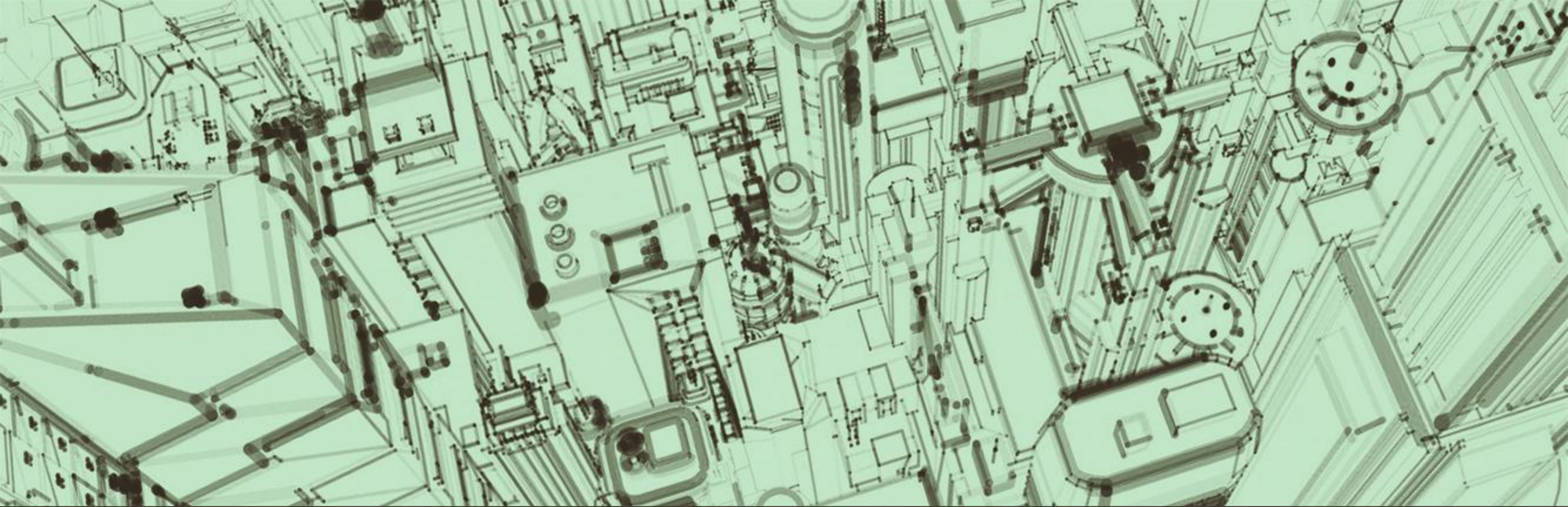


图

欧拉路径

欧拉路问题。有两种欧拉路。第一种叫做 Eulerian path(trail)，沿着这条路径走能够走遍图中每一条边；第二种叫做 Eularian cycle，沿着这条路径走，不仅能走遍图中每一条边，而且起点和终点都是同一个顶点。注意：欧拉路要求每条边只能走一次，但是对顶点经过的次数没有限制。

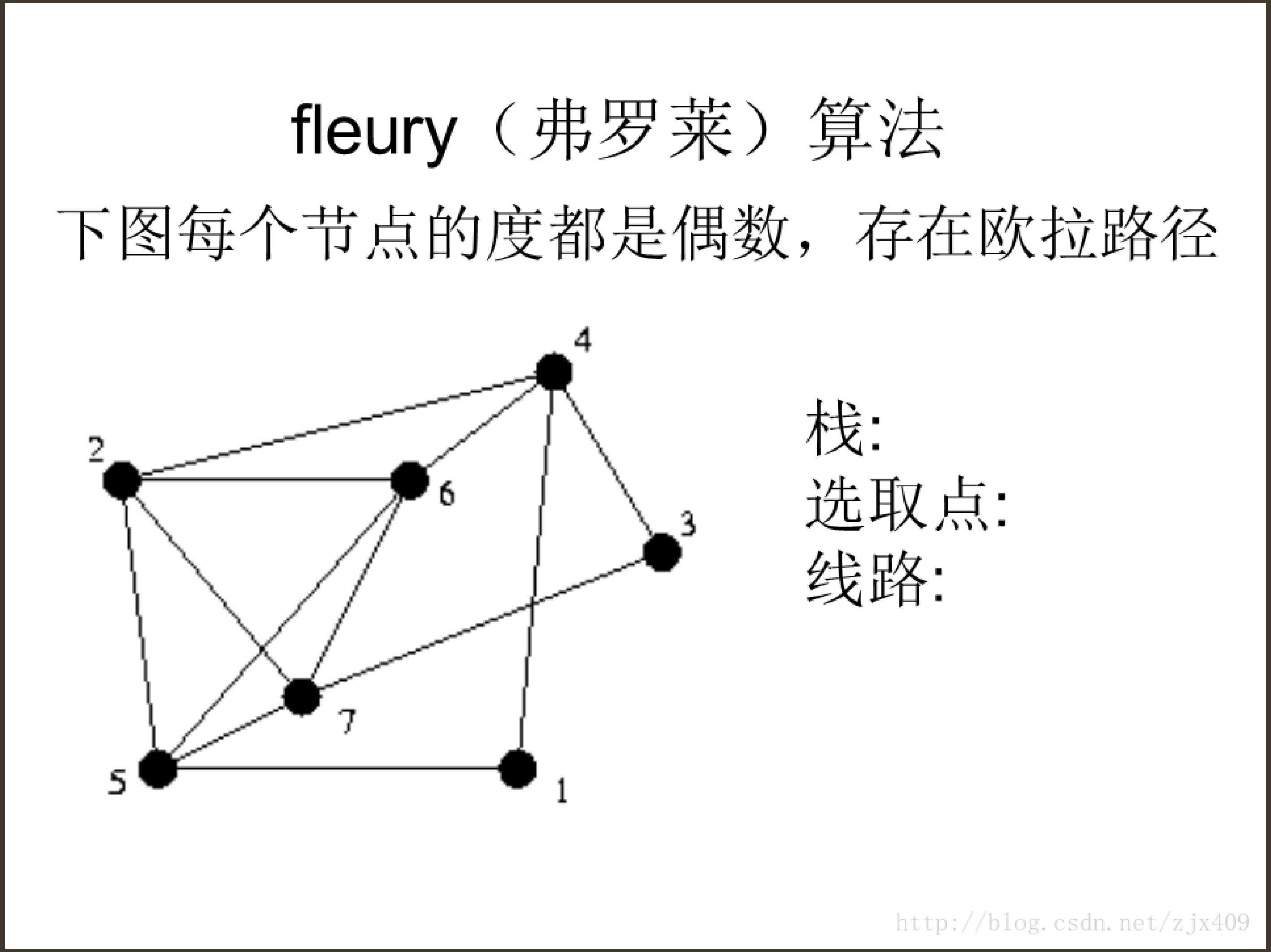
解决算法-- Fleury 算法

Fleury 算法每个回合进行到一个顶点上的时候，都会删除已经走过的边。在选择下一条边的时候，不应该出现这样的状况：在删除下一条边之后，连通图被分割成两个不连通的图。除非没有别的边可选择。该算法从一个奇度数顶点开始（若所有顶点度数均为奇，则任选一个顶点）。当所有的边都走完的时候，该算法结束，欧拉路径为删除路径的顺序。

```java
public void euler(int[][] graph,Stack s,int current,int start){
        int num_edges;  //graph中的边数
        boolean flag = false;  //是否还有与x关联的边
        s.push(current);
        for(int i = start; i < graph.length ; i++){
            //从start开始搜索是否有边
            if(graph[i][x] > 0){ //i与current有边
                flag = true;
                graph[i][current] = 0; graph[current][i] = 0; //删除边
                euler(graph,s,i,0);  //从i开始搜索
                break;
            }
        }

        if(flag){ //如果没有变与当前节点current相连
            s.pop();
            int m = s.peek();
            G[m][current] = G[current][m] = 1; //恢复边
            int new_start = current + 1;
            if(s.size() == num_edges){ //完成回路
                return;
            }else{
                euler(graph,s,m,new_start);
            }
        }
```

例子



- 按照算法的欧拉回路为
- 1→5→7→ 6 → 4 → 3 → 7 → 2 → 6 → 5 → 2 → 4 → 1
- 由于算法要经过每条边，所以时间必然是Ω(E)。在最坏情况下，在每个节点处进行一次 DFS，节点会重复走所以以边计算，所以算法复杂度应该是 O(E(E+V))

哈密尔顿问题

哈密尔顿回路在过各个顶点的巡回线路中，若每个顶点只经过一次，则称为哈密尔顿回路。 哈密顿回路图，与欧拉回路图正好互相呼应，欧拉回路要求通过每条边一次且仅仅一次，而哈密顿回路图则要求通过每个顶点一次且仅仅一次。哈密顿回路图有一个重要的问题：traveling salesperson problem,TSP，就是所谓的 *货郎担* 的问题-->要求在图中发现经过所有顶点且总距离最短的路线。

解决算法-- nearest neighbor algorithm

- ① 从任何节点开始，将其加入到解的集合中
- ② 从与该结点连接的边中选择最短的那条边的结点加入到解的集合中，这就是所谓的最近邻居。若同时有多条边距离相等，　　　　任选一条即可。
- ③ 从上述运算所选的最近邻居出发，重复上述过程，但应避免已选择过的结点，以免形成回路。
-
- ④ 当所有结点都加到解的集合中后，将最后加入的结点与起始结点连接，就可以得到哈密顿回路了。

中国邮递员问题

邮递员从邮局出发送信，要求对辖区内每条街都至少通过一次，再回邮局。在此条件下，怎样选择一条最短路线？

如果街区形成的图本身就是一个欧拉回路，最短路自然是所有街道的总长度。可是如果不是，我们将必须重复一些路径，换句话说，怎样使重复的路径的总长度最短。

解决算法

（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；

旅行推销员问题

TSP问题（Travelling Salesman Problem）即旅行商问题，又译为旅行推销员问题、货郎担问题，是数学领域中著名问题之一。假设有一个旅行商人要拜访n个城市，他必须选择所要走的路径，路径的限制是每个城市只能拜访一次，而且最后要回到原来出发的城市。路径的选择目标是要求得的路径路程为所有路径之中的最小值。

解决算法—动态规划

假设从顶点s出发，令d(i, V’)表示从顶点i出发经过V’(是一个点的集合)中各个顶点一次且仅一次，最后回到出发点s的最短路径长度。

推导：(分情况来讨论)

①当V’为空集，那么d(i, V’)，表示从i不经过任何点就回到s了，如上图的 城市3->城市0(0为起点城市)。此时d(i, V’)=Cis(就是 城市i 到 城市s 的距离)、

②如果V’不为空，那么就是对子问题的最优求解。你必须在V’这个城市集合中，尝试每一个，并求出最优解。

d(i, V’)=min{Cik + d(k, V’-{k})}

注：Cik表示你选择的城市和城市i的距离，d(k, V’-{k})是一个子问题。

综上所述，TSP问题的动态规划方程就出来了：

$$d(i,V')=\begin{cases} c_{is} & V'=\phi, i\neq s \\ \min\limits_{k\in V'}\{c_{ik}+d(k,V'-\{k\})\} & V'\neq\phi \end{cases}$$