面向对象设计与构造2022第三单元总结

零、任务简介

本次作业，需要完成的任务为实现简单社交关系的模拟和查询，学习目标为 **入门级JML规格理解与代码实现。**

本单元作业基本主干已有模板，故而描述也变得谜语人起来，第一次作业会很摸不到头脑。

示例输入：

```
ap 1 jack 100
ap 2 mark 100
ar 1 2 100
qv 1 2
qbs
qps
qci 1 2
```

示例输出：

```
ok
ok
ok
100
1
2
1
```

一、Task1

1) 任务说明

设计指令如下

```
add_person id(int) name(String) age(int)
add_relation id(int) id(int) value(int)
query_value id(int) id(int)
query_people_sum
query_circle id(int) id(int)
query_block_sum
add_group id(int)
add_to_group id(int) id(int)
del_from_group id(int) id(int)
```

2) 整体结构

第一次作业的整体架构如下。简洁考虑，隐去了异常处理类。

想来第一次作业，同学们的架构都很接近吧，主要叙述交互逻辑方面。

根据性能需求，需要最差时间复杂度期望为O(n)，故而本人搭建的复杂度列表为

指令	最差时间复杂度
ap	O(1)
ar	O(n)
qv	O(1)
qps	O(1)
qci	O(1)
qbs	O(1)
ag	O(1)
atg	O(1)
dfg	O(1)

qbs

大部分均采用并查集算法，并使用了路径压缩进行了算法优化。

但是我在此处并没有选择这种方式，而是选择了标记联通块，在ar时，视情况合并两个联通块。则可以使用 O(1) 复杂度解决问题。

这样的代价是ar的最差复杂度为 O(n) 。但是依据本人此前的原则，最差复杂度反而从 O(nlogn) 降低至 O(n)。虽然并查集路径压缩会使得后续复杂度远低于 O(nlogn)，但是从单次查询角度看，确实更慢。

而且，对于所有询问类指令，都可以实现 O(1) 查询。从设计理念上，更贴近实际应用。

3) 互测体验

强测AC

Hack零次。

我房的人都tql，一个都刀不中qwq，全员成功率0%

4) 性能优化

前文已述

5) 额外完成任务

- **qgvs**
- **qgav**

二、Task2

1) 任务说明

- 添加**最小生成树。**有关查询指令
- 添加message信息，以及相关操作

2) 整体结构

与Task1相同部分不再赘述。

自初新建了message类作为信息的存贮载体。Mypair类是一个中间类，表示二元组，方便coding而设置。

对于以下几种方法，有不同的实现策略

qgvs

1. 在atg、dfg、ar时维护组内value总和
2. 遍历组内点集的连通块
3. 遍历边集

qgav

均维护了年龄总和，但在最后有所不同

1. 维护ageSum的同时，维护ageVar
2. 每次qgav时遍历组内成员，计算ageVar

qlc

1. 使用破圈算法，在ar时动态维护最小生成树
2. 使用克鲁斯卡尔算法，临时计算最小生成树

由于我在上次作业中，确定了以ar为核心操作，所有询问操作均为O(1)的基本策略。所以在qgvs和qlc中，均选择了方法1

注意到最小生成树为n-1边，故破圈法的复杂度为O(v) = O(n)

3) 互测体验

强测AC

Hack两次，被hack一次。

Hack主要的原因为在qgvs中，有同学选择的遍历两遍点集，复杂度为O(n2)。但实际上只需要遍历边集，复杂度为O(v)。故而构造了一组数据，卡了超时，刀中两人。

被Hack是因为，在维护最小生成树时，未考虑全部边权为0的情况，我默认边权均为正，被一位同学的评测机刀中。

4) 性能优化

前文已述

5) 额外完成任务

无

三、Task3

1) 任务说明

- 对message进行了细分，并加入相关指令

2) 整体结构

与Task2相同部分不再赘述。

第三次作业的难度相比前几次并没有很大区别，添加的类比较友好（除了jml比较谜语人）

3) 互测体验

强测AC

Hack八次，被hack零次。

很好奇，为什么这次作业这么多人被刀。可能是jml太谜语人了。

这次互测当天比较忙，就评测机随缘发刀了，刀中很多也不知道为什么。似乎是因为这次中测强测都很弱，同房的人从巨佬变成了佬。我甚至刀中一个把id当金额的人，他竟然活过来中测强测，难以置信。

4) 性能优化

本次似乎想过需要迪杰斯特拉+堆优化，但这个算法并不难，且网上大量模板。除此之外，没有可以优化的点。

5) 额外完成任务

无额外目标

四、互测

笔者在互测中，使用了测评机和对拍机。

使用对拍的方式进行测试。未使用Junit（真的不好用，又不能造数据）

完全随机

采用完全随机的策略，通过扩大测试次数以及单次测试的指令规模发现Bug

单元随机

可动态调整各指令出现的概率和指令中id生成的概率等，集中测试相应的功能模块

评测机暴力Hack，大概率是Hack不下来的。假如这位同学有评测机，那么大概率在A房，地毯式轰炸并不一定能保证效果。对于高手互博，往往功能本身不会出错，只有想怪数据，反常识数据，才可克敌制胜。

既然如此，看代码就是必不可少的。和刀我的同学交流，发现他也是通过看代码才刀到。

据我不完全观察，防御力较低的代码遵循以下规律。

- 优化多
- 父子类之间很少调用方法
- 重复代码过多

五、Network扩展

```
/*@ public normal_behavior
      @ requires (\exists int i; 0 <= i && i < people.length;
people[i].euqals(getPerson(procuderId)) && people[i] instanceof Producer);
       @ requires (\exists int i; 0 <= i && i < people.length;
people[i].equals(getPerson(procuderId)) && people[i] instanceof Advertiser);
      @ assignable getPerson(producerId).advertisers;
      @ assignable getPerson(AdvertiserId).producers;
      @ ensures getPerson(producerId).advertisers.length ==
\old(getPerson(producerId).advertisers.length) + 1;
      @ ensures (\exists int i;0 <= i <
getPerson(producerId).advertisers.length;getPerson(producerId).advertisers[i].eq
uals(getPerson(AddVertiserId)));
      @ ensures (\forall int i; 0 <= i && i <
\old(getPerson(producerId).advertisers.length;
      @             \exists int j;0 <= j <
getPerson(producerId).advertisers.length;
      @
\old(getPerson(producerId).advertisers[i]).euqals(getPerson(producerId).advertis
ers[j]));
      @ ensures (\exists int i;0 <= i <
getPerson(advertiserId).producers.length;getPerson(advertiserId).producers[i].eq
uals(getPerson(addVertiserId)));
      @ ensures (\forall int i; 0 <= i && i <
\old(getPerson(advertiserId).producers.length;
      @             \exists int j;0 <= j <
getPerson(producerId).producers.length;
      @
\old(getPerson(advertiserId).producers[i]).euqals(getPerson(advertiserId).produc
ers[k]));
      @ also
      @ public exceptional_behavior
      @ signals (PersonIdNotFoundException e) !(\exists int i; 0 <= i && i <
people.length;
      @
people[i].equals(getPerson(producerId)));
      @ signals (PersonIdNotFoundException e) (\exists int i; 0 <= i && i <
people.length;people[i].equals(getPerson(producerId)))
```

```
      @ && !(\exists int i; 0 <= i && i < people.length;
people[i].equals(getPerson(advertiserId)));
      @*/
void addAdvertise(int producerId,int advertiserId);


/*@ public normal_behavior
      @ requires (\exists int i; 0 <= i && i < people.length;
people[i].euqals(getPerson(customerId)) && people[i] instanceof Customer);
       @ requires (\exists int i; 0 <= i && i < people.length;
people[i].equals(getPerson(AdvertiserId)) && people[i] instanceof Advertiser);
      @ assignable getPerson(customerId).attentions;
      @ ensures getPerson(customerId).attentions.length ==
\old(getPerson(customerId).attentions.length) + 1;
      @ ensures (\exists int i;0 <= i <
getPerson(customerId).attentions.length;getPerson(customerId).attentions[i].equa
ls(getPerson(AdvertiserId)));
      @ ensures (\forall int i; 0 <= i && i <
\old(getPerson(customerId).attentions.length;
      @             \exists int j;0 <= j <
getPerson(customerId).attentions.length;
      @ also
      @ public exceptional_behavior
      @ signals (PersonIdNotFoundException e) !(\exists int i; 0 <= i && i <
people.length;
      @
people[i].equals(getPerson(customerId)));
      @ signals (PersonIdNotFoundException e) (\exists int i; 0 <= i && i <
people.length;people[i].equals(getPerson(customerId)))
      @ && !(\exists int i; 0 <= i && i < people.length;
people[i].equals(getPerson(AdvertiserId)));
      @*/
void addAttention(int customerId,int AdvertiserId);


/*@ public normal_behavior
      @ requires (\exists int i; 0 <= i && i < people.length;
people[i].euqals(getPerson(customerId)) && people[i] instanceof Customer);
      @ requires (\exists int i; 0 <= i && i < people.length;
people[i].equals(getPerson(AdvertiserId)) && people[i] instanceof producerId);
      @ requires (\exists int i; 0 <= i && i <
getPerson(customerId).attentions.length; (\exists int j;0 <= j && j <
getPerson(customerId).attentions.length;getPerson(customerId).attentions[j].equa
ls(getPerson(producerId)));
      @ assignable getPerson(customerId).money;
      @ assignable getPerson(producerId).saleVolume;
      @ ensures getPerson(customerId).money ==
\old(getPerson(customerId).attentions.length) -
getPerson(producerId).getProce();
      @ ensures (\exists int i;0 <= i <
getPerson(customerId).attentions.length;getPerson(customerId).attentions[i].equa
ls(getPerson(AdvertiserId)));
      @ ensures getPerson(customerId).volume ==
\old(getPerson(customerId).volume) + 1;
      @ also
```

```
      @ public exceptional_behavior
      @ signals (PersonIdNotFoundException e) !(\exists int i; 0 <= i && i <
people.length;
      @
people[i].equals(getPerson(customerId)));
      @ signals (PersonIdNotFoundException e) (\exists int i; 0 <= i && i <
people.length;people[i].equals(getPerson(customerId)))
      @ && !(\exists int i; 0 <= i && i < people.length;
people[i].equals(getPerson(producerId)));
      @*/
void purchase(int customerId,int producerId);
```

六、心得感想

~~（坐大牢）~~

第三单元的OO课程，比第二单元轻松不少，但还是难度不小。我学到了许多面向对象编程的方法和思维，更加熟练地掌握了Java语言。

当然，这一单元的训练也让我认识到了自己的许多不足，比如思路不清晰，在 Task1 到 Task2 中血泪重构。希望下一单元，可以吸取教训，更上一层楼。