# 问题重述

由于中国经济的快速发展，城市空间环境的复杂性急剧增加，事故和灾害时有发生，安全风险增加。消防和救援队承担的任务也多种多样，复杂多样。某处有15个区域，根据火灾报警数据建立数学模型，以完成以下问题：

问题1：每天设置三个时间段，每个时间段安排至少5人值班。假设消防队每天待命30人，确定消防队每年2月、5月、5月、11月的第一天需要待命的人数。

问题2：根据2016年1月1日至2020年12月31日的数据，建立消防救援警察部队人数预测模型，并把2020年的数据作为验证集，并预测2021年消防救援出警次数。

问题3：根据7个事件的时间，建立了多种事件数量与月数关系的数学模型，每个事件的最佳模型确定最佳拟合度作为评价标准。

问题 4：分析 2016 年至 2020年该地区事件密度的空间相关性，并为给出相关性最强的事件类别。

问题5：分析事件密度与人口密度之间的关系。

问题6：目前这个地区有两个消防站，考虑不同的因素，确定应该在哪个地区建造新的消防站。如果从2021年至2029年每三年建造一个新的消防站，应该依次建设在哪些区域？

# 问题分析

## 问题一的分析

本题将每天分为三个时间段（0:00-8:00为时段Ⅰ，8:00-16:00为时段Ⅱ，16:00-24:00为时段Ⅲ），最终目的为确定消防队在每年2月、5月、8月、11月中第一天的三个时间段各应安排多少人值班。根据常识我们知道，不同时间段所安排的值班人数应当与该时间段预计的消防工作量呈正相关。所以，解决本问题也就是解决：2月、5月、8月、11月中第一天各个时间段的工作量的估计问题。为了对所求日期各个时间段的工作量进行较为精确的预估，我们应该从往年的数据入手，提取出大量较能反应该日工作量的数据。就比如对二月一号估计时，二月一号是一月二月的分界日期，所以取一月和二月的所有数据取均值，得到估计数据。与此同时，我们还应该考虑偶然性较强但是也具有一定代表性的数据，比如单取往年的二月一号的数据，进行统计，得到第二组数据。再通过层次分析的方法，确定这两组因素的影响程度再得到工作量的最终预估值。

## 问题二的分析

问题二要求我们预测2021年的出警次数，这是一个典型的时间序列问题。我们只需建立时间序列模型，求得季节因子与长期趋势变动因子，就可以实现预测。

## 问题三的分析

## 问题四的分析

## 问题五的分析

本题目要求分析该地各类事件密度与人口密度之间的关系。根据已给数据，我们首先应当计算出各类事件的事件密度以及相应的人口密度。在得到数据后，我们可以初步观察得知，事件密度与人口密度大致呈正相关的关系。所以，为了确定二者之间的函数关系。我们可以进行Polynomial函数拟合以及Power函数拟合。通过拟合之后的误差分析以及过拟合状态的判断，确定我们的最佳拟合函数曲线，依此分析出各类事件密度与人口密度之间的关系。

## 问题六的分析

问题六要求我们探求一个消防站的建立区域以及三个消防站的建立位置与建立顺序。消防站的位置应当使平均出警距离最短。这样，本题就转化为图论当中的最小路径问题。这样，我们通过Dijkstra算法寻求到最短路径，解决了第一小问和第二小问的第一部分。第二小问的第二部分我们转化为一个规划问题，使得建设期间的出警距离最短，进而求解出建设顺序。

# 模型假设

1. 消防站建成后立刻投入使用。
2. 出警路上不会堵车。
3. 出警的频率占比在模型的使用中视为概率。

# 符号说明

|  |  |  |
| --- | --- | --- |
| **符号** | **说明** | **单位** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# 模型的建立与求解

## 问题一模型的建立与求解

### 模型的建立

本题目的目标为：确定2月、5月、8月、11月中第一天的三个时间段各应安排多少人值班。根据我们的生活常识可知，人数的安排应该和工作量的分布成正相关。那么，本题的目标即可转化为：对2月、5月、8月、11月中第一天的三个时间段出警频率的估计。

为了求解各个时间段预计的出警频率，我们从往年出警数据中提取了两类数据进行分析。首先，因为求解的是：2月、5月、8月、11月中第一天。所以我们提取了1月和2月、4月和5月、7月和8月、10月和11月的出警数据，进行了数据统计，并依此得到了第一组工作强度分配数据。其次，考虑到这四个月的“第一天”可能存在一些共性的影响成分，所以我们单独提取了2016年到2020年2月、5月、8月、11月中第一天的出警数据。统计后，我们得到了第二组工作强度分配数据。但是，鉴于第二组数据样本容量过小，且偶然性过大，可参考的程度较小。鉴于所考虑的对工作强度的影响因素较少，我们采取层次分析法对两组数据进行了处理，并依此得到最终的工作强度分配估值。

### 模型的求解

得出第一组数据的方式为：将1月和2月、4月和5月、7月和8月、10月和11月的出警数据中，各个时间段各自相加并与总出警次数作比。第二组数据的得出方式与第一组相似，不过统计的对象变为了单一的2月、5月、8月、11月中第一天。得出的结果见图 1。

图 1 工作强度分配比例图

鉴于两组数据的可靠性不同，我们采用层次分析法对两组数据进行处理。我们认为，对于最终的判断，第一组数据相对于第二组数据呈“极端重要”，权重比为9:1。在经过一致性检验后，我们得到了最终的分配比例，见图 2。

图 2 最终分配比例图

依据题目要求，每天共有30人可以安排，每个时段至少安排5人。则依此可得人员分配数量y与工作强度分配比的关系为：其中基础人员数量，人员增量与强度增量相关增长系数。而在仅考虑工作强度的情况下，人员分配与工作强度分配比的关系为：其中，。对于这两种函数关系，我们再次使用层次分析法。对于这两种考虑方式，我们认为其“同等重要”，权重比为1:1。经过一致性检验后，我们得到第一题的最终结果，每年2月、5月、8月、11中第一天的三个时间段各应安排值班人数，见图 3。

图 3 每天各时段分配人数图

## 问题二模型的建立与求解

时间序列法是一种定量预测法, 在统计学中作为一种常用的预测手段已被广泛应用。时间序列模型主要包括剔除季节变动法、自回归模型、移动平均模型、自回归移动平均模型、差分自回归移动平均模型。

一般而言，时间序列有4种变动因素：长期趋势（T）、周期变动（C）、季节变动（S）、扰动项（I）。我们这里不考虑周期变动，那么我们可以使用叠加模型或乘积模型建立时间序列预测模型。

* 叠加模型：
* 乘积模型：

我们采用叠加模型，首先以该地2016年1月1日至2019年12月31日的数据为基础利用SPSS求出季节变动因子（S），见表 1。

表 1 季节因子

|  |  |
| --- | --- |
| 周期 | 季节因子 |
| 1 | -4.54456 |
| 2 | 13.20544 |
| 3 | -2.51678 |
| 4 | -20.93345 |
| 5 | 67.20544 |
| 6 | 53.89988 |
| 7 | -12.50984 |
| 8 | -23.23900 |
| 9 | -18.37789 |
| 10 | -30.01678 |
| 11 | -18.15567 |
| 12 | -4.01678 |

下面根据剔除季节因子后的数据建立一元线性回归模型来求得长期趋势因子（T）。

其中，代表时间与2015年12月之间相距的月数，解得，，。

下面我们根据上述模型求解，结果见图 4、表 2。

图 4 时间序列图

以2020年1月1日至2020年12月31日的数据作为模型的验证数据集，为0.5141，均方误差为1043.15，说明模型的预测效果较好。

表 2 问题2的结果

|  |  |
| --- | --- |
| 月份 | 预测值（次） |
| 2021年1月 | 53 |
| 2021年2月 | 70 |
| 2021年3月 | 54 |
| 2021年4月 | 36 |
| 2021年5月 | 123 |
| 2021年6月 | 110 |
| 2021年7月 | 43 |
| 2021年8月 | 32 |
| 2021年9月 | 37 |
| 2021年10月 | 25 |
| 2021年11月 | 37 |
| 2021年12月 |  |

## 问题三模型的建立与求解

### 模型的建立

图 5各类事件次数与月份图

本题需要求解的是7种事件发生次数与月份关系的数学模型。因为研究对象与年份无关，所以先将数据根据年份进行求和根据月份进行分类，见图 5。

通过分析每一类事件的发生次数与月份时间的关系的折线图，可以发现每一类事件的次数曲线的变化趋势都不相同，因此我们先采用绘制时间序列图，来确认事件发生次数是否符合趋势，再根据不同发生年份的折线趋势是否交叉判断是否存在季节性，绘制图像如下。

图 6 事件①次数年度折叠时间序列图

图 7 事件②次数年度折叠时间序列图

由图 6、图 7可以看出，事件①的发生次数无明显趋势和季节性，事件②~⑦有明显的季节性。因此我们将使用时间序列预测来描述描述事件①，和事件②~⑦，由于事件②~⑦具有一定的相似性，因此我们以事件②为代表进行分析。

### 模型的求解

### 事件①模型求解

1. 确定趋势成分

首先绘制时间序列的线图，利用回归分析为次数和月份的关系拟合一条趋势线，然后对回归系数进行显著性检验，见图 8。

图 8 事件①时间序列

若用线性趋势进行描述得到回归方程为y = -0.0194x + 7.4073，R² = 0.004。观察到时间序列两端高于趋势线的数值，表明可能存在非线性趋势。如果拟合二阶曲线y = 0.0103x2 - 0.6492x + 13.916，R² = 0.2737大于一阶。说明二阶拟合性更好。

1. 确定季节成分

利用2016~2020五年的数据，绘制出年度折叠时间序列图。观察各个年份图线间的交叉情况，确定是否有季节性。

图 9 事件①次数年度折叠时间序列图

根据图 9发现事件①既不存在趋势也不存在季节性，因此采用平滑预测法来拟合数据。根据公式：

得出α为0.2预测效果最好。

### 事件②的模型求解

与上步相同，首先进行确定趋势成分，然后再进行季节成分分析，发现事件②有较好的趋势性和季节性，因此分离季节因数见表 3。

表 3 季节因子

|  |  |
| --- | --- |
| 周期 | 季节因子（%） |
| 1 | 84.0 |
| 2 | 79.6 |
| 3 | 71.1 |
| 4 | 153.4 |
| 5 | 184.2 |
| 6 | 37.9 |
| 7 | 106.3 |
| 8 | 48.0 |
| 9 | 103.3 |
| 10 | 51.7 |
| 11 | 164.5 |
| 12 | 116.1 |

图 10 事件②拟合预测表

通过数据整理去除季节因子，得出数据并绘制以下图像：

通过回归分析得出六阶回归方程为

得到。

### 模型总结与分析

### 事件①的平滑指数模型短期预测效果较好，但不适用于长期预测，因此在本题中我们通过回归检验认为季节性预测法的拟合和预测效果较好。

## 问题四模型的建立与求解

## 问题五模型的建立与求解

### 模型的建立

本题目要求分析该地各类事件密度与人口密度之间的关系。通过所给的数据，我们不难求出各个地区的人口密度和各个地区各个事件的事件密度。由于本题探讨的重心在于事件密度与人口密度，所以我们忽略了不同地区的地区差异，以及接警时间这个因素。

经过大致观察，我们可以发现：各类事件的事件密度与人口密度均呈正相关。所以，我们以人口密度为自变量，事件密度为因变量，以各个区域为数据点，进行了Polynomial一次、二次函数拟合以及Power函数拟合。在比较了各类函数的拟合效果之后，大致确定了各类事件的事件密度与人口密度间的函数关系。

### 模型的求解

对每组数据均进行了拟合后，根据我们拟合所得函数的拟合度，我们确定了七个事件所对应的事件密度与人口密度的最优函数关系。各函数的最优拟合度参数见表 4。

表 4 拟合效果表

|  |  |  |
| --- | --- | --- |
| 事件\拟合度 | SSE | R² |
| 1 | 1.06E-06 | 0.9998 |
| 2 | 5.98E-07 | 0.9989 |
| 3 | 2.75E-06 | 0.9998 |
| 4 | 8.59E-07 | 0.9998 |
| 5 | 3.52E-08 | 0.9995 |
| 6 | 2.29E-07 | 1.0000 |
| 7 | 1.50E-05 | 0.9999 |

以人口密度为x，事件密度为y，各类事件的最佳拟合函数如下：

事件1：

事件2：

事件3：

事件4：

事件5：

事件6：

事件7：

图表

中度可信度描述已自动生成

图 11 拟合效果图

在进行Polynomial拟合的过程中，如果将次数提升至三次或更高则会出现过拟合现象，虽然值更趋近于1，但是不予采纳。以事件3作为过拟合示例，结果见图 11。

通过误差检验，我们的拟合函数在误差不大于0.001%的条件下，认为可以反应出人口密度与事件密度的关系。拟合完成后，我们即可得到共性的结论：各类事件的事件密度均与人口密度呈正相关关系。且1、2、3、5事件更加符合二次多项式的函数关系，4、6、7事件较为符合幂函数的函数关系。各类事件的大体趋势相近，下面只展示拟合效果最佳的事件6作为示范，见图 12。

图表, 折线图

描述已自动生成

图 12 拟合效果图

## 问题六模型的建立与求解

本题首先要求建立一个消防站，那么消防站的位置一定是最便利的，也就是到各个点的平均出警距离是最小的。那么本题就转化为一个图论中的最短路径问题，我们下面利用Dijkstra算法求解。

### 求解各对顶点之间的最短距离

Dijkstra算法，是1959年由荷兰科学家狄克斯提出的用于计算有向图中从一个顶点发到达图中其余顶点所经过的最短路径算法。算法把图中的所有顶点分成两个集合和。集合中存放已找到最短路径的顶点，中存放当前还未找到最短路径的顶点。算法将按照最短路径长度递增的顺序逐个将集合中的元素加入到集合中，直至所有顶点 都进入到集合为止。

具体步骤如下：

**STEP 1**：初始时，集合中仅包含源点，集合中包含除源点以外的所有顶点，到V-S中各顶点的路径长度或者为某个权值（如果它们之间有弧相连），或者为无穷（没有弧相连）；

**STEP 2**：按照最短路径长度递增的次序，从集合中选出顶点路径长度最短的顶点加入到S集合中；

**STEP 3**：加入之后，为了寻找下一个最短路径，必须修改从到集合中剩余所有顶点的最短路径。若在路径上加入之后，使得到的路径长度比原来没有加入时的路径长度短，则修正到的路径长度为其中较短的；

**STEP 4**：重复以上步骤，直至集合中的顶点全部被加入到集合中；

**STEP 5**：根据集合，求出各对顶点之间的最短距离，见表 5。

表 5 最短距离矩阵

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 距离 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | P |
| A | 0 | 11.1 | 19.1 | 11.4 | 25.7 | 18.3 | 30.7 | 39.9 | 48.9 | 17.8 | 23.2 | 27.6 | 8.2 | 21.4 | 19.9 |
| B | 11.1 | 0 | 8.2 | 12.8 | 19.3 | 17.6 | 33.4 | 51 | 60 | 25.5 | 34.3 | 38.7 | 19.3 | 22.8 | 21.3 |
| C | 19.1 | 8.2 | 0 | 7.7 | 11.1 | 9.4 | 28.3 | 46.6 | 55.6 | 20.4 | 29.9 | 34.3 | 22 | 17.7 | 16.2 |
| D | 11.4 | 12.8 | 7.7 | 0 | 14.3 | 6.9 | 20.6 | 38.9 | 47.9 | 12.7 | 22.2 | 26.6 | 14.3 | 10 | 8.5 |
| E | 25.7 | 19.3 | 11.1 | 14.3 | 0 | 7.4 | 29.2 | 51.7 | 60.7 | 25.5 | 35 | 39.4 | 28.6 | 18.6 | 22.8 |
| F | 18.3 | 17.6 | 9.4 | 6.9 | 7.4 | 0 | 21.8 | 44.3 | 53.3 | 18.1 | 27.6 | 32 | 21.2 | 11.2 | 15.4 |
| G | 30.7 | 33.4 | 28.3 | 20.6 | 29.2 | 21.8 | 0 | 26.8 | 35.8 | 12.9 | 13.4 | 14.5 | 22.5 | 10.6 | 16.5 |
| H | 39.9 | 51 | 46.6 | 38.9 | 51.7 | 44.3 | 26.8 | 0 | 9 | 26.2 | 16.7 | 12.3 | 31.7 | 33.1 | 30.4 |
| I | 48.9 | 60 | 55.6 | 47.9 | 60.7 | 53.3 | 35.8 | 9 | 0 | 35.2 | 25.7 | 21.3 | 40.7 | 42.1 | 39.4 |
| J | 17.8 | 25.5 | 20.4 | 12.7 | 25.5 | 18.1 | 12.9 | 26.2 | 35.2 | 0 | 9.5 | 13.9 | 9.6 | 6.9 | 4.2 |
| K | 23.2 | 34.3 | 29.9 | 22.2 | 35 | 27.6 | 13.4 | 16.7 | 25.7 | 9.5 | 0 | 4.4 | 15 | 16.4 | 13.7 |
| L | 27.6 | 38.7 | 34.3 | 26.6 | 39.4 | 32 | 14.5 | 12.3 | 21.3 | 13.9 | 4.4 | 0 | 19.4 | 20.8 | 18.1 |
| M | 8.2 | 19.3 | 22 | 14.3 | 28.6 | 21.2 | 22.5 | 31.7 | 40.7 | 9.6 | 15 | 19.4 | 0 | 16.5 | 13.8 |
| N | 21.4 | 22.8 | 17.7 | 10 | 18.6 | 11.2 | 10.6 | 33.1 | 42.1 | 6.9 | 16.4 | 20.8 | 16.5 | 0 | 5.9 |
| P | 19.9 | 21.3 | 16.2 | 8.5 | 22.8 | 15.4 | 16.5 | 30.4 | 39.4 | 4.2 | 13.7 | 18.1 | 13.8 | 5.9 | 0 |

其中，代表第个点到第个点的最短距离。其中，当时，自身之间的距离为0。

### 最优位置的选择

消防站位置应当是到各个点路径之和最短的一点，但是每个点的事故发生频率并不相同，我们基于其在总接警次数的百分比设置不同的权重，见表 6。

表 6 接警地区分布表

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 地区 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | P |
| 接警次数 | 157 | 117 | 92 | 153 | 245 | 153 | 196 | 105 | 73 | 155 | 180 | 101 | 216 | 99 | 1641 |
| 权重 | 4.26% | 3.18% | 2.50% | 4.15% | 6.65% | 4.15% | 5.32% | 2.85% | 1.98% | 4.21% | 4.89% | 2.74% | 5.86% | 2.69% | 44.56% |

下面需要求得一点使得平均出警距离

最小。其中分别代表其与已经建成的消防站J、N的距离。经过计算，结果见图 13，消防站应当建立在P点，使得平均出警距离最小，为7.31。

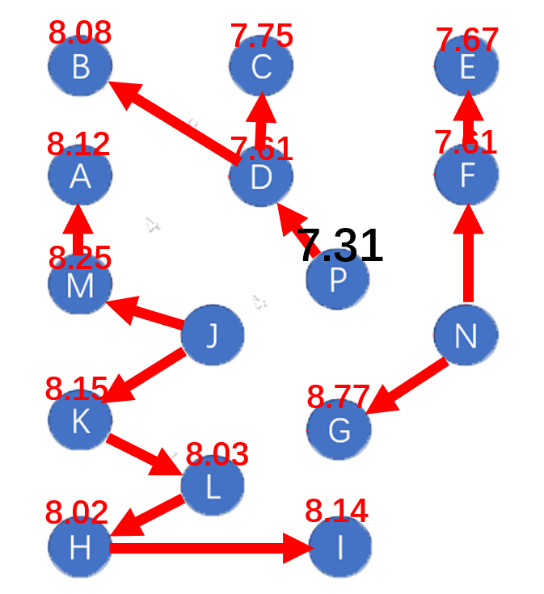


图 13 问题六路径图

### 最优建立顺序的确定

本题要求建立三个消防站，并确立它们的建立顺序。我们首先关注三个消防站的最终建立位置，使得消防员的出警距离最短。

下面需要求得三点使得平均出警距离

最小。经过三层循环枚举计算，出警路线见图 14，三个消防站应当建立在E、H、P点，使得平均出警距离最小，为4.42。

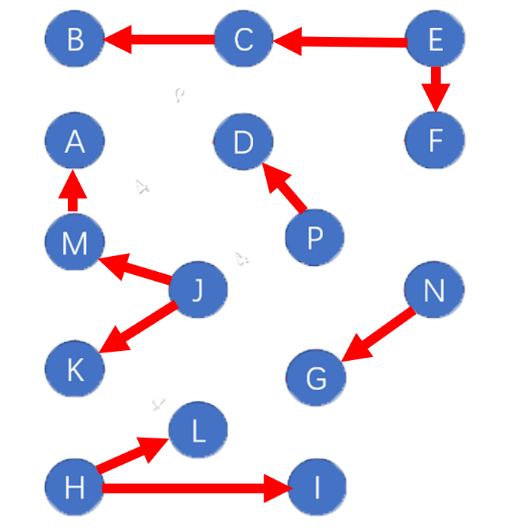


图 14消防站出警路线图

下面假设依次建设a、b、c三个消防站，建设顺序应当使建设期间的总出警距离

最小，化简与剔除常数项后即为

经过Python求解，建设顺序应为P、E、H。

# 模型的分析与检验

针对第六题的第一问，我们尝试对每个区域设置相同的权重，即认为每个区域的出警概率是一致的。那么此时求得的最优设置区域则为H，平均出警距离为10.22.这说明我们的模型对区域出警的概率变化是敏感的。

# 模型的评价、改进与推广

## 模型的优点

1. 两次层次分析法的使用过程中，我们均以误差较小的范围内通过了一致性检验，改模型的运用具有较高的稳定性保障。
2. 第五题中每条曲线的拟合度均在误差小于0.001%的范围内，六号事件的拟合曲线甚至达到1，可见我们拟合的准确程度。
3. 第六题中采用的Dijkstra算法简明，能得到全局最优解。

## 模型的缺点

由于时间有限，对于第一题，我们在使用层次分析法的时候对两组数据加以的权重值较为主观，考虑因素可能稍有不全。

## 模型的改进

可以通过专家或调查问卷法获得更加客观的权重。

## 模型的推广

我们在第六问中使用的最短路径算法可以推广到其他寻路问题。

# 参考文献

附录

|  |
| --- |
| 附录1 |
| 问题1的matlab代码 |
| 1. >> B=[1,9;1/9,1]; 2. [n,m]=size(B); 3. for i=1:n 4. for j=1:n 5. if B(i,j)\*B(j,i)~=1 6. fprintf('i=%d,j=%d,B(i,j)=%d,B(j,i)=%d**\n**',i,j,B(i,j),B(j,i)) 7. end 8. end 9. end 10. [V,D]=eig(B); 11. tz=max(D); 12. tzz=max(tz); 13. c1=find(D(1,:)==max(tz)); 14. tzx=V(:,c1); 15. %权 16. quan=zeros(n,1); 17. for i=1:n 18. quan(i,1)=tzx(i,1)/sum(tzx); 19. end 20. Q=quan; 21. CI=(tzz-n)/(n-1); 22. RI=[0,0,0.58,0.9,1.12,1.24,1.32,1.41,1.45,1.49,1.52,1.54,1.56,1.58,1.59]; 23. CR=CI/RI(1,n); 24. if CR>=0.1 25. fprintf('没有通过一致性检验**\n**'); 26. else 27. fprintf('通过一致性检验**\n**'); 28. end 29. 通过一致性检验 |

|  |
| --- |
| 附录2 |
| 问题6的python代码 |
| 1. **import** numpy **as** np 2. **def** 赋值(): 3. d=np.zeros((15,15)) 4. d[0][1] = d[1][0] = 11.1 5. d[0][3] = d[3][0] = 11.4 6. d[0][12] = d[12][0] = 8.2 7. d[1][2] = d[2][1] = 8.2 8. d[1][3] = d[3][1] = 12.8 9. d[2][3] = d[3][2] = 7.7 10. d[2][4] = d[4][2] = 11.1 11. d[2][5] = d[5][2] = 9.4 12. d[3][5] = d[5][3] = 6.9 13. d[3][13] = d[13][3] = 10 14. d[3][14] = d[14][3] = 8.5 15. d[3][9] = d[9][3] = 12.7 16. d[3][12] = d[12][3] = 14.3 17. d[4][5] = d[5][4] = 7.4 18. d[5][13] = d[13][5] = 11.2 19. d[6][9] = d[9][6] = 12.9 20. d[6][11] = d[11][6] = 14.5 21. d[6][10] = d[10][6] = 13.4 22. d[6][13] = d[13][6] = 10.6 23. d[7][11] = d[11][7] = 12.3 24. d[7][8] = d[8][7] = 9 25. d[9][10] = d[10][9] = 9.5 26. d[9][12] = d[12][9] = 9.6 27. d[9][13] = d[13][9] = 6.9 28. d[9][14] = d[14][9] = 4.2 29. d[10][11] = d[11][10] = 4.4 30. d[10][12] = d[12][10] = 15 31. d[13][14] = d[14][13] = 5.9 32. **for** i **in** range(15): 33. **for** j **in** range(15): 34. **if** d[i][j]==0 **and** i!=j: 35. d[i][j]=999 36. **return** d 37. **def** startwith(start: int, mgraph: list) -> list: 38. passed = [start] 39. nopass = [x **for** x **in** range(len(mgraph)) **if** x != start] 40. dis = mgraph[start] 41. **while** len(nopass): 42. idx = nopass[0] 43. **for** i **in** nopass: 44. **if** dis[i] < dis[idx]: idx = i 45. nopass.remove(idx) 46. passed.append(idx) 47. **for** i **in** nopass: 48. **if** dis[idx] + mgraph[idx][i] < dis[i]: 49. dis[i] = dis[idx] + mgraph[idx][i] 50. *#print(str(idx)+" "+str(i))* 51. *#print(dis[5])* 52. **return** dis 53. d=赋值()*#建立邻居接矩阵* 54. res=np.zeros((15,15)) 55. w=[0.0426,0.0318,0.0250,0.0415,0.0665,0.0415,0.0532,0.0285,0.0198,0.0421,0.0489,0.0274,0.0586,0.0269,0.4456] 56. **for** i **in** range(15): 57. res[i,:]=startwith(i,d) 58. min0=1000000000 59. min0=100000000 60. **for** i **in** range(15): 61. length=0 62. **for** x **in** range(15): 63. length+=w[x]\*min(d[x][i],d[x][9],d[x][13]) 64. **print**(str(i)+" "+str(length)) 65. **if** length<min0: 66. min0=length 67. a=i 68. **print**(a) 69. **print**(min0) 70. **for** i **in** range(15): 71. **for** j **in** range(15): 72. **for** k **in** range(15): 73. **if** i!=j: 74. length=0 75. **for** x **in** range(15): 76. length+=w[x]\*min(d[x][i],d[x][j],d[x][k],d[x][9],d[x][13]) 77. *#print(str(order[i])+" "+str(order[j])+" "+str(length))* 78. **if** length<min0: 79. min0=length 80. a,b,c=i,j,k 81. **print**(a) 82. **print**(b) 83. **print**(c) 84. **print**(min0) 85. min0=10000000000 86. **for** i **in** range(15): 87. **for** j **in** range(15): 88. **if** i!=j: 89. length=0 90. **for** x **in** range(15): 91. length+=w[x]\*min(d[x][i],d[x][9],d[x][13]) 92. length+=w[x]\*min(d[x][i],d[x][j],d[x][9],d[x][13]) 93. *#print(str(order[i])+" "+str(order[j])+" "+str(length))* 94. **if** length<min0: 95. min0=length 96. a,b=i,j 97. **print**(a) 98. **print**(b) 99. **print**(min0) |

|  |
| --- |
| 附录3 |
| 介绍：该代码是某某语言编写的，作用是什么 |
|  |