一笔画问题或欧拉路径

1、问题概述

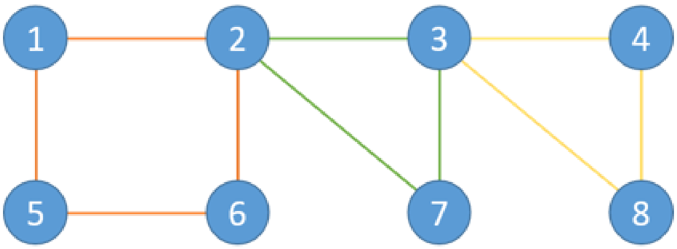
欧拉路是指从图中任意一个点开始到图中任意一个点结束的路径，并且图中每条边通过的且只通过一次。

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

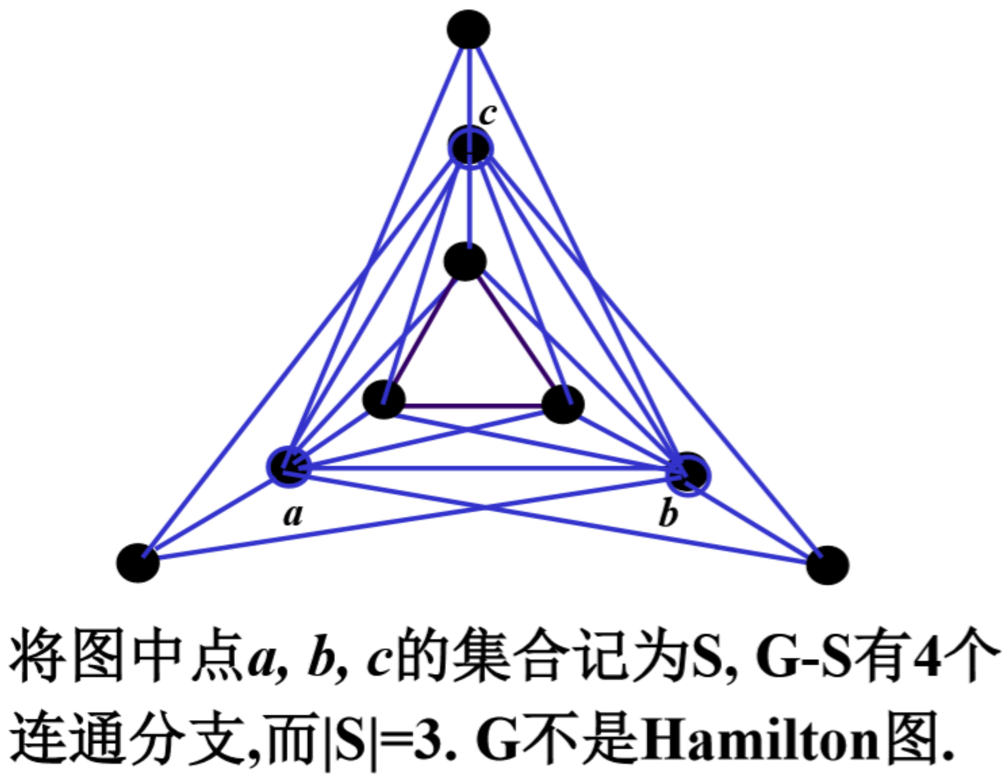
1、问题概述

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

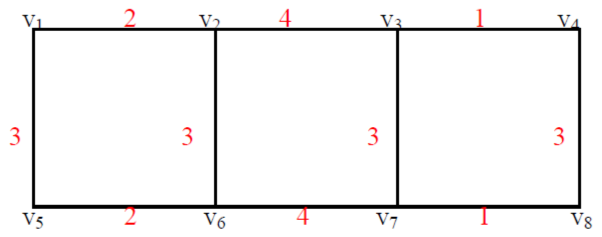
邮递员每天从邮局出发，走遍该地区所有街道再返回邮局，问题是他应如何安排送信的路线可以使所走的总路程最短。

2、求解算法思想

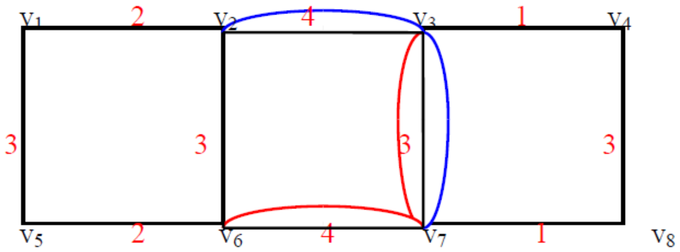
由命题 1，简单图 G 的奇点个数为偶数，可设为 $v_1, v_2, \cdots, v_{2k}$，对每个 $1 \le i \le k$,找 v_{2i-1} 至 v_{2i} 的链 p_i，将 p_i 的边重复一次。对于每一个 p_i 而且除两端点外，其它点保持原奇偶性，即此时图中无奇点。再将添加边多于 1 条的边，成对删去，仍保持点的奇偶性。由命题 2，存在伪 Euler 圈。将添加的边组成一个可行集，由命题 3 检验是否为最优，如果非最优的，则存在一圈不满足命题 3，将该圈中非重复边重复一次，重复边删去一次，图的各点奇偶性不变。

3、举例说明求解过程

例5. 一个邮递员从邮局出发投递信件，然后再返回邮局，如果他必须至少一次地走过他负责投递范围内的每条街道，街道路线如下图所示，问选择怎样的路线才能使所走的路为最短?

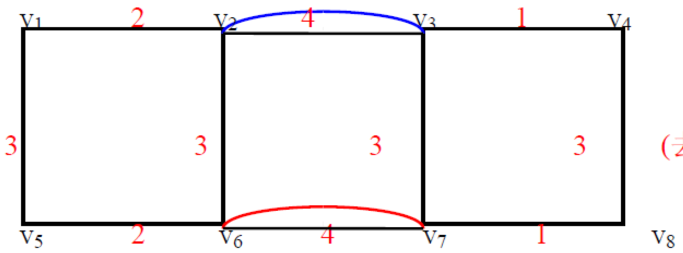


对于例 5，我们有



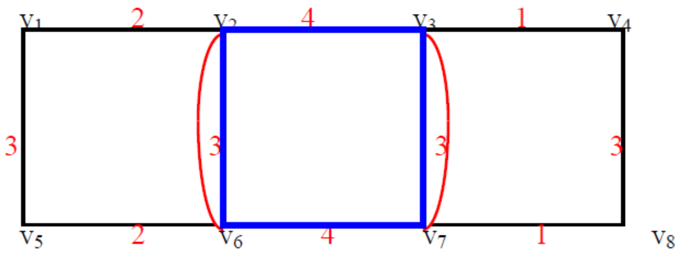
(找奇点、消奇点)

v_3 , v_6 , v_2 ,v_7



(去重边、检查)

圈 v_2 v_3 v_7 v_6 v_2



(调整)

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

1. 找两两最短路
   - Floyd–Warshall算法：$O(v^3)$
2. 构造完全图：$O(v^2)$
3. 找最小权完美匹配
   - Edmonds算法：$O(v^3)$
   - 另有 $O\left(\sqrt{\nu}\varepsilon\right)$ 算法
4. 添加重边：$O(\varepsilon)$
5. 找Euler闭迹
   - Fleury算法：$O(\varepsilon^2)$

旅行推销员问题

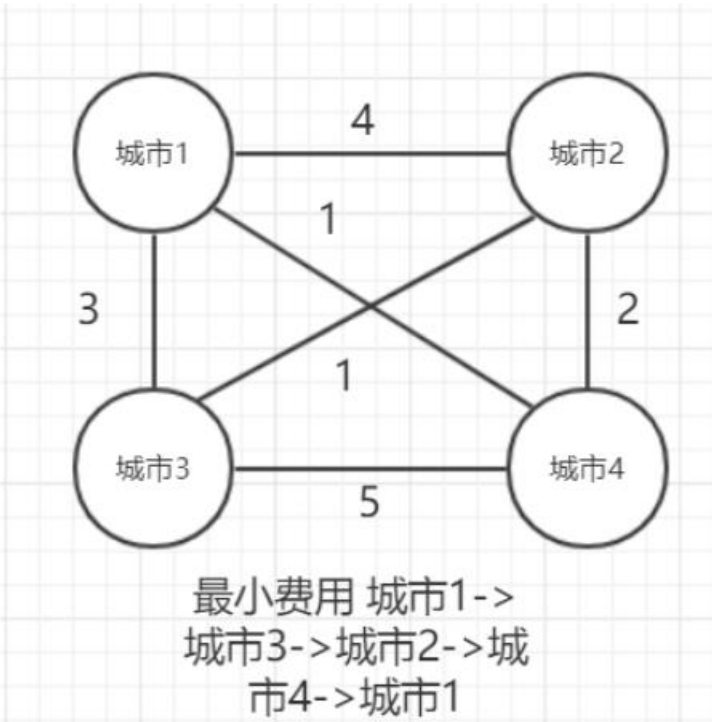
1、问题概述

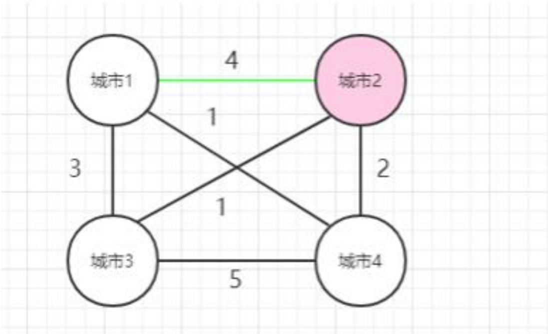
一个商品推销员要去若干个城市推销商品，该推销员从一个城市出发，需要经过所有城市后，回到出发地。应如何选择行进路线，以使总的行程最短。

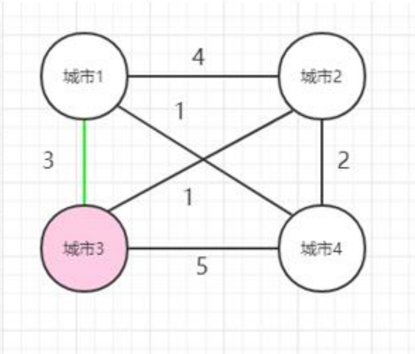
2、求解算法思想

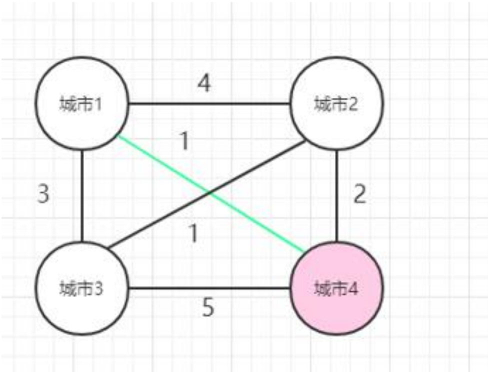
贪心寻找最近的未访问的邻点

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：$O(2^n n^2)$

引用

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