文件professional\_motif\_algrithm.py 里面有四阶和五阶模体的计数算法，适用于有向无环图，每个节点只有一个父节点的网络，也就是树型网络。计算速度很快。

算法在networkx 的基础上开发的，使用networkx里面的DiGraph作为数据结构来存储网络。

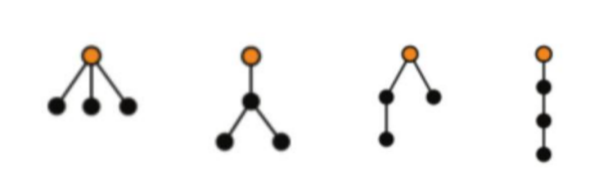
使用时直接调用相应的函数，参数是网络。例如：

data = pandas.read\_csv("data/politifact279.txt", sep=" ", names=["a", "b", "w"])  
G = networkx.from\_pandas\_edgelist(data, source="a", target="b", create\_using=nx.DiGraph())  
cal\_motif5\_1(G)

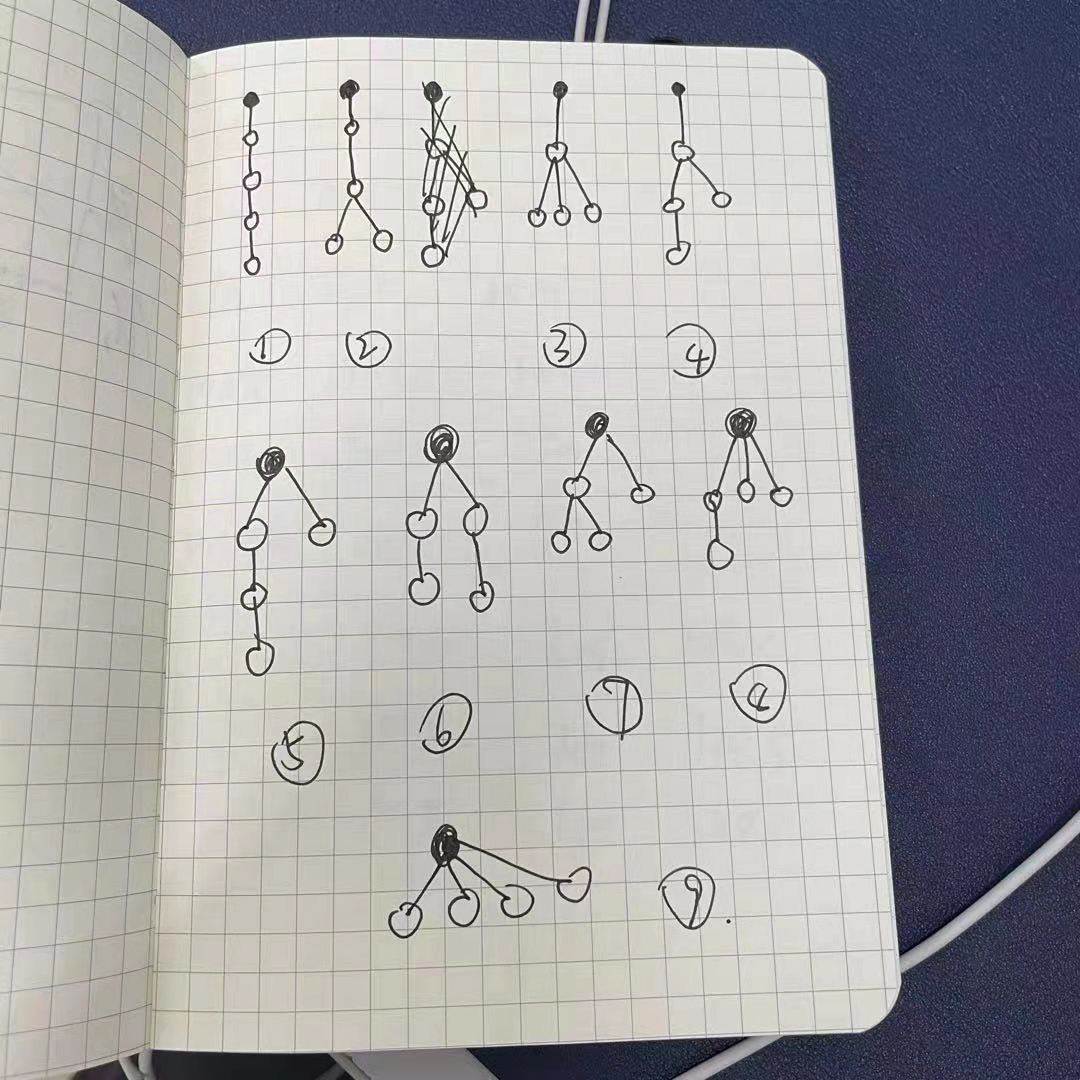
其中前两句代码是读取文件中的网络并存储在G中

cal\_motif5\_1(G) 这句代码是计算G中的M5模体的数量。

四阶模体，从左到右依次是M1—M4



五阶模体，从左到右从上到下依次是M1—M9



文件motifNum4.py和motifNum3.py里面有模体的通用算法，可以计算各阶模体在有向、无向、加权、无权、符号等网络的点模体、边模体、模体总数。

算法在networkx 的基础上开发的，使用networkx里面的Graph和DiGraph作为数据结构来存储网络。

使用时只需要调用这三个函数即可

node\_motif\_num(G,G\_motif,node,directed=False,weighted=False)

edge\_motif\_num(G,G\_motif,edge,directed=False,weighted=False)

total\_motif\_num(G,G\_motif,directed=False,weighted=False)

其中G为网络，G\_motif为模体，node 是G中的节点，edge，是G中的边也是节点对（node1,node2）。Directed为是否为有向网络默认为否，weighted为是否为加权网络默认为否。

注意：

1. 点模体算法计算的是网络G中的节点node作为模体中的第一个节点参与构成模体的数量。

2. 边模体算法计算的是网络G中的边edge作为模体中的第一条边参与构成模体的数量。

3. 若要计算作为其他节点或者边，请控制模体建立时的输入顺序和序号大小。

例如：

创建一个模体M1

M1 = nx.DiGraph()  
M1.add\_nodes\_from([1, 2, 3, 4])  
M1.add\_edges\_from([(1, 2), (2, 3), (1, 3), (3,4)])

调用：

node\_motif\_num(G,M1,directed=Ture,weighted=False)