总体方案设计说明

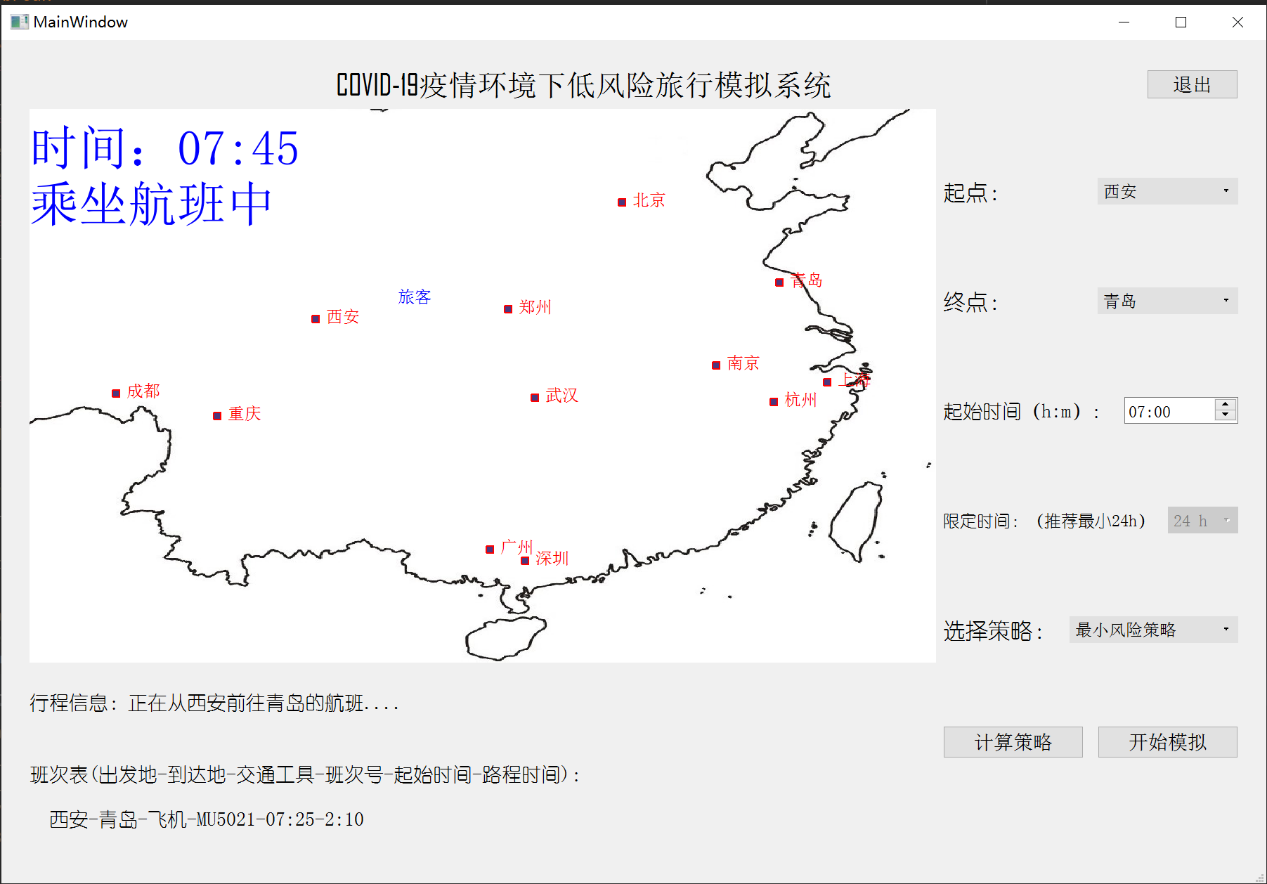
1. 软件开发环境
2. Win10系统
3. Python语言，Pycharm IDE
4. Sqlite3 数据库
5. Pyqt 图形界面设计（QT的python版本）
6. 总体结构
7. 数据部分
   1. 建立12个城市的信息，包括城市名字、经纬度坐标（模拟真实的位置）
   2. 建立班次的时间表，包括飞机、火车、汽车
   3. 班次包括起点、终点、交通工具、起始时间、旅程时间
   4. 为城市设立风险值；为不同交通工具设置不同风险值（完成选做）
8. 旅行计算部分
   1. 建立旅客信息，包括起点、终点、当前时间、选择的策略、选择的航班等
   2. 在选择所有的参数后，能够输出计算出来的路程计划，包括所有计划的航班
   3. 选择的策略包括最小风险策略、限时最小风险策略
   4. 最小风险策略：无时间限制，风险最少即可

策略：使用A\* 算法，可以快速计算出最小路径，它相比dijkstra算法，多了启发式的预估移动耗费（从节点到终点的预估），有更好的先见之明，所以能够更快的寻找最短路径。相比BFS来说，它有更好的性能，因为它不需要遍历所有可能路径。

每调用一次A\* 算法，弹出从开启列表中加权危险值最小的节点（城市），然后计算从这个节点能够到达的相邻节点的加权危险值，在把他们加入到开启列表中。当弹出的节点为目的地时，即完成计算旅游路径。

* 1. 限时最少风险策略：在规定的时间内风险最少

策略：同最小风险策略，使用A\* 算法，但是我们在算法的过程中加入了限制条件，即在每次计算时，需要判断达到相邻节点时是否会超出限时的时间，若超出，则不把它加入到开启列表中。

1. 旅行模拟部分、图形交互界面（完成选做）
   1. 建立图形界面，包括实时显示航班信息和旅客信息、选择参数
   2. 能够模拟真实的时间（比例缩放时间），限时旅客位置
   3. 设置日志功能，能够输出到log日志文件中
2. 模块划分
3. 主函数模块
   1. 调用数据模块生成数据
   2. 调用主图形界面模块显示界面
   3. 在主界面中调用策略模块生成路径，在主界面中实时模拟旅客位置
4. 数据模块（Data.py）
   1. 读取所有城市的信息：城市名、经纬度
   2. 读取班次信息：包括飞机、火车、汽车
   3. 处理所有数据，使得他们成为更容易使用的数据结构（如构成字典或列表型的数据结构）
5. 策略模块
   1. 实现计算旅行路程的功能
   2. 能够处理不同的策略和不同选择下的计算，返回计算出来的最优路径
6. 主图形界面
   1. 模拟真实的地图，按照经纬度比例缩放显示所有城市
   2. 显示所有可以改变的参数，包括：起点、终点、起始时间、限时时间、选择策略
   3. 显示计算出来的旅行路径，并能够模拟真实时间，在城市之间移动