值，时间复杂度：

**引用**

[1] csu菜鸟，<https://blog.csdn.net/weixin_43175029/article/details/91386661> [DB/OL]