模拟退火算法

1. 实验目的

使用模拟退火算法来解决TSP实际问题。

1. 实验内容

用模拟退火算法求解31个城市的TSP问题（从第一个城市出发最后再回来）

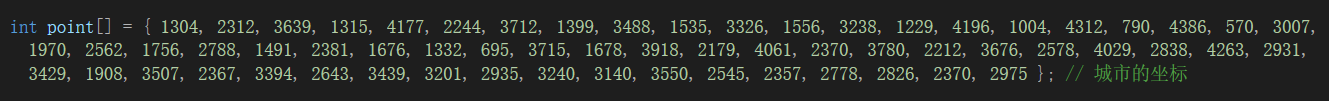
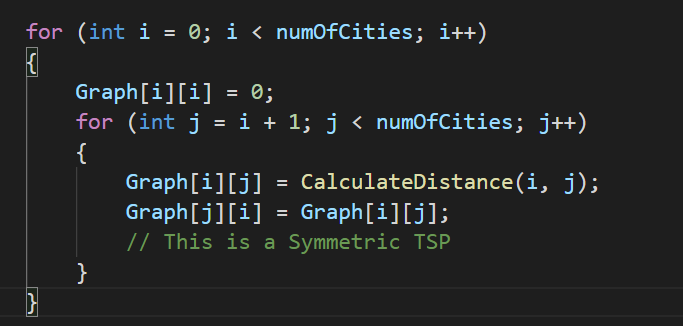
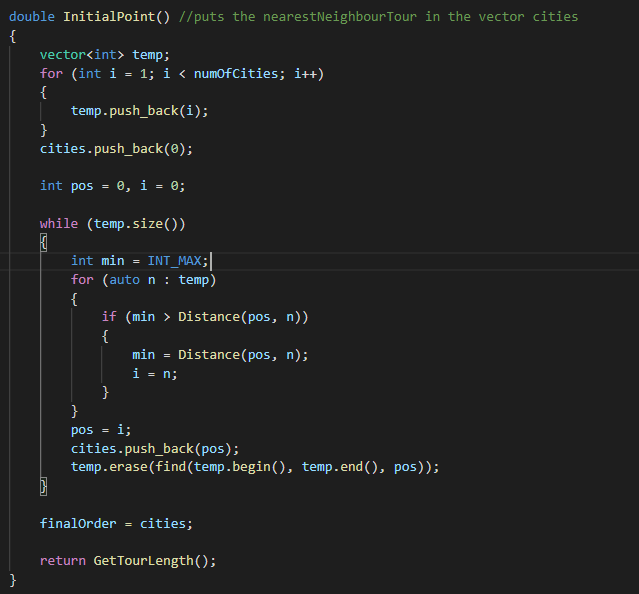
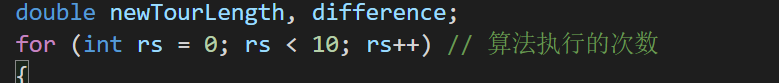
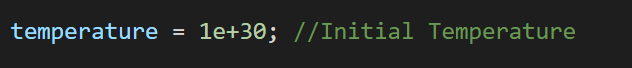
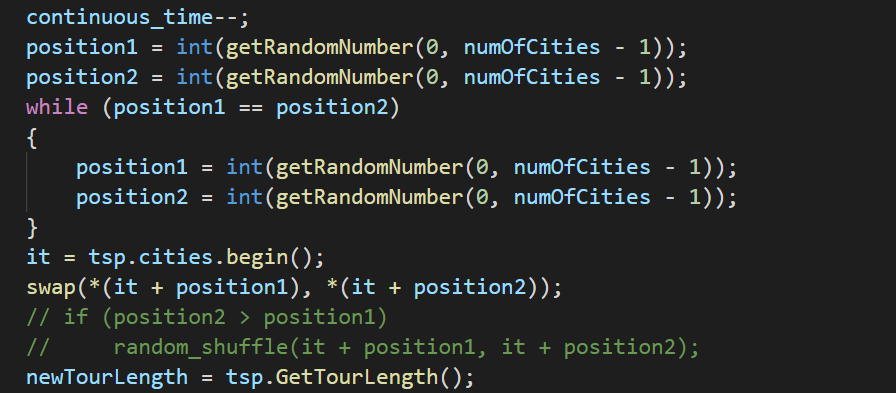
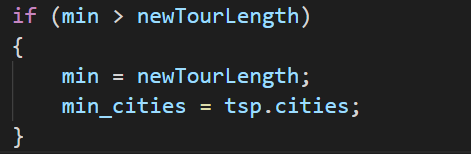
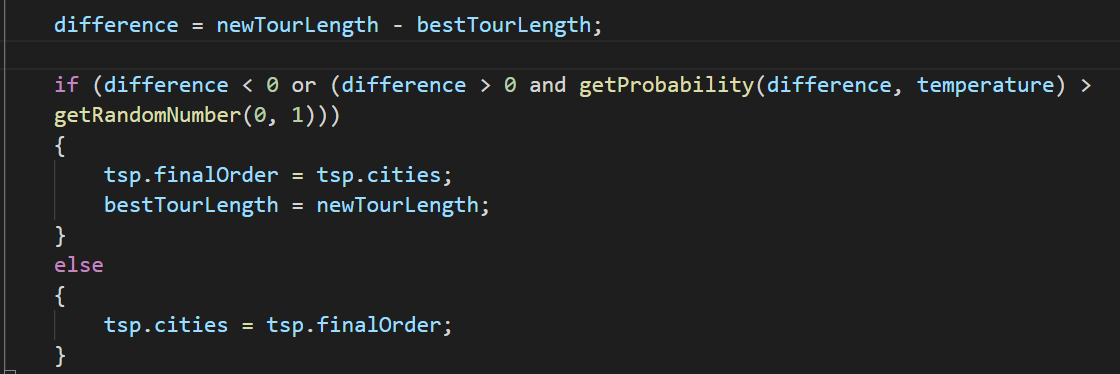
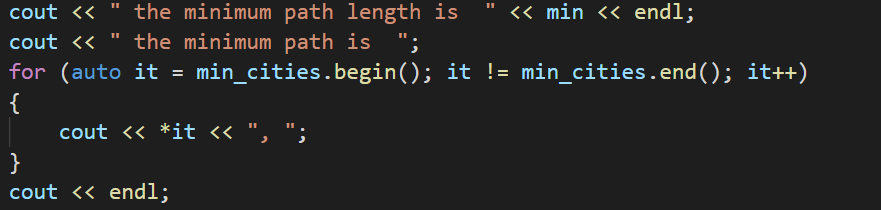
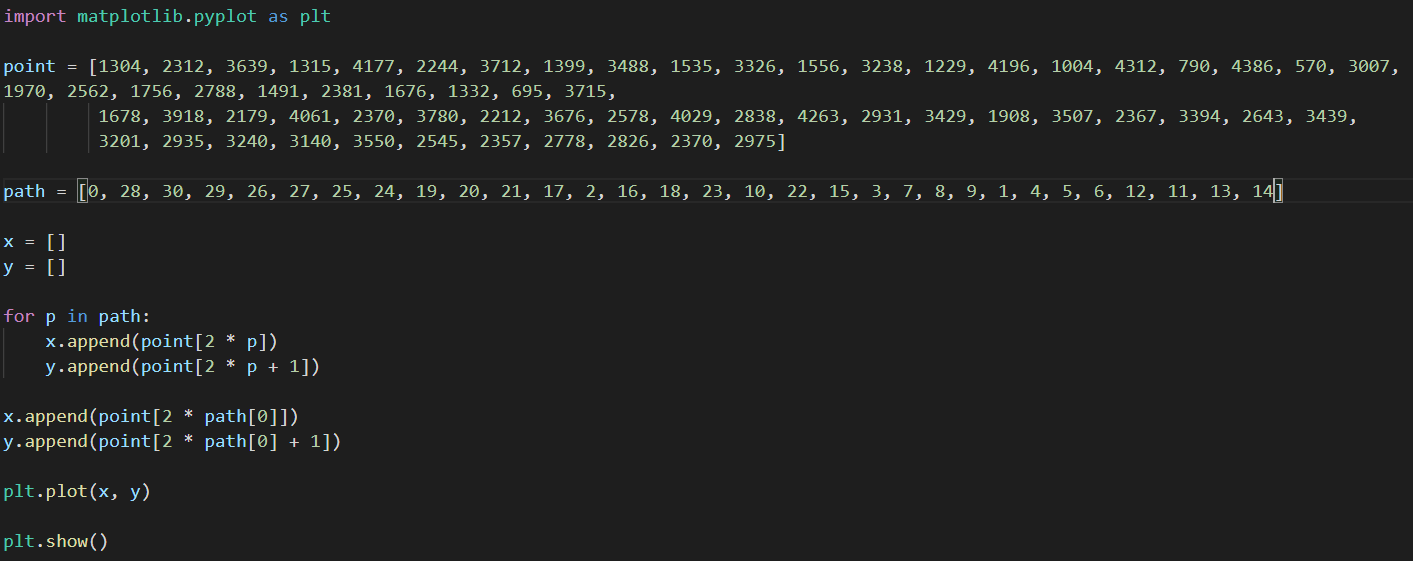
全国31个省会城市的坐标为[1304 2312; 3639 1315; 4177 2244; 3712 1399; 3488 1535; 3326 1556; 3238 1229; 4196 1004; 4312 790; 4386 570; 3007 1970; 2562 1756; 2788 1491; 2381 1676; 1332 695; 3715 1678; 3918 2179; 4061 2370; 3780 2212; 3676 2578; 4029 2838; 4263 2931; 3429 1908; 3507 2367; 3394 2643; 3439 3201; 2935 3240; 3140 3550; 2545 2357; 2778 2826; 2370 2975]

1. 实验环境

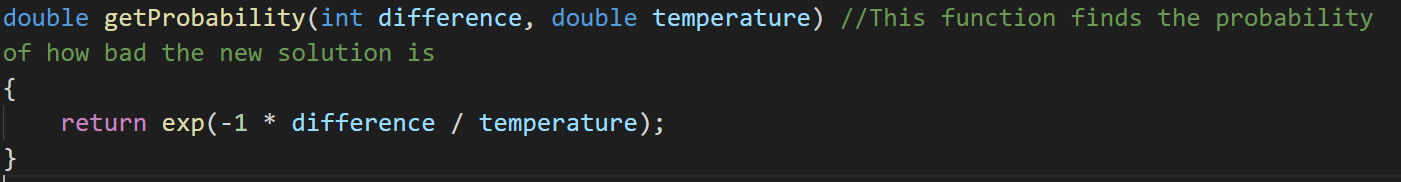
Visual Studio 2019

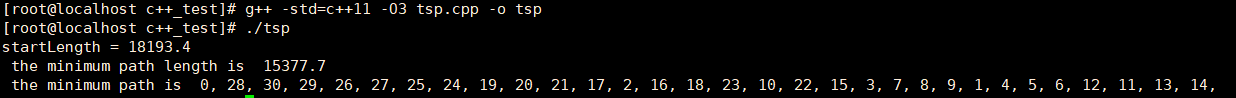
本地进行测试，服务器调节数据跑实验

C++语言进行开发

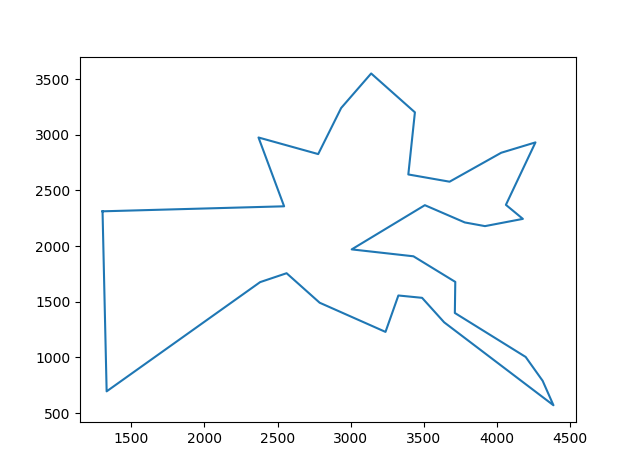
1. 实验步骤
2. 首先初始化每个城市的坐标。
3. 创建一个记录每个点之间的距离的矩阵，并计算出相应的值放入矩阵。
4. 使用贪心算法初始化一个路径并计算该路径的长度。
5. 由于该算法初始值会影响结果，因此我考虑在程序中使用不同的初始值多次运行。
6. 设置初始温度。
7. 判断温度是否低于最低温度。
8. 设置在同一个温度下所处的时间（次数）。
9. 随机找两个城市互换。
10. 如果是最小路径，存储最短路径和对应的长度。
11. 通过接受准则判断是否接受当前解。
12. 在当前温度持续时间结束之后，设置当前温度下的最好解为当前解，并进行降温。
13. 随机打乱初始解，进行下一次算法的运行。
14. 最终输出答案。
15. 结果可视化。
16. 实验结果

在这里，我简单的将程序进行了并行，以将CPU多核充分利用并提高效率。

根据我计算机的配置，我选择了如下的参数：每个核算法的执行次数为10，初始温度为1e+30，降温速率为0.99，持续时间为1000，计算可能性函数我使用的是如下的常用的函数

下面是使用该参数得到的结果。

下面是将该路程可视化后得到的结果。



可以大致看出没有交叉路线，并且我进行了多次执行程序，每次都输出了这个答案，说明该程序的鲁棒性很好，也说明可能该解是一个很好的解。

1. 实验总结

模拟退火算法来源于固体退火原理，是一种基于概率的算法，将固体加温至充分高，再让其徐徐冷却，加温时，固体内部粒子随温升变为无序状，内能增大，而徐徐冷却时粒子渐趋有序，在每个温度都达到平衡态，最后在常温时达到基态，内能减为最小。模拟退火算法是通过赋予搜索过程一种时变且最终趋于零的概率突跳性，从而可有效避免陷入局部极小并最终趋于全局最优的优化算法。

由于该实验的结果受初始值、初始温度，降温速率、在某一温度下的持续时间、计算可能性的函数影响较大。通过我进行的多次实验，我发现，  
想要降低初始值的影响，就需要增加算法的执行次数并记录最好的结果；  
想要得到越好的结果，这三个数值 初始温度，降温速率、在某一温度下的持续时间 都需要相应的增大，但同时算法又需要更多的时间。