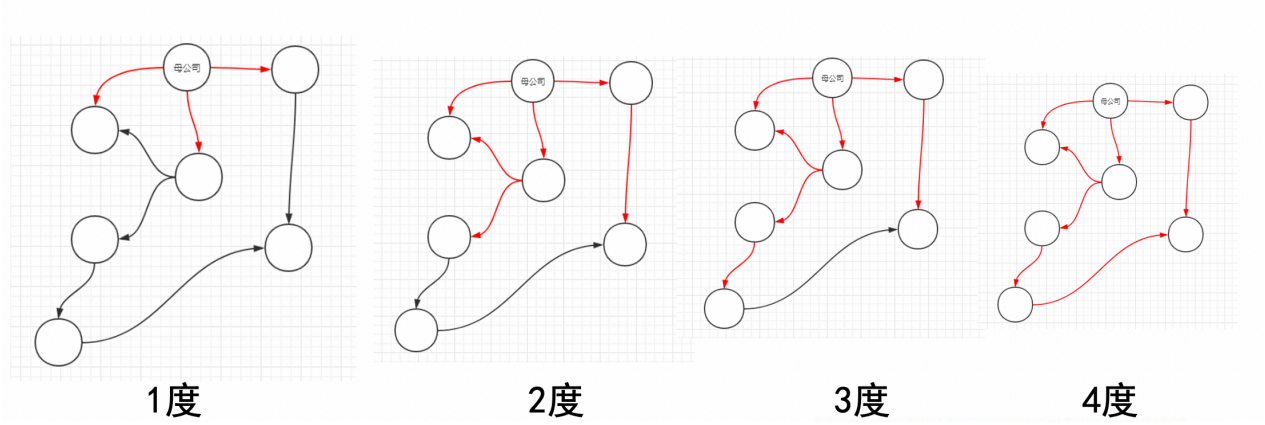
背景

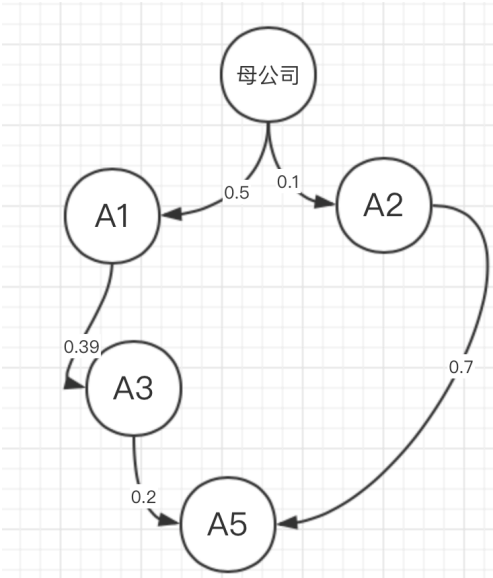
现有如下图格式的数据，其为各公司之间的投资关系及持股比例。要求给定一个母公司（即输入），输出满足如下三个规则的子公司及控股路线。

集团名称	股东名称	被投资公司名称	营业状态	持股比例
母1	A1	B1	在营（开业）	0.6
母1	A1	B2	在营（开业）	0.7
母1	A2	B3	在营（开业）	0.01
母2	A3	B4	在营（开业）	0.3

规则一：从母公司出发，沿着51%以上控股路径 n度关联到的公司；

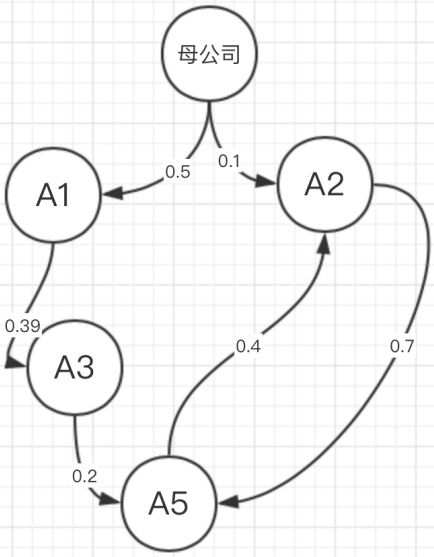


规则二：从母公司出发，沿着控股路径 n度关联到的公司中，不被规则一命中的公司，间接控股比例（公式一、二）超过51%；



母公司对A5的持股比例为：

0.5×0.39×0.2+0.1×0.7

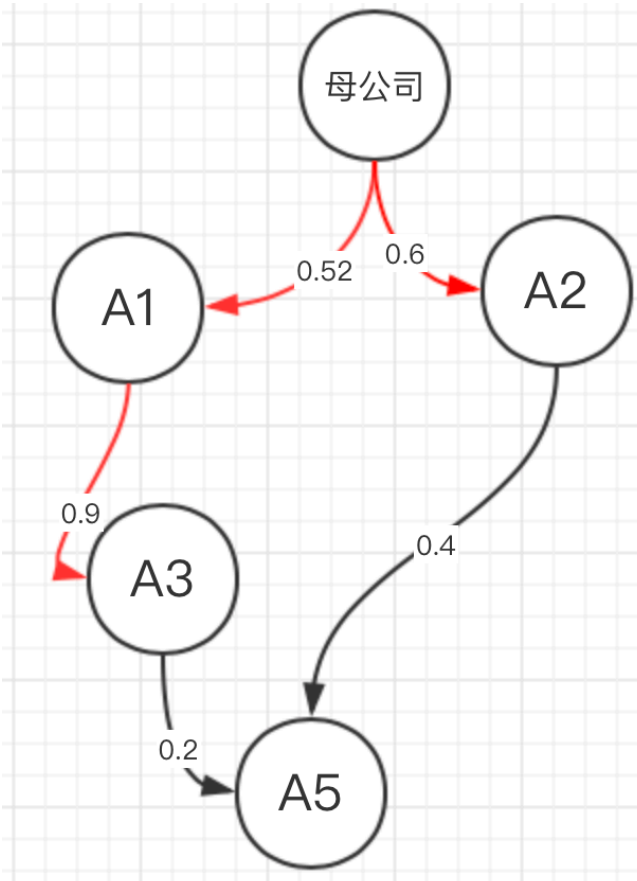


A2与A5存在交叉持股情况，修正A2到A5的控股比例为：
0.1×0.7/（1−0.4×0.7）=0.097

则母公司对A5的持股比例为：

0.5×0.39×0.2+0.1×0.097

规则三：从母公司出发，沿着控股路径 n度关联到的公司中，不被规则一二命中但上级公司被规则一命中的公司，间接控股比例（被规则一命中的上级公司对该公司的持股比例之和）超过51%。



母公司对A5的持股比例为：

0.2+0.4

图结构

Graph gqyj(gqyj_v_company_node:v, gqyj_e_share_relation:e, gqyj_e_share_relation_by:e)

VERTEX gqyj_v_company_node

attribute	type	解释
id	STRING	PRIMARY_ID
cust_name	STRING	公司名称
unn_soc_cr_cd	STRING	统一社会信用代码
entp_inf_idr	STRING	
status	STRING	经营状态
data_version	DATETIME	数据版本

DIRECTED EDGE gqyj_e_share_relation/gqyj_e_share_relation_by

FROM gqyj_v_company_node, TO gqyj_v_company_node

attribute	type	解释
fndd_pctg_desc	DOUBLE	持股比例
data_version	DATETIME	数据版本

实现逻辑

1. 标记母公司n度内可到达的子公司；
2. 找出交叉持股并在顶点上保存反向持股比例；
3. 找出母公司到每个子公司的全部路线及持股比例；
4. 规则一判断；
5. 规则二判断；
6. 规则三判断；
7. 结果输出文件。

下面对实现代码进行解析，同时请补充其中的练习一、第五步、第六步。

代码解析

第一步，我们首先将母公司可以到达的公司即子公司全部找出来。这里我们将采用一个循环并将每次循环找到的公司标记为true，可以看出每次循环的输出是下一次循环的输入：

遍历次数	输入start	输出start
1	母公司	母公司直接控股的第一层子公司
2	第一层子公司	第一层子公司直接控股的第二层子公司
…	…	…

```
create query qgyj_rule_v5 (vertex<gqyj_v_company_node> target_v,int step,file
f,bool vis_flag=false,datetime datatime)
FOR GRAPH gqyj returns(INT){
  /*
  输入：target_v：母公司
       step：计算到指定层级
       vis_flag：是否可视化展示
       datatime：数据的版本
  */
  OrAccum @visited;

  start = {target_v};
  all_v = {};  # 这里存储所有子公司，后面的步骤将会用到
  all_v = all_v union start;
  foreach sub_step in range[1, step] do
    start = select t
            from start:s-(gqyj_e_share_relation:e)->:t
            where trim(t.status) == "在营（开业）"
              and t.data_version == datatime
              and e.data_version == datatime
            post-accum t.@visited = true;
    all_v = all_v union start;
    print sub_step, all_v.size();
  end;
}
```

第二步：这一步我们将找出交叉持股的节点，并保留它的反向控股比例（交叉持股：A公司投资B公司，B公司投资A公司）：

```
# 这里我们以map的形式把交叉持股的反向持股比例存储到该节点
MapAccum<string,double> @circle_share;

# 找出交叉持股节点与股份

start = select s
```

```
        from all_v:s-((gqyj_e_share_relation):e)->:t  #all_v为第一步中的计算结果
        where trim(t.status) == "在营（开业）"
          and t.data_version == datatime
          and e.data_version == datatime
          and t.@visited == true
          and t!=s  # 自持股并不是交叉持股，去除
          and t in s.neighbors("gqyj_e_share_relation_by")
        accum s.@circle_share += (t.cust_name->e.fndd_pctg_desc);
```

第三步：这一步我们将找出母公司到每个子公司的全部路线，每条路线都将存放在子公司（即这条路线的终点）的@finished_share（一个MapAccum）中。这一步需做出如下判断：

①防止环的生成（if not contains_string(k,t.cust_name)）；

②防止路线已存在（if not t.@finished_share.containsKey(tg)）；

③找出交叉持股（if not t.@circle_share.containsKey(s.cust_name)）。

```
  start = select t from start:t POST-ACCUM t.@send_share +=(t.cust_name->1);
  all_v0 = all_v;
  @@update_vertex = 999;
  while @@update_vertex>0 do
  @@update_vertex=0;
   all_v0 = select t from all_v0:s-(gqyj_e_share_relation:e)->:t
           where trim(t.status) == "在营（开业）" and t.data_version == datatime
and
            e.data_version == datatime
            and t.@visited == true
            and e.fndd_pctg_desc> (0 + epsilon)
            and e.fndd_pctg_desc< (1 + epsilon)

           accum
              foreach (k,v) in s.@send_share do
                  if not contains_string(k,t.cust_name) then
                          string  tg =k+"->"+to_string(e.fndd_pctg_desc)+"-
>"+t.cust_name,
                      if not t.@finished_share.containsKey(tg) then
                        if not t.@circle_share.containsKey(s.cust_name) then
                            t.@finished_share +=(tg-> v*e.fndd_pctg_desc),
                            t.@recive_share += (tg-> v*e.fndd_pctg_desc)
                        else
                          ①
                        end
                      END
                 end
              end
            POST-ACCUM
                @@update_vertex += t.@recive_share.size() ,
```

```
                t.@send_share = t.@recive_share,
                t.@recive_share.clear();

  end;
```

练习一：请依据规则二的图片中给出的交叉持股计算方式，补充第三步的空缺部分①。

第四步：找出母公司到子公司的每一步控股比例均>0.51的路线，并对这些公司标记其为规则一命中的公司，将结果写入一个集合：

```
OrAccum @is_control_rule1;
SetAccum<vertex> @@is_controled_company_rule1;
double epsilon=0.00000001;

start = {target_v};
# role1
foreach sub_step in range[1, step] do
  L1 = select t
       from start:s-(gqyj_e_share_relation:e)->:t
       where e.fndd_pctg_desc > (0.51-epsilon)
          e.fndd_pctg_desc < (1+epsilon)
          and t.data_version == datatime
          and e.data_version == datatime
          and trim(t.status) == "在营（开业）"
          and t.@is_control_rule1 != TRUE
          and t.@visited == true
       post-accum t.@is_control_rule1 = TRUE,
                  @@is_controled_company_rule1 += t;
  if L1.size()==0 then
      break;
  end;
  start = L1;
end;
```

第五步：请补充对规则二的判断；

```

```

第六步：请补充对规则三的判断；

```

```

第七步：结果输出文件；

```
# 输出
#规则号--> 集团名称      最大股东名称 统一社会信用代码   被投资公司名称 --> 被投资公司信用代码    出资比例   层级   经营状态
result = select t from all_v :s-(gqyj_e_share_relation:e)->:t
          where  t.data_version == datatime and e.data_version == datatime
         accum t.@item_tuple+=item(s,e.fndd_pctg_desc,s.unn_soc_cr_cd)
         post-accum
                foreach (path,per) in t.@finished_share DO
                       t.@max_level += count_string(path,"->")/2 +1 ,
                    t.@level_list += count_string(path,"->")/2 +1
                end,
                 if  t.@is_control_rule1 == true then
                 f.println("rule1",target_v,
                   t.@item_tuple.top().up_company,
                   t.@item_tuple.top().unn ,
                   t,
                   t.unn_soc_cr_cd,
                   t.@rule2_sum,
                   t.@max_level,
                   t.status,t.@level_list),

                 @@is_controledd_company_num +=1
                else if t.@is_control_rule2 == true then
                 f.println("rule2",target_v,
                   t.@item_tuple.top().up_company,
                   t.@item_tuple.top().unn ,
                   t,
                   t.unn_soc_cr_cd,
                   t.@rule2_sum,
                   t.@max_level,
                   t.status,t.@level_list),
                 @@is_controledd_company_num +=1
                    else if t.@is_control_rule3 == true then
                 f.println("rule3",target_v,
                   t.@item_tuple.top().up_company,
                   t.@item_tuple.top().unn ,
                   t,
                   t.unn_soc_cr_cd,
                   t.@rule2_sum,
                   t.@max_level,
                   t.status,t.@level_list),
                 @@is_controledd_company_num +=1
                    end;
```

参考答案

练习一

```
t.@finished_share +=(tg-> v*e.fndd_pctg_desc/(1-
e.fndd_pctg_desc*t.@circle_share.get(s.cust_name))),
                            t.@recive_share += (tg-> v*e.fndd_pctg_desc/(1-
e.fndd_pctg_desc*t.@circle_share.get(s.cust_name)))
```

第五步

```
# role2
L2 = select t from all_v:t
     where t!= target_v
        and  trim(t.status) == "在营（开业）"
        and t.data_version == datatime
     accum
        foreach (k,v) in t.@finished_share do
           t.@rule2_sum += v
        end
     post-accum
        if t.@rule2_sum >(0.51-epsilon) and t.@is_control_rule1 != TRUE then
           t.@is_control_rule2 = true,
           @@is_controled_company_rule2 += t
        end;
```

第六步

```
# rule3
L3 = select s
     from all_v:s-(gqyj_e_share_relation_by:e)->:t
     where s.@is_control_rule1 != true
      and s.@is_control_rule2 != true
      and t.@is_control_rule1 == true
      and  trim(t.status) == "在营（开业）"
      and t.data_version == datatime
      and e.data_version == datatime
     accum s.@rule3_v_sum += e.fndd_pctg_desc
     having s.@rule3_v_sum >(0.51-epsilon) ;

L3 = select t from L3:t
    post-accum t.@is_control_rule3 =true ,
               @@is_controled_company_rule3 +=t;
```