**目录**

[1、需求分析 1](#_Toc56214332)

[**1.1问题描述** 1](#_Toc56214333)

[**1.1.1基本功能** 1](#_Toc56214334)

[**1.1.2扩展功能** 1](#_Toc56214335)

[**1.2功能实现分析** 1](#_Toc56214336)

[**1.2.1数据来源** 1](#_Toc56214337)

[**1.2.2数据操作相关功能** 1](#_Toc56214338)

[**1.2.3 GUI相关功能** 1](#_Toc56214339)

[**1.3数据处理分析** 2](#_Toc56214340)

[**1.4开发环境** 2](#_Toc56214341)

[**1.5用户界面设计** 2](#_Toc56214342)

[2、数据结构设计 4](#_Toc56214343)

[**2.1跳表介绍** 4](#_Toc56214344)

[**2.1.1基本概念** 4](#_Toc56214345)

[**2.1.2效率分析** 5](#_Toc56214346)

[**2.2数据存储结构** 5](#_Toc56214347)

[**2.3程序模块结构** 6](#_Toc56214348)

[**2.2.1整体结构** 6](#_Toc56214349)

[**2.2.2各模块功能描述** 7](#_Toc56214350)

[3、详细设计 7](#_Toc56214351)

[**3.1主界面绘制及控制模块Main** 7](#_Toc56214352)

[**3.1.1局部变量定义** 7](#_Toc56214353)

[**3.1.2函数定义** 7](#_Toc56214354)

[**3.2随机信息生成模块RandomHuman** 8](#_Toc56214355)

[**3.2.1局部变量定义** 8](#_Toc56214356)

[**3.2.2函数定义** 8](#_Toc56214357)

[**3.3跳表类模块SkipList** 9](#_Toc56214358)

[**3.3.1局部数据结构定义** 9](#_Toc56214359)

[**3.3.2函数定义** 9](#_Toc56214360)

[**3.4人员信息类模块Human/Human\_SinglePlace** 11](#_Toc56214361)

[**3.4.1局部变量定义** 11](#_Toc56214362)

[**3.5信息读取与处理模块tracer\_logic** 11](#_Toc56214363)

[**3.5.1局部变量定义** 11](#_Toc56214364)

[**3.5.2函数定义** 11](#_Toc56214365)

[**3.6地点信息界面绘制及控制模块Main** 11](#_Toc56214366)

[**3.6.1局部变量定义** 11](#_Toc56214367)

[**3.6.2函数定义** 12](#_Toc56214368)

[**3.7现实省份疫情状况获取模块status\_getter** 12](#_Toc56214369)

[**3.7.1函数定义** 12](#_Toc56214370)

[4、测试 12](#_Toc56214371)

[**4.1多日信息读取及处理功能** 12](#_Toc56214372)

[**4.2地图风险等级计算功能** 14](#_Toc56214373)

[**4.3现实疫情数据获取功能** 15](#_Toc56214374)

# 

**1、需求分析**

**1.1问题描述**

设计实现一个根据确诊病例的行踪,自动追踪其密切接触者的模拟追踪器.

### **1.1.1基本功能**

足量的候选人群和地点数据：

以地图形式,模拟确诊病人可能到达的至少8个地点场景（可以以你熟悉的真实地点的地图模拟）,比如,某小区、某单位、某学校、宾馆、超市、饭馆、交通工具、医院等；

每个场景需要有一定数量的人员信息（至少10人）,记录其在不同时间段曾经驻足,其中包含未采取有效防护措施人员.

参考新闻报道和相关资料,设计实现一个追踪算法,以某确诊病人在（至少在1天中）各个时间段到达的若干场景为（输入）依据（比如,此人依次去了5个地点）,自动查找出密切接触者,输出时,要求：

在地图中,以不同颜色,以图示方式按时间先后,依次动态显示每处结果（为了认真了解输出,显示的停留时间需要允许人为控制,比如按任意键继续）2.2、也能展示每处地点密切接触者的详细信息；改写相关地点的风险等级,图中用颜色区分风险等级；

结果保存到文件中,文件名设计要求合理.

### **1.1.2扩展功能**

允许用户添加新发现的某时段某地驻足人员；

允许一人多天追踪；

允许多人一起追踪,图中用不同颜色分别动态展示；

选择2个你本人觉得最实用功能,自行设计、实现并展示.

**1.2功能实现分析**

### **1.2.1数据来源**

由于需求中明确指出需要足量的候选人群数据,考虑到任务所需的数据难以通过网络爬取,且主要考查点在于功能实现和延时,因此选择编写算法随机生成合理的行程数据,好处在于可以随时取得大量的数据,便于debug和演示.

### **1.2.2数据操作相关功能**

数据存取功能——包括文件形式的结果存取,以及将数据从文件读入内存以进行后续操作的功能.

场景模拟——需要模拟8个地点,对应人员的行程信息,并可存储展示对应风险等级.

病情追踪——核心功能,将人员的行程信息与病情进行比对,追踪与病人密切接触的人员.

人员添加——允许用户向文件中存入新的人员信息.

多人多天追踪——允许多日的数据存取和多人信息的读取和追踪.

### **1.2.3 GUI相关功能**

场景地图——将8个地点场景可视化地展示为地图.

操作界面——实现一个便于用户操作和观察数据的界面.

疫情展示——将实际生活中的省份对应疫情信息进行展示.

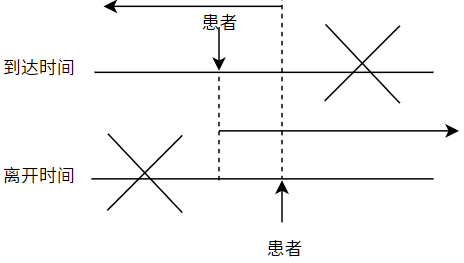
**1.3数据处理分析**

根据题目要求,输入的数据为文件中存储的多日人员的个人信息以及行程信息,经过数据处理后输出为与其产生密切接触的人员的个人信息和行程.

其中,需要进行关键处理、存取和比对的数据为各人员的行程信息,选择任意一个人员,都需要找出其他所有人员在其去过的对应地点中的行程信息进行比对,而输入的数据集本身可能也会很大,找出密接者以后还需要对其个人信息进行改写,因此选择的数据结构应当尽可能满足查找、插入、修改对应的时间复杂度都比较良好的条件,才能有较好的性能.

综上,考虑存储行程信息的数据结构,符合数据结构需求条件的结构有红黑树、跳表SkipList(四项基本操作时间复杂度均为O(logn))、B+树,而由于程序中涉及到的磁盘I/O操作并不多,B+树的性能相比前两者稍弱,可以首先排除,分析算法所需的具体操作在红黑树和跳表中进行选择.

具体到密接者查找的实现上,已知患者在一个地点下的到达时间和离开时间,首先排除掉一类人,这类人的离开时间在该患者的到达时间之前,或者到达时间在该患者的离开时间之后.剩余的人群即为密接人群.

****

*图1 密接者查找逻辑图示*

考虑四项基本操作以外最多的运算,即为查找某一区间内的数据,该操作在红黑树中需要通过遍历实现,但跳表由于最下层为顺序表,在其中只需要查找到对应区间的端点,之后向着一个方向继续查询其前驱或后继即可,此处时间复杂度对比为O(logn)与O(n),因此最终选择了跳表来存储各地点下人员的行程信息.

**1.4开发环境**

考虑到有着较实用的GUI需求,选择了JavaFX框架下的GUI实现,程序整体逻辑由JAVA实现.开发环境为IntelliJ IDEA(jdk 1.8.0\_261),搭配JavaFX SceneBuilder进行GUI实现.

**1.5用户界面设计**

考虑到地图模拟、多日追踪以及便捷地观察数据的需求,设计使用两个Scene,主界面Main中存放地图(实质上为按钮)以及日期选择框,按下地点对应的按钮后,打开一个TableView展示该地点下所有人员的个人和行程信息.同时,考虑到程序应当更加面向现实解决问题,添加实际省份疫情数据情况获取功能.使用JavaFX SceneBuilder可便捷地直接编写一个GUI Demo,并在实现完成后进行美化,各设计实现见下图.

****

****

*图2、3 GUI初步设计*

****

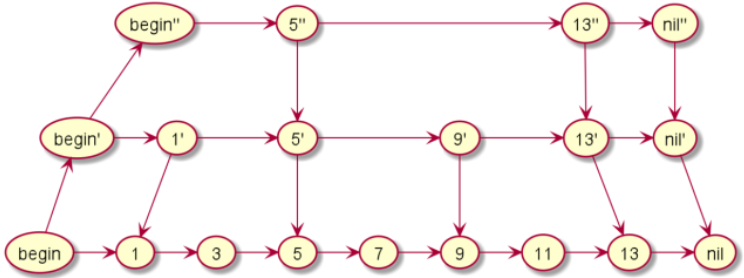
*图4 美化并添加完整功能后主界面*

**2、数据结构设计**

**2.1跳表介绍**

### **2.1.1基本概念**

跳表全称为跳跃列表,它允许快速查询,插入和删除一个有序连续元素的顺序表.跳跃列表的平均查找和插入时间复杂度都是O(logn).其实现快速查询的方式是通过维护一个多层次的链表,且每一层链表中的元素是前一层链表元素的索引（见示意图）.开始时,算法在最稀疏的层次进行搜索,直至需要查找的元素在该层两个相邻的元素中间.这时,算法将跳转到下一个层次,重复刚才的搜索,直到找到需要查找的元素为止.由于跳表的高效率和与红黑树或平衡树相比之下实现的简便性,在工业界中极为常用,如Redis数据库中的索引就是使用跳表进行实现.

****

*图5 跳表定义图*

### **2.1.2效率分析**

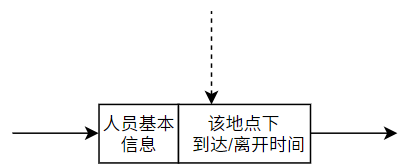
假设一个初始的顺序表有n个结点,如果每两个结点抽取出一个结点建立索引,那么第一级索引的结点数就是[n/2],第二级索引的结点数为[n/4],以此类推,第m级索引的结点数为[n/(2^m)].

假设现有m级索引,最高层的第m级的结点数为2个,通过上述规律,可得出[n/(2^m)]=2,从而求得m=log(n)-1.加上原始链表,整个跳表的高度就是log(n).在查询跳表时,如果每一层都需要遍历k个结点,那么最终的时间复杂度就为O(k\*log(n))=O(log(n)).根据建立索引的规则,k值将发生变化,在上述假定与本次实现中,均采用了两个结点对应一个索引的规则,故k值为2.具体到实现中时,严格遵循每2个结点建立一个索引的规则对输入的数据有一定要求,且实现也极为复杂,因此一般实现中都通过规定每个结点有随机的50%概率向上产生一级索引的设计来近似模拟,在数据量足够大的情况下,每个结点的平均层数符合的规律.但由于本设计中数据量有限,此处的时间复杂度为理想情况,实际情况可能在O(log(n))上下存在一定浮动.

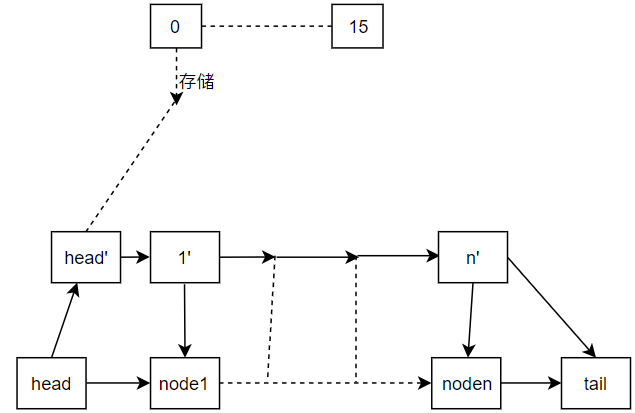
由于跳表底层为顺序表,插入和删除的时间复杂度也直接与查询的时间复杂度挂钩,同样为O(log(n)).

**2.2数据存储结构**

在1.5节中,已阐述了选择跳表作为主要数据存储结构的原因,该节阐述该存储结构的具体实现.当初始化将外部存储的数据加载到内存中时,首先分离出单个的人员,并将该人员的基本信息和行程信息继续分离,行程信息再进行进一步的分离,从而将外部数据存储到N(地点数,演示中定义为8)\*2(由于以时间作为索引键值,必须将到达/离开时间分开存储)共16个跳表中,这16个跳表以数组的形式进行存储,第[i]个跳表存储该地点下的到达时间,第[i+8]个跳表存储该地点下的离开时间.

****

*图6 跳表数据结点定义图(虚线表示可能存在的上层索引)*

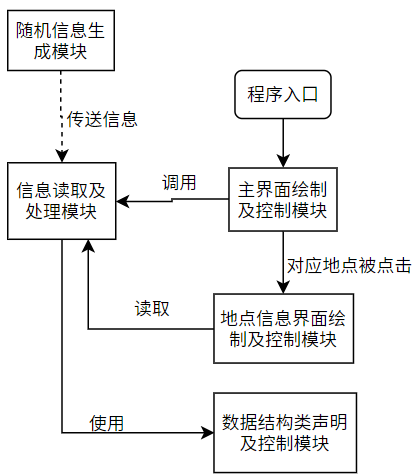
****

*图7 整体数据存储结构示意图*

**2.3程序模块结构**

### **2.2.1整体结构**

程序执行的整体流程设计如下:进入Main函数后,调用JavaFX相关函数进行界面绘制,读取界面中选择框内的日期,此时调用信息读取及处理模块读取对应日期的数据文件,通过声明的数据结构类对随机生成的信息进行处理后返回绘制模块.当界面绘制完毕后,等待用户点击任意地点,用户点击后,调用地点信息绘制模块,需要显示该地点下的人员信息,读取,绘制模块将对应信息显示到表单里,一次信息处理过程结束.

****

*图8 整体结构流程示意图(虚线表示仅在逻辑上关联,并无实际调用)*

### **2.2.2各模块功能描述**

(1)随机信息生成模块:给定所需人员信息的数量,随机生成该数量的人员信息以后将信息保存到指定文件当中.

(2)主界面绘制及控制模块:配合FXML文件,生成主界面并进行必要的初始化,以及进行信息处理模块的调用完成密接者查找,并按结果改写地区风险等级(通过地区颜色进行区分).

(3)信息读取及处理模块:将随机生成的数据读取到内存当中,调用跳表结构完成数据的存储,并在读取的过程中完成密接者的查找,改写密接者对应信息后保存到文件和内存中.

(4)地点信息界面绘制及控制模块:配合FXML文件,当用户按下地图中对应的按钮时,打开Table界面展示所有来过该地人员的行程信息.

(5)数据结构类声明及控制模块:共分三个类,第一个类为跳表类,该类中实现跳表结点和跳表类的定义,并定义对跳表进行增插删查的基本操作和配合信息处理模块完成密接者查找的函数;第二个类为人员信息类,存储人员的基本信息,用于跳表中的人员信息存储;第三个类为单个地点下的人员信息类,由于TableView中展示动态信息所需的数据类型为SimpleStringProperty,故定义该类实现数据的动态显示.

**3、详细设计**

**3.1主界面绘制及控制模块Main**

### **3.1.1局部变量定义**

(FXML文件中读取获得)Button型变量Button0~Button7:用于控制改变按钮颜色.

(FXML文件中读取获得)ChoiceBox型变量dayBox:用于控制添加和读取日期选择框内可选内容.

Stage型变量primaryStage、AnchorPane型变量rootLayout:在模块内部传递,用于场景构建.

String型静态变量ButtonName:被按下的按钮对应的场景名,用于传递到其他模块完成函数调用.

ChoiceBox型静态变量dayBoxSelected:选择框中选中的日期,用于传递到其他模块完成函数调用.

### **3.1.2函数定义**

1.main:程序入口点,调用JavaFX下的Launch函数显示界面.

2.start:设置主界面的Stage及标题,调用initRootLayout实现主界面场景构建.

3.initRootLayout:读取FXML文件到Scene中,设置相关参数后显示主界面.

4.ShowPersonOverview:调用人员信息控制模块,读取对应FXML文件,显示对应地点下的人员信息.

5.FloorClicked:地点按钮被按下时的回调函数,通过调用ShowPersonOverview显示人员信息.

6.setButtonColor:调用信息处理模块,获取密接人员数量,从而改写地区风险等级(通过颜色表明,当无人感染时为浅蓝色,有密接者则为浅黄色,密接者大于30人为橘色,大于50人为红色).

7.initialize:主界面初始化控制函数,将源目录下存有的信息文件名读取到选择框中,并初始化读取和计算人员信息为首选的日期对应的信息,并设置对应地点的风险等级.当选择内容改变时,重新读取计算数据并完成风险等级设置.

**3.2随机信息生成模块RandomHuman**

### **3.2.1局部变量定义**

String型静态变量firstName、woman、man:分别存储大量姓,女性名字和男性名字,用于随机姓名的生成.

String型静态数组road、places、telHead:分别存储大量街道名,地点名(此处演示设定为8个)和电话号头,用于人员的住址、电话号以及行程信息生成.

### **3.2.2函数定义**

1.main:程序入口,循环调用getInfo函数完成指定人数的人员信息生成,并保存到指定文件当中.

2.getInfo:分别调用其他各随机信息生成函数并通过字符串拼接的方式组合到一起,生成一个完整的人员信息并返回.

3.getNum:调用随机数函数生成指定范围内的一个随机整数.

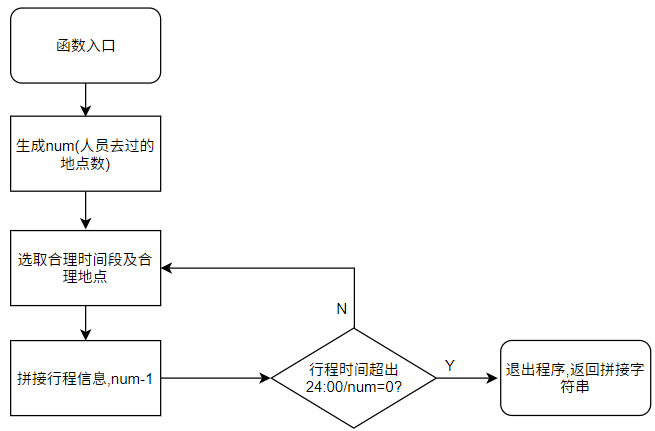
4.getTel:将电话号分为三部分,3位头,4位中部,4位尾部,其中头通过随机在telHead中选取获得,中部和尾部则通过不同范围内的(随机数+10000)删去一位获得,最终拼接组成一个完整的电话号码.

5.getChineseName:从对应的静态变量中选取对应的姓和名并拼接,通过随机生成0/1的方式决定人名是2位还是3位.

6.getRoad:生成人员住址和工作地点,在对应数组中选取街道后随机生成11~150的街道号,生成一个随机地址.

7.calTime:随机生成一个合理的时间,该时间满足:在输入的时间的20分钟~300分钟以内,以确保不会出现如在一个地点停留24小时的极端情况出现.

8.getRoute:生成人员的行程信息,首先随机生成可能去过的地点数,再通过calTime生成一段合理的时间,并在未去过的地点中挑选一个随机地点,由此生成一个地点下的行程信息,再通过循环执行即可实现随机地点数的随机行程信息生成.

****

*图9 getRoute函数流程图*

**3.3跳表类模块SkipList**

### **3.3.1局部数据结构定义**

在该类文件下定义一结点类SkipListNode,存储人员的个人和行程信息(其中int型变量time作为比较的键值,为便于比较,时间值在该结点中以整型存储)以及指向上下左右的四个指针.

在SkipList类内,定义静态double型变量PROBABILITY=0.5,作为跳表中结点上升一层的概率(理论上,想要达到严格的O(logn),上层结点个数也必须严格满足下层的1/2,但会因此产生大量多余操作, 代码也会十分复杂.考虑到实际应用中数据量并非无限大,这些多余操作无法忽略 ,因此通过概率方式近似模拟).

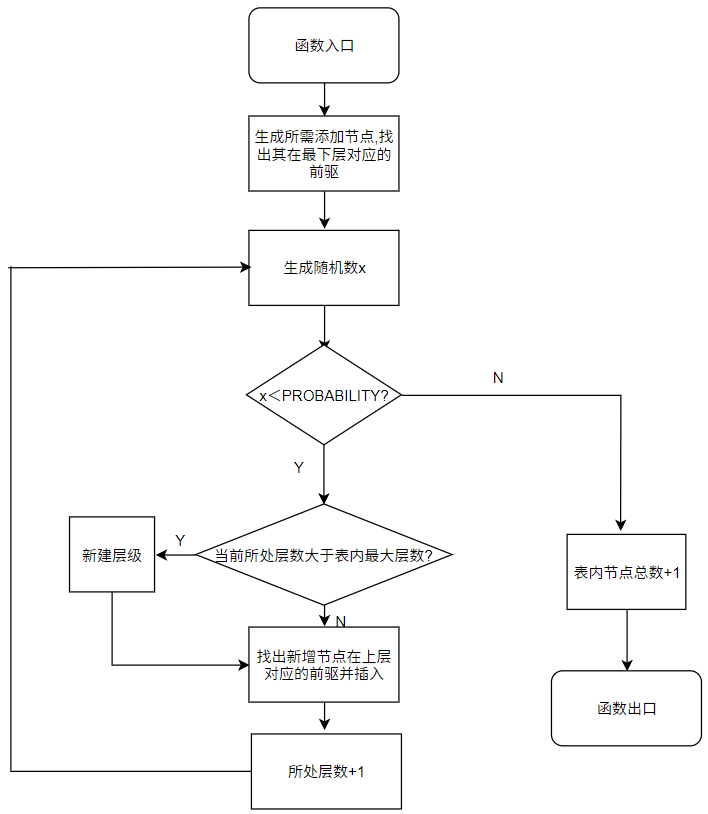
定义SKipListNode型变量,head及tail,int型变量HEAD\_KEY及TAIL\_KEY分别作为跳表的头尾结点和头尾结点标记值,其值分别为0和1440(对应0:00和24:00).

定义int型变量nodes和listLevel,分别作为跳表的结点总数和层数.

定义Random型变量random,用于产生随机数上升层数.

### **3.3.2函数定义**

1.put:向跳表中添加键值为k,索引值为v的结点,并产生随机值调整层级,产生上层结点.具体逻辑为:生成一个随机数,若此随机数小于PROBABILITY,则该结点需要产生上层结点.首先判断当前结点所处的层级是否超过了SkipList中存储的高度,如果是则新建一层并为其设置头尾结点,如果不是则找出其上升以后在上一层中应有的前驱,此后连接前驱后继以及上下结点,当前层数随之提升一层,如此循环直到生成的随机数大于等于PROBABILITY.

****

*图10 put函数流程图*

2.连接函数backLink、horizontalLink、verticalLink:功能分别为:1)在输入的第一个结点后插入输入的第二个结点.2)在输入的两个结点间建立横向双向连接.3)在输入的两个结点间建立垂直双向链接.三个函数均只需改变结点内的指针指向即可实现.

3.findNode:查找对应键值在跳表中应处于的位置,返回该位置的前一个结点.从最上层开始查找索引值,步入到最下层检索即可实现.

4.search:返回键值对应的结点,实现方式同findNode.

5.findPersonbyPhone:返回电话号码对应的结点.由于电话号码与键值无关,必须在最下层遍历所有结点查找.

6.findIlledPerson/findSuspectedPerson:返回跳表中从输入结点开始的第一位病人/密接者所对应的结点(通过结点下存储的个人信息区分),同样通过在最下层遍历实现.

7.getComparedNode:返回跳表中键值最接近输入值但比输入值大(小)的结点.只需返回输入值对应结点的前驱/后继即可.

8.FindCommonNodes:密接者查找的核心函数.找出两个跳表(分别存储到达和离开时间)下所有离开时间晚于病人到达时间的人员,再在其中选出所有同时在到达时间上小于等于病人离开时间的人员,这部分人员就是与输入数据相关联的密接者.在实现上,先通过findNode找到第一个离开时间晚于病人到达时间的结点头,再从该头开始向后遍历通过findPersonbyPhone在到达时间的跳表中查找满足条件的结点,最后将所有满足条件的结点数组返回.

**3.4人员信息类模块Human/Human\_SinglePlace**

### **3.4.1局部变量定义**

Human类中,定义了如下局部变量.

String型变量name,workspace,home,phoenumber存储对应人员的个人信息.

Boolean型变量isMaskOn表明该人员是否佩戴口罩.

Int型变量IllLevel表明该人员的感染等级(0:未感染,1:有防护密接,2:无防护密接,3:感染者).

Human\_SinglePlace中,仅将所有String类型改为SimpleStringProperty型,便于TabelView中信息的动态更新,此处不再赘叙.

**3.5信息读取与处理模块tracer\_logic**

### **3.5.1局部变量定义**

String数组类变量places:定义各地点的名称,便于与跳表数组对应.演示时设定共有8个地点.

SkipList数组类变量SL\_Places:定义存储各地点下人员的来往信息的跳表,长度为places.length\*2(演示中为16个跳表).

### **3.5.2函数定义**

1.Time\_String\_to\_Int:由于跳表底层将时间转化为一天中的分钟数,此类涉及信息的读写,因此需要有将String类的时间转化为Int型分钟的函数.

2.readInfo:读取对应日期的信息存储文件,并通过字符串的分离函数将所有信息按顺序拆分为:所有信息——单个人员的信息——个人信息与行程信息——具体个人信息与具体行程信息.拆分结束后将数据存入对应的跳表中,并调用calIllLevel完成所有地点下密接者的查询.

3.calIllLevel:对对应跳表中存储的所有人员进行感染等级计算.通过找出所有的已感染者,根据SkipList中的算法调用其中的FindCommonNodes函数找出所有密接者.再根据防护措施情况改写其感染等级.

4.getNumOfContacted:找出一个地点下所有密接者和感染者的总数,以便于地区风险等级的计算.实现时只需不断调用跳表类中的findSuspectedPerson找出下一个密接者即可.

**3.6地点信息界面绘制及控制模块Main**

### **3.6.1局部变量定义**

(FXML文件中读取获得)TableColumn型变量nameCol,phoneCol,workspaceCol,homeCol,arrivetimeCol,leavetimeCol,MaskCol,LevelCol以及TableView型变量information:用于表单的初始化和信息展示.

String型数组遍历places:存储地点名称,协助信息展示时的信息读取.

### **3.6.2函数定义**

1.initialize:FXML规定的界面初始化入口函数,调用setTableColumns进行表单信息设置,并对两个.

2.setTableColums:表单信息设置函数,通过与tracer\_logic中读取信息类似的方法,将字符串通过符号分割并改写(isMaskON从boolean型改写为是/否,感染等级也转换为对应的中文)后存入Human\_SinglePlace型临时变量中,并由各TableColumn完成信息读取和展示.

3.getHumanPlaceInfo:获取传入跳表中的人员信息函数,通过分离函数按格式分出各个信息,存为一个完整数组后传入setTableColumns中设置单个地点下的表单信息.

**3.7现实省份疫情状况获取模块status\_getter**

### **3.7.1函数定义**

1.sendGet:根据输入的url字符串,向对应服务器发送GET请求,获取服务器返回的内容并存入字符串返回.

2.get\_Province\_status:API接口调用函数,输入对应的省份名后,调用丁香园提供的信息获取API得到数据(通过url存放),调用sendGet函数进行GET获取,并进行属性的拆分处理后返回.

**4、测试**

**4.1多日信息读取及处理功能**

通过RandomHuman函数生成5日\*500人的随机数据后,修改部分人员的数据为已感染,点击任意地点的按钮进行查看,即可获取该地点下所有来往过的人员的相关信息,以及密接计算情况.

****

*图11 信息读取功能测试图*

与随机生成的数据文件进行对比,并通过人工比较来往时间判断密接情况,该表格显示的内容正确无误.

****

*图12 日期对应文件读取功能测试图*

再进一步测试日期选取情况,可看到日期对应文件被正常读取到多选框中,选择2日再次打开.

****

*图13 多日信息读取功能测试图*

经与实际文件比对,该表格显示内容正确无误.

**4.2地图风险等级计算功能**

****

*图14 风险等级计算测试图*

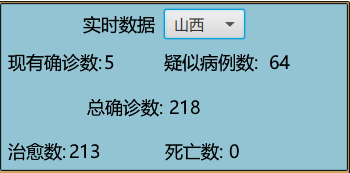
在1日的数据集中添加一定量的感染者,打开主界面观察地区风险情况,与实际密接者数量进行比较,风险等级绘制正确无误.

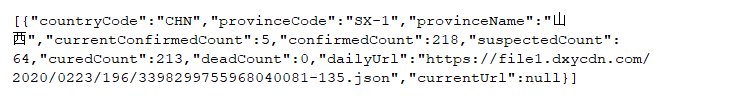
****

*图15 未感染地区测试图*

**4.3现实疫情数据获取功能**

在选择框中任意选择一个省份,观察API调用结果,并与直接GET到的数据进行比对.

****

****

*图16、17 疫情数据测试验证图*

经过比对,显示的数据与实际获取的完全相符.