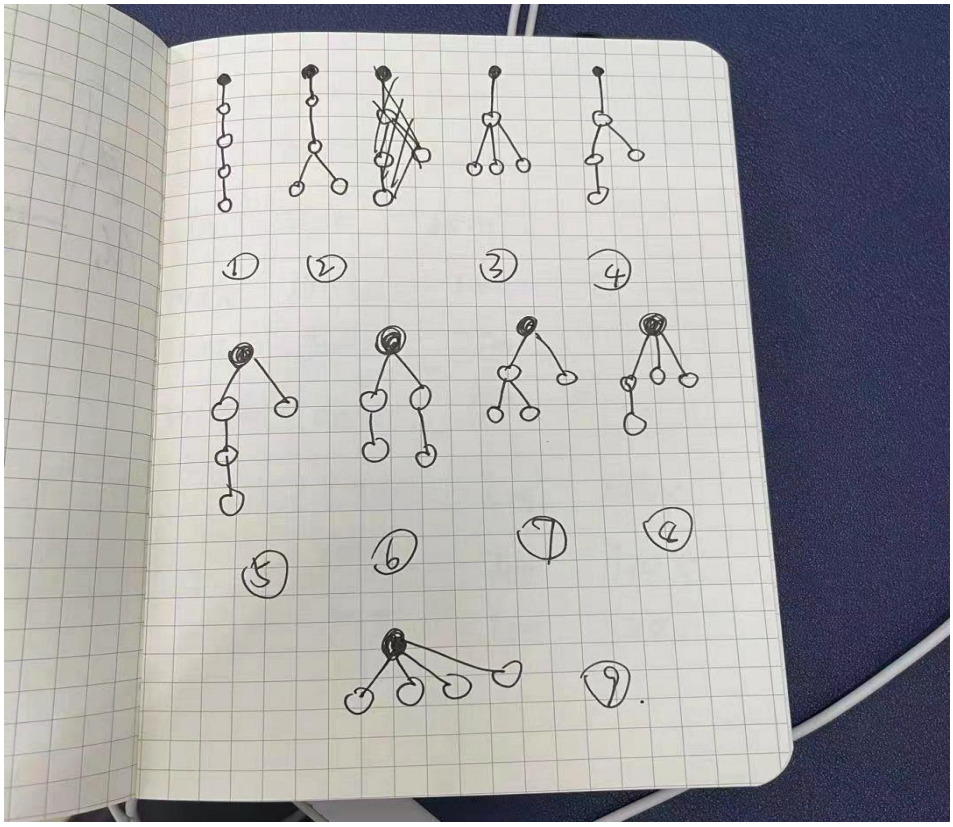
文件 professional_motif_algrithm.py 里面有四阶和五阶模体的计数算法，适用于有向无环图，每个节点只有一个父节点的网络，也就是树型网络。计算速度很快。
算法在 networkx 的基础上开发的，使用 networkx 里面的 DiGraph 作为数据结构来存储网络。
使用时直接调用相应的函数，参数是网络。例如：

```
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)