Section 3.3 Eulerian Tour 欧拉通路

by 多维数组

例题：Riding the Fences

农夫 John 拥有许多篱笆，他需要定期检查他们是否完整。农夫 John 有一个表，上面记有所有交叉点与各个篱笆的端点，他用这个表来得到它的篱笆的地图。每个篱笆有两个端点，每个端点都在交叉点上。然而有的交叉点可能只与一个篱笆相连。当然，有两个以上的篱笆可能共有同一个端点。现在给你农夫 John 的表，计算是否存在一条路，能使农夫 John 骑马经过他所有的篱笆且每个篱笆只经过一次。农夫 John 可在任何位置出发或结束，但他不能穿越他的农场。请问是否存在这样的一条路径。若存在，请找出它。

抽象模型

现在给你一副无向图。寻找一条包含所有边的路径，其中每一条边只经过一次。这被叫做欧拉通路。若这条路径的起点与终点为同一点，则为欧拉回路。

算法

判定一个图是否存在欧拉通路或欧拉回路比较容易，这里提供两种不同的判定法则。定理 1：一个图有欧拉回路当且仅当它是连通的（即不包括 0 度的结点）且每个结点都有偶数度。定理 2：一个图有欧拉通路当且仅当它是连通的且除两个结点外，其他结点都有偶数度。定理 3：在定理 2 的条件下，含奇数度的两个结点中，一个必为欧拉通路的起点，另一个必为终点。

此算法的基本思想是从图中的某个结点出发求出一个能回到该结点的路径。现在，该路径已被加入（根据它的结果，逆序存放）。检验在该路径中的所有结点的每条边是否已被应用。若某个结点存在未被应用的边，则算法进一步找出一条从此结点的这条边开始的新回路，并将这个新的回路连入原路径中。算法一直运行到最初的路径中的所有结点的每一条边都已被应用。既然改图是连通的，这也就表明图中所有的边都已经被应用，所以最终结果也就是欧拉回路。

更加正式的说，寻找图的欧拉回路的方法是选取一个起点，并对它进行递归，递归的步骤为：

- 选取一个起点并在此结点上，按以下步骤递归：
  - 若此结点无邻结点，则添加该结点到回路中并退出。
  - 若此结点有邻结点，则生成一个邻结点表并对表中的结点进行操作（遍历一个点，删除一个点）直到列表为空。最后将处理的结点加入回路中。
- 对表中结点的操作为：删去当前处理的结点（指的是当前递归下的结点）与表中的结点相连的边，然后对这个结点（指表中的结点，我觉得好难表述，建议直接看伪代码.....）进行递归。

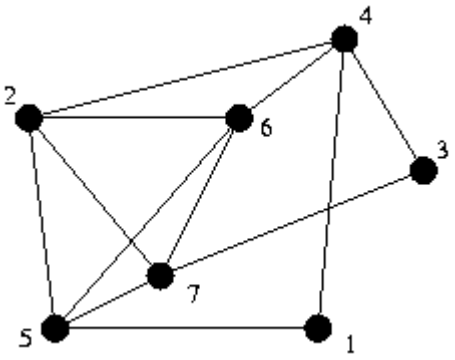
以下是伪代码：

```
# circuit is a global array
   find_euler_circuit
     circuitpos = 0
     find_circuit(node 1)

# nextnode and visited is a local array
# the path will be found in reverse order
  find_circuit(node i)

    if node i has no neighbors then
      circuit(circuitpos) = node i
      circuitpos = circuitpos + 1
    else
      while (node i has neighbors)
          pick a random neighbor node j of node i
          delete_edges (node j, node i)
```

```
        find_circuit (node j)
    circuit(circuitpos) = node i
    circuitpos = circuitpos + 1
```

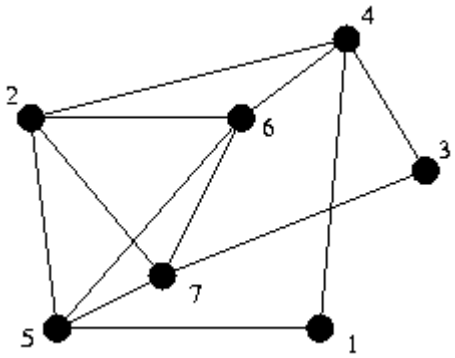
寻找欧拉通路的方法是先找到一个含有奇数度的节点，然后以它为参数调用 find_circuit。如果你用邻接表存储图，这两个算法的时间复杂度均为 O(m+n)，其中 m 为边的个数、n 为结点个数。若图非常的大，则很容易栈溢出。所以你应该自己建栈。

运行实例

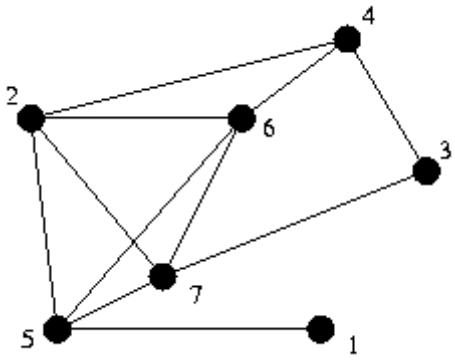


考虑以下的图

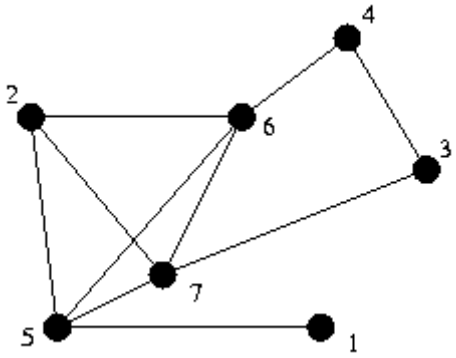
假设先选择编号最小的邻结点：



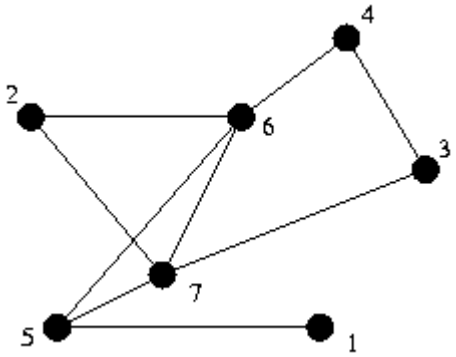
Stack: Location: 1 Circuit:



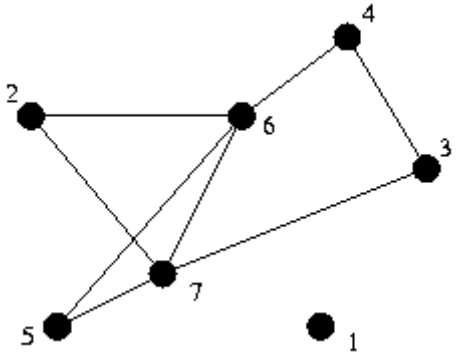
Stack: 1 Location: 4 Circuit:



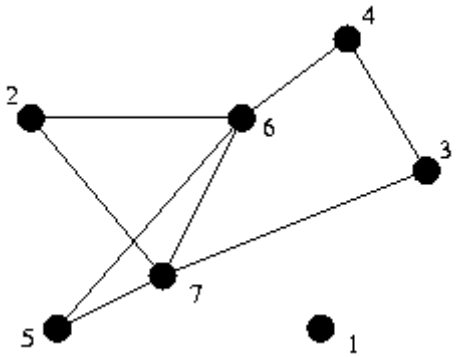
Stack: 1 4 Location: 2 Circuit:



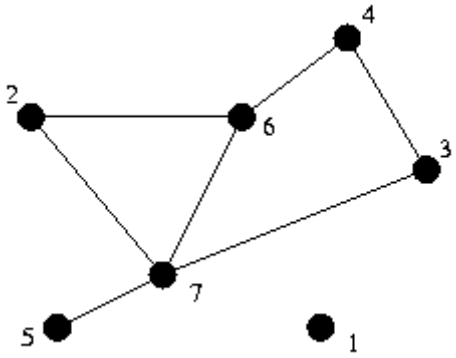
Stack: 1 4 2 Location: 5 Circuit:



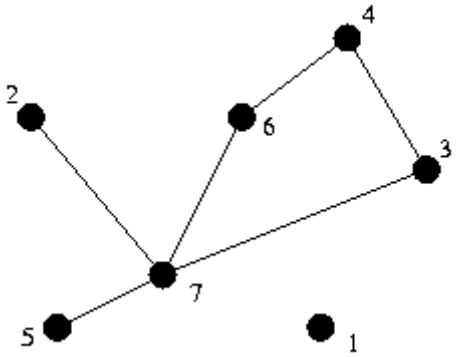
Stack: 1 4 2 5 Location: 1 Circuit:



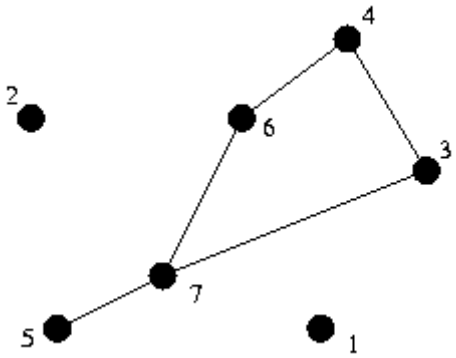
Stack: 1 4 2 Location: 5 Circuit: 1



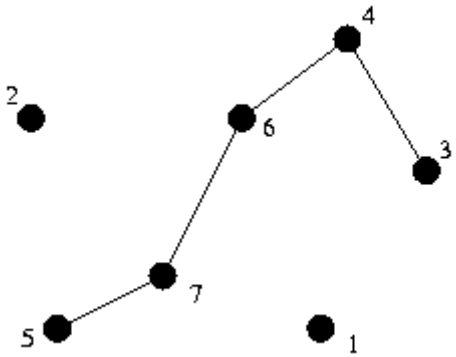
Stack: 1 4 2 5 Location: 6 Circuit: 1



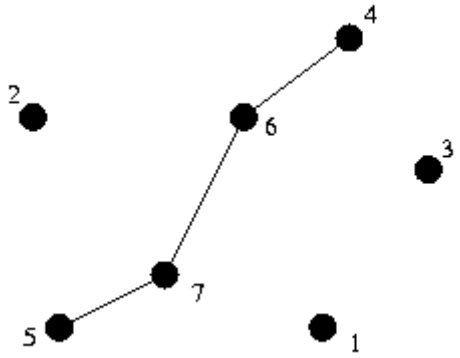
Stack: 1 4 2 5 6 Location: 2 Circuit: 1



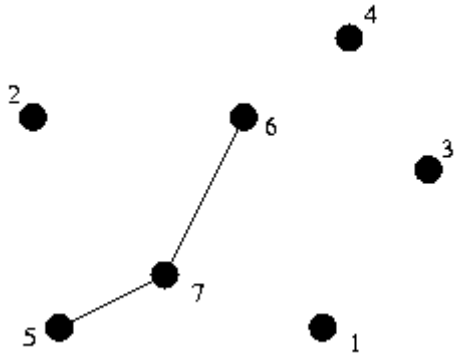
Stack: 1 4 2 5 6 2 Location: 7 Circuit: 1



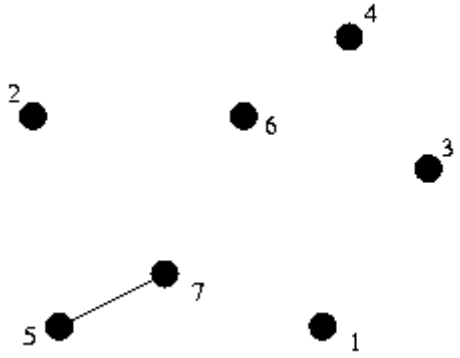
Stack: 1 4 2 5 6 2 7 Location: 3 Circuit: 1



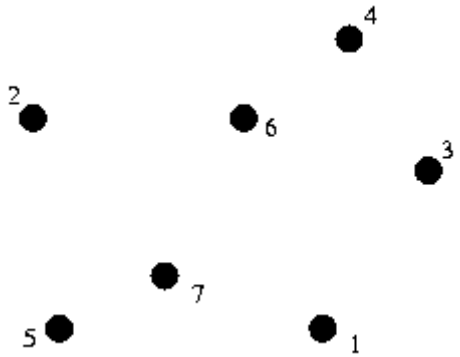
Stack: 1 4 2 5 6 2 7 3 Location: 4 Circuit: 1



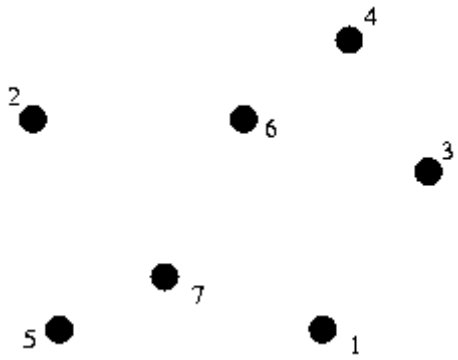
Stack: 1 4 2 5 6 2 7 3 4 Location: 6 Circuit: 1



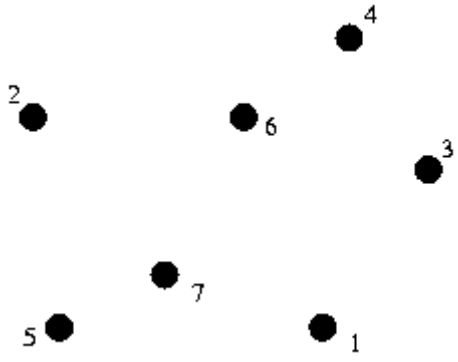
Stack: 1 4 2 5 6 2 7 3 4 6 Location: 7 Circuit: 1



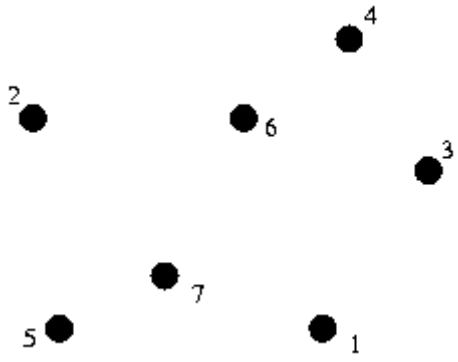
Stack: 1 4 2 5 6 2 7 3 4 6 7 Location: 5 Circuit: 1



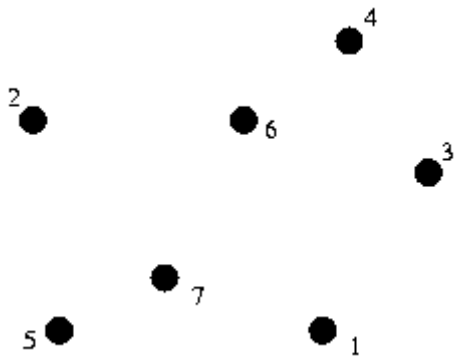
Stack: 1 4 2 5 6 2 7 3 4 6 Location: 7 Circuit: 1 5



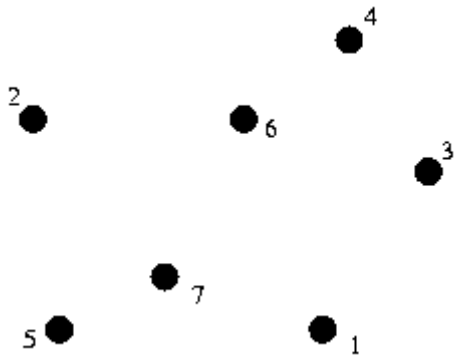
Stack: 1 4 2 5 6 2 7 3 4 Location: 6 Circuit: 1 5 7



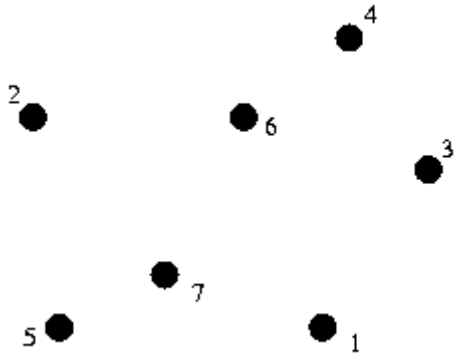
Stack: 1 4 2 5 6 2 7 3 Location: 4 Circuit: 1 5 7 6



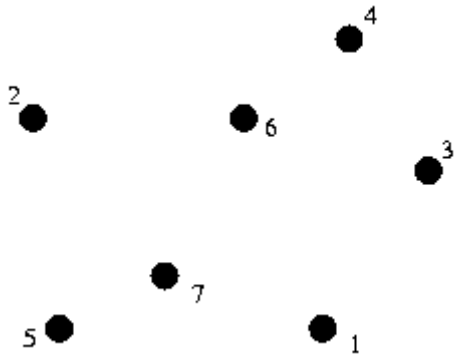
Stack: 1 4 2 5 6 2 7 Location: 3 Circuit: 1 5 7 6 4



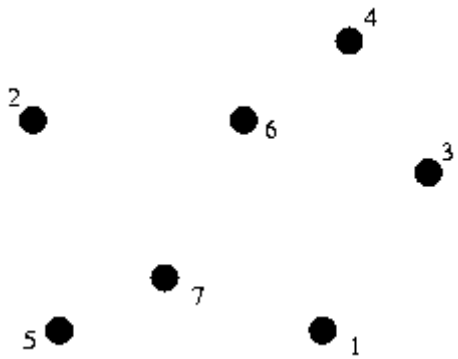
Stack: 1 4 2 5 6 2 Location: 7 Circuit: 1 5 7 6 4 3



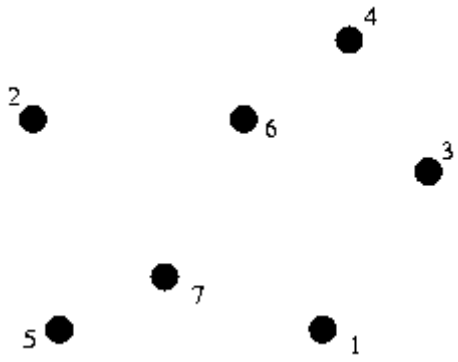
Stack: 1 4 2 5 6 Location: 2 Circuit: 1 5 7 6 4 3 7



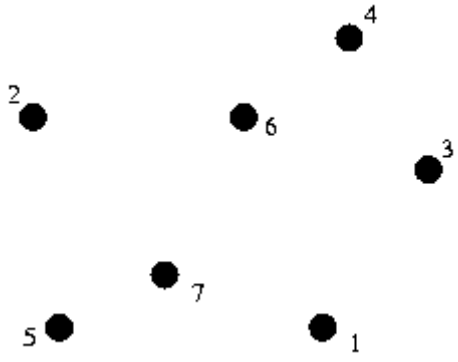
Stack: 1 4 2 5 Location: 6 Circuit: 1 5 7 6 4 3 7 2



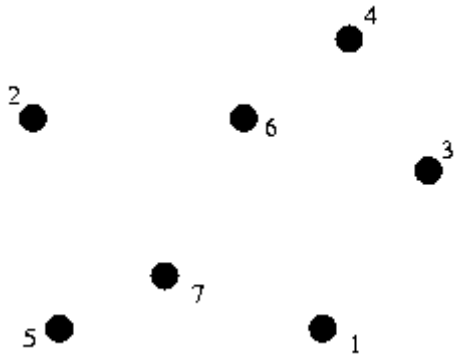
Stack: 1 4 2 Location: 5 Circuit: 1 5 7 6 4 3 7 2 6



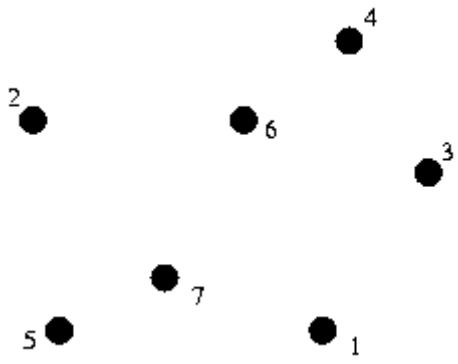
Stack: 1 4 Location: 2 Circuit: 1 5 7 6 4 3 7 2 6 5



Stack: 1 Location: 4 Circuit: 1 5 7 6 4 3 7 2 6 5 2



Stack: Location: 1 Circuit: 1 5 7 6 4 3 7 2 6 5 2 4



Stack: Location: Circuit: 1 5 7 6 4 3 7 2 6 5 2 4 1

扩展知识

结点间的重边能用相同的算法解决。自环也能被同样的算法解决。把它们看成给结点加上 2（一进一出）的度。

当一个有向图是强连通图（除了入度和出度均为 0 的结点），并且每个点的入度和出度相等时，其中存在一条欧拉回路。算法也是一样的，只是由于它是逆序找到的，你需要反向遍历它。在有向图中寻找一条欧拉回路更难一些。如果你感兴趣，可以查阅 Sedgewick [(其实就是《Algorithms in ……》那一系列书)]的相关资料。

练习 Airplane Hopping 给出一系列城市，以及城市间的航班。请找出一条路线能搭乘所有的航班并再次回到出发城市。
解析：这等价于在有向图中寻找欧拉回路。

(以下待翻译) Analysis: This is equivalent to finding a Eulerian circuit in a directed graph. Cows on Parade Farmer John has two types of cows: black Angus and white Jerseys. While marching 19 of their cows to market the other day, John's wife Farmeress Joanne, noticed that all 16 possibilities of four successive black and white cows (e.g., bbbb, bbbw, bbwb, bbww, ..., wwww) were present. Of course, some of the combinations overlapped others.
Given N (2 <= N <= 15), find the minimum length sequence of cows such that every combination of N successive black and white cows occurs in that sequence.
Analysis: The vertices of the graph are the possibilities of N-1 cows. Being at a node corresponds to the last N-1 cows matching the node in color. That is, for N = 4, if the last 3 cows were wbw, then you are at the wbw node. Each node has out-degree of 2, corresponding to adding a black or white cow to the end of the sequence. In addition, each node has in-degree of 2, corresponding to whether the cow just before the last N-1 cows is black or white.
The graph is strongly connected, and the in-degree of each node equals its out-degree, so the graph has a Eulerian circuit.
The sequence corresponding to the Eulerian circuit is the sequence of N-1 cows of the first node in the circuit, followed by cows corresponding to the color of the edge.