紧急医疗救护站设置模型

本文讨论紧急医疗救护站设置问题。通过借鉴图论的思想，使用枚举，排序等算法，从人口、距离等角度对所设立站点位置可行性情况进行综合分析，得出最优化结论。

首先，我们对该县地图和该县各乡镇村的人口数据进行初步分析，思考建立站点的基本情况；接着，我们进行了在不同乡镇村建立医疗救护站能过覆盖范围情况的分析，以及对已对有路程数据使用矩阵思想进行处理，将实际问题转化为数学问题，利用图论思想，使用广度最优和深度最优相结合的方式将矩阵中的数据根据是否能够满足已知条件将其转化为三个只含 0（表示不满足条件）和 1（表示满足条件）元素的矩阵；然后对这三个矩阵进行逻辑“与”运算，得到一个满足所有条件的矩阵；最后利用这个矩阵找出建立医疗救护站的可行位置。

在分析最优站点分布位置时，我们采用枚举法，利用计算机在一定范围内穷举全部具有代表性的可能出现的情况，首先根据是否满足条件进行筛选，其次对符合条件的的情况按照它所能到达的乡镇村的数量的多少进行排序，找出最优站点分布位置。

在解决了第一个问题之后，我们依然使用穷举的方法，假设在不同的乡镇村建立救护站，计算它所能覆盖到的人口总数，再次进行排序，找出满足题目要求的救护站点的位置，给出合理结论。

关键词：图论思想，矩阵处理，枚举，排序。

一、问题重述

假设有一个县医疗救护中心有一些救护车为全县服务，如果这些救护车平时都放置在县城，那么当偏远地区病人需要紧急运往医院时就不能及时到达现场。所以需要在各乡镇及村庄建立紧急医疗救护站，分散放置这些救护车，以便尽可能使得全县任何地方有病人时可以在 20 分钟内到达现场。假设一个救护站配备一辆救护车，司机随时值班，需要时即可使用。假设正常路况下救护车车速为 60 公里/小时，该县地图如图 1 所示，图中各边上的数字为路的长度（公里）：

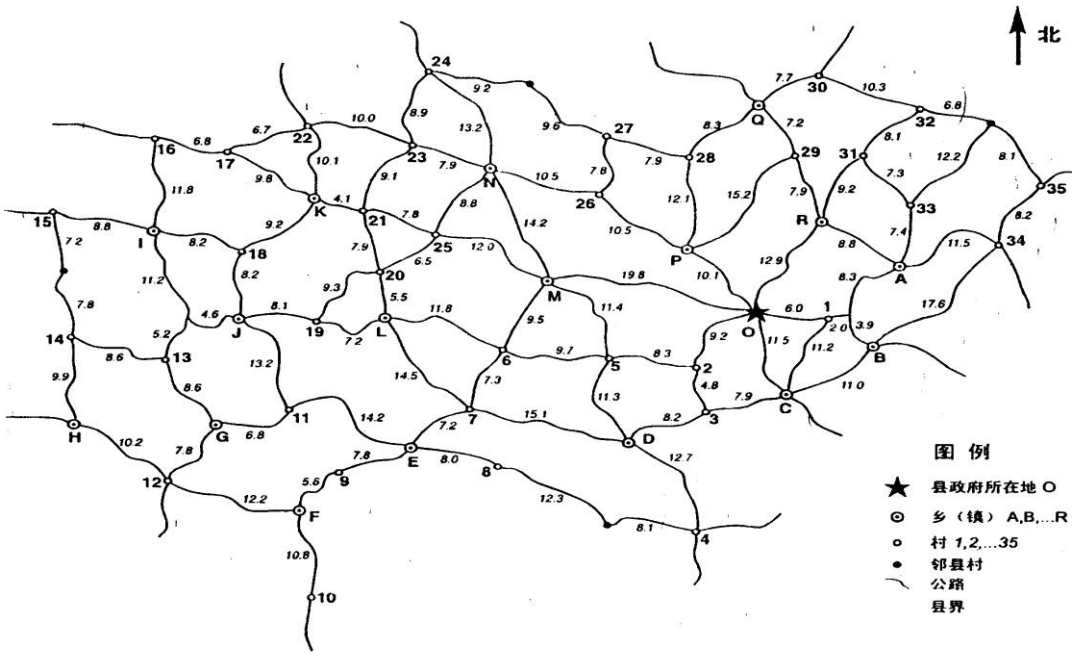

图 1：某县交通图

住在县城、各乡（镇）和村子里的人口数（万人）如表 1-表 2 所示:

表 1：县城及各乡镇人口（万人）

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
0.44	0.52	0.35	0.48	0.37	0.61	0.55	0.25	0.42	0.4	0.63	0.54	0.39	0.52	15	0.43	0.5	0.46

表 2：各村人口（万人）

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0.16	0.22	0.21	0.19	0.24	0.15	0.23	0.25	0.12	0.27	0.35	0.26	0.18	0.23	0.15	0.15	0.25	0.14
19	20	21	22	23	24	25	26	17	28	29	30	31	32	33	34	35	
0.27	0.22	0.19	0.27	0.36	0.11	0.12	0.15	0.27	0.29	0.16	0.29	0.31	0.21	0.17	0.18	0.23	

请建立数学模型解决以下问题：

1. 设救护站可以设在各乡镇及村庄，那么最少需要设置多少救护站可以使得所有人口位于救护车 20 分钟车程能够覆盖的范围内？给出各救护站位置。如果救护站只能位于乡（镇）所在地，结果怎样？

2. 假设共有 10 辆救护车，怎样设置救护站的位置可以使得尽可能多的人口位于救护车 20 分钟车程能够覆盖的范围内？能覆盖所有人口吗？如果不能，多少人不能被覆盖？如果救护车的数量为 7 辆，再次回答上述问题。

二、问题分析

问题要求我们通过建立数学模型，分析计算结果得出紧急医疗救护站设置方法。对于该县交通图和人口数据的综合分析，已经有很多成型的算法，如：多元规划、图论算法、神经网络、穷举法等。究竟采用哪种算法确定医疗救护站位置是我们需要解决的第一个问题。经过多次计算分析，我们发现将图论算法、穷举法和排序算法结合使用更能得出清晰准确的结果，进而得到较为精确的结论。

第一问要求我们设置最少的救护站并使救护车在 20 分钟车程内覆盖全县范围，我们对该县地图进行分析，建立了一阶路段 F、二阶路段 S、三阶路段 T 矩阵（n 阶路段表示从某个乡村镇到另一个乡村镇的路线，在该路线上，两个乡村

镇之间有 n-1 个本县的乡村镇）。在相应程序的帮助下，得出矩阵详细的参数。接着，将三个矩阵进行逻辑“与”运算，得出一个新矩阵 P，用于表示若在任一乡村镇设立救护站，能够在约束条件下从该点所能到达的全部地点。然后对可能的组合情况进行枚举，根据枚举结果进行排序，得出在建立最少救护站时覆盖全县的合理方案。

第二问要求在救护车一定的情况下所能覆盖的最大人口。我们可以借助第一问得出的 P 矩阵，将不同乡镇村的人口数量考虑进去，通过枚举得出各种情况，减去重叠人口即为所能覆盖的人口，并根据已知条件对其排序，然后进行比较得出最优建立救护站方案。

通过综合各方面因素建立数学模型并对其求解分析，我们得到了一个建立救护站的合理方案。

1.基本假设

1.救护站所能覆盖的乡镇村不会同时出现需要急救的情况。
2.路况畅通，救护车可以按给定速度行驶。
3.不存在人为干扰因素，司机可以及时出勤，医疗电话通信畅通。
4.不会出现突发自然状况，影响救护车的运行。
5.将乡村镇不考虑面积大小，试做点来处理。

2.基本符号说明

1.F———一阶路段矩阵。
2.S———二阶路段矩阵。
3.T———三阶路段矩阵。
4.P———站点覆盖矩阵。P = F & S & T。
5.P[ij]=1 表示在约束条件下 i 点能到达 j 点。
P[ij]=0 表示在约束条件下 i 点不能到达 j 点。
6.Zi——表示第 i 个集合。
7.Wi——表示第 i 个集合中的元素由大到小的排序。

三、最少紧急医疗救护站设置模型

3.1 模型的建立与求解

3.1.1 初步分析数据

通过对该县交通地图和题意的初步分析，我们知道了紧急医疗救护站位置和它覆盖的村庄之间的距离应该小于等于 20km，接着我们统计了从任意一个乡村镇出发，只走一段路可以到达的其它乡村镇以及它与其它每一乡镇村之间的距离，然后我们设计了一个算法（见附录程序 1），通过计算机找出从任意一个乡村镇出发，它可以覆盖到的其它乡村镇，将乡村镇全部不考虑大小，全部当做点处理，通过对程序运行结果的分析统计和数据处理，我们得到了三个矩阵，分别是一阶路段矩阵 F，二阶路段矩阵 S，三阶路段矩阵 T，它们分别是：

$$F=\begin{bmatrix} F11 & F12 & F13 \\ F21 & F22 & F23 \\ F31 & F32 & F33 \end{bmatrix},\ S=\begin{bmatrix} S11 & S12 & S13 \\ S21 & S22 & S23 \\ S31 & S32 & S33 \end{bmatrix},\ T=\begin{bmatrix} T11 & T12 & T13 \\ T21 & T22 & T23 \\ T31 & T32 & T33 \end{bmatrix}$$

其中 F，S，T 是三个 53 行 53 列的只含 0 和 1 的矩阵，0 表示从这一点出发不能到达另一点，1 表示从这一点出发能够到达另一点，F 表示两点之间只有一段路，S 表示两点之间有两段路，T 表示两点之间有三段路；其中 F11，S11，T11 均是 18×18 的矩阵，表示从 A~R 通往 A~R 的情况；F12，S12，T12 均是 18×17 的矩阵，表示从 A~R 通往 1~17 的情况；F13，S13，T13 均是 18×18 的矩阵，表示从 A~R 通往 18~35 的情况；F21，S21，T21 分别是 17×18 的矩阵，表示从 1~17 通往 A~R 的情况；F22，S22，T22 均是 17×17 的矩阵，表示从 1~17 通往 1~17 的情况；F23，S23，T23 均是 17×18 的矩阵，表示从 1~17 通往 18~35 的情况；F31，S31，T31 分别是 18×18 的矩阵，表示从 18~35 通往 A~R 的情况；F32，S32，T32 均是 18×17 的矩阵，表示从 18~35 通往 1~17 的情况；F33，S33，T33 均是 18×18 的矩阵，表示从 18~35 通往 18~35 的情况（F11，S11，T11，F12，S12，T12，F13，S13，T13，F21，S21，T21，F22，S22，T22，F23，S23，T23，F31，S31，T31，F32，S32，T32，F33，S33，T33 的详细参数结果来源于附录结果 1）（P 矩阵的详细表示参见附表 1）。

3.1.2 图论算法与优化算法运行及其结果讨论

3.1.2.1 求解最少紧急医疗站模型的建立

我们利用图论思想和优化算法设计了一个程序，它的主要功能是通过计算 3.1.1 中分析得到的矩阵数据，从而找出满足能够覆盖全县的所有医疗救护站位置的不同集合的方案，然后依据每个集合的势对集合从大到小排序，那么所得到的大小关系中，势最小的那个集合就是所求结果，其中元素便是可以建立紧急医疗救护站的位置。

具体做法就是将 F，S，T 矩阵使用 Matlab 软件按逻辑或操作合成一个矩阵 P，P=F | S| T；P 是一个 53×53 的矩阵，这个矩阵表示从任意一点出发在不超过 20km 的范围内，它所能到达的其它点（不管它需要走几段路，只要能到就行），利用线性代数知识，P 可以看成是一个由 53 个 53 维的行向量组成的向量组，每一个向量表示从这一点出发到其它所有点的情况（能到达记为 1，不能到达记为 0），拿出任意一个向量然后寻找其它向量进行加法运算，要找的这个向量是要在经过加法运算后，得到的那个向量的不同维度出现尽可能多的 1，即使得第一个向量为 0 的维度得到补充，预期结果是能够在遍历所有组合的方式之后得到多个由不同的原始向量组合之后得到的全 1 向量，这样就可以得到许多不同的组合方式，将每一种组合方式定义成一个集合，将参与组合的向量作为集合中的元素，然后根据每个集合的势的大小，将其进行排序，势最小的那个集合就是建立医疗救护站的最佳方式，其内元素就是最佳站点位置。

最后我们利用 C 源程序（见附录程序 2）实现了这个算法，程序运行得到了结果（详细结果见附录结果 2）；紧接着我们利用数学统计的方法对所得到的结果进行了分析。

3.1.2.2 图论和优化算法讨论

在得到了程序的运行结果之后，我们综合 3.1.1 中得到的数据以及题目中所给的交通图就程序运行结果的可靠性和准确性进行讨论。在设计算法的过程中，我们借鉴了图论算法中加权的有向图来计算给定距离长度的两个对象之间的路径可行性情况；模拟了计算机的运行情况，同时进行了复杂度计算，确定存在

这样一个递归函数，它能够在 53 个向量之中先确定某几个向量，然后在遍历剩下的其余所有向量，得到一个或几个符合所给条件的向量时，就跳出递归循环，然后开始下一次的递归，如此进行，直到遍历所有情况，就会得到所有符合情况的可行解；然后对所有可行解运用集合关系建立集合，计算集合的势，根据势由大到小的结果进行人工挑选，便可以得到最优结果。在查阅了一些关于图论算法的资料后，我们得出结果，我们设计的算法是可行的，而且程序运行得到的结果也是较为可靠的。

3.2 结果分析

对于程序运行结果得到的大量数据经过我们人工整理筛减以后，得到如下建站方案（枚举了只建立 8 个站,9 个站的所有方案，对于建立 10 个站，11 个站的方案，各自列举了 5 种，对于建立 12 个站以上的方案没有进行统计和列举）（使用集合表示，并按照集合势的大小对其排了序）：

集合势为 8：

Z1={A,D,9，13，18，20，27，Q}

Z2={A,D,9，13，18，20，27，30}

Z3={A,D,9，13，18，20，27，31}

Z4={A,D,9，13，18，20，27，32}

集合势为 9：

Z5={A,D,N,P,Q,7,10,13,18}

Z6={A,D,N,P,Q,7,F,,13,18}

Z7={A,D,N,P,Q,7,,9,13,18}

Z8={A,D,N,P,Q,6,10,13,18}

Z9={A,D,N,P,Q,6,F,,13,18}

Z10={A,D,N,P,Q,6,,9,13,18}

Z11={A,D,N,O,Q,7,10,13,18}

Z12={A,D,N,O,Q,7,F,,13,18}

Z13={A,D,N,O,Q,7,,9,13,18}

Z14={A,D,N,O,Q,6,10,13,18}

Z15={A,D,N,O,Q,6,F,,13,18}

Z16={A,D,N,O,Q,6,,9,13,18}

集合势为 10：

Z18={A,Q,P,N,D,E,19，10，I,H}

Z19={A,Q,P,N,D,E,L，10，I,H}

Z20={A,Q,P,N,D,E,19，F，I,H}

Z21={A,Q,P,N,D,E,19，9，I,H}

Z22={A,P,N,D,E,19，10,30，I,H}

集合势为 11：

Z23={A,D,E,H,I,P,Q,N,M,10，19}

Z24={A,D,E,H,I,P,Q,N,L,10，19}

Z25={A,D,E,H,I,P,Q,N,O,10，19}

Z26={A,D,E,H,I,P,Q,N,6,10，19}

Z27={A,D,E,H,I,P,Q,N,5，10，19}

容易看出当救护站能够设置在乡镇和村庄时，最少只需要设置 8 个紧急救护站就

可以覆盖全县范围，站点的位置分别是：
第一种方案：设在 A,D,9，13，18，20，27，Q 这几个乡村镇；
第二种方案：设在 A,D,9，13，18，20，27，31 这几个乡村镇；
第三种方案：设在 A,D,9，13，18，20，27，32 这几个乡村镇；
第四种方案：设在 A,D,9，13，18，20，27，33 这几个乡村镇；
当救护站只能设在乡镇时，从上述的集合之中可以这样分析：将势为 8 的集合中的所有村庄所覆盖的位置用其余乡镇去覆盖，看能否覆盖完全且不改变集合的势，对势为 9，为 10，的所有集合都这样做，最终可以得出结论：当紧急医疗站只能建立在乡镇时，至少需要建立 10 个医疗站点才可以覆盖全县，列举其中两种可行方案如下：
第一种方案：设在 A,D,E,F,G,I,LN,P,Q 这几个乡镇；
第二种方案：设在 A,D,E,F,H,I,LN,P,Q 这几个乡镇；
然后利用 Matalab 将得到的站点位置全部标记在所给地图上进行验证，发现找到的位置如果建立紧急医疗站可以覆盖到全县范围；接着将表示所找到的站点位置的向量进行相加最后得到了一个全 1 向量（详细情况参见附表 2），这也说明所选的位置是十分合理的。

3.3 模型的评估与改进方向

我们的模型可以说明在何处建立紧急医疗救护站，建站数量可以最少；我们看到题后明白它是一个选址问题，我们想到了图论算法，我们进行了要素的取舍并且建立了合理的理论体系，利用矩阵思想将实际问题完全转化为一个数学问题；利用最优化原理，我们利用排序的思想对所得的可能情况进行排序，很快便找出了最优建立紧急医疗救护站的方式。

在建立模型的整个过程中，我们始终以图论和最优化做为指导思想，进行模型建立和算法的设计，主要利用矩阵的思想处理原始数据，在得到程序的运行结果之后，利用 excel 表格进行了方案验证，以确保结果准确合理。

虽然在整个过程之中，我们的思想方法和所建立的数学模型都没有太大问题，但是总体来说仍旧存在着诸多不足，首先在分析数据时，程序 1 运行得到的结果较为凌乱和繁杂，给统计造成了很大的难度，这种情况下，由于人为因素极有可能产生随机误差；其次，设计的算法并没有得到最大优化，以至于程序在运行时耗时较大，而且结果凌乱，计算方案也有一些粗糙。但最终得到的方案在经过人工验证之后可以肯定是较为合理可靠的。

综合来看我们的模型仍需改进，以期得到更加完美的方案。

四、站点数量固定，可以覆盖最多人口的站点设置模型

4.1 模型的建立与求解

4.1.1 人口数据分析

我们先对题目之中所给的表 1（县城及各乡镇人口数据）和表 2（村庄人口数据）进行分析，发现该县的人口分布存在着如下几个特点：①县城及各乡镇人口数量大约都在 0.35 万人以上，只有 H 乡的人口数量是 0.25 万；②村庄的人口数量大约都在 0.35 万以下，更多的集中分布在 0.25 万左右，只有 23 村的人口数量是 0.36 万人；③村庄人口总数少于乡镇人口总数；④各乡镇人口数量明显多于各村庄人口数量；⑤各乡镇和村庄人口分布都较为均匀；然后我们对该县位

置图进行了分析，推测如果在像 A 或者 K 这样连接更多乡村镇的地点建立医疗救护站，可以覆盖到更多的人；

4.1.2 站点数量一定，可以覆盖最多人口数量模型的建立

在本题的第二个问题之中，问到了两种情况，第一问：当救护车的数量为 10 辆时，能否做到覆盖全县；第二问：当救护车的数量为 7 辆时，能否做到覆盖全县，如果不能，怎样安排站点位置，才能覆盖更多的人口。

显然，根据我们建立的最少紧急医疗站建立模型，我们知道，要想覆盖全县，至少需要建立 8 个紧急医疗站，所以，对于第二题第一问，我们可以得到准确结果，当救护车的数量为 10 辆时，我们通过计算发现建站在 A,D,E,F,N,Q 这六个乡镇是必须的；按照集合势为 10 的所有组合都可以覆盖全县所有人口，例如可以有如下安排方式：

第一种方案：设在 A,D,E,F,G,I,LN,P,Q 这几个乡镇；

第二种方案：设在 A,D,E,F,H,I,LN,P,Q 这几个乡镇；

当救护车的数量为 7 辆时，根据我们建立的最少紧急医疗站模型可知，如果要覆盖全县，至少需要 8 辆救护车，所以 7 辆车显然是无法覆盖全县的，因此我们设计了如下算法，当 7 辆车时，能够覆盖到尽可能多的人口；具体算法如下:根据最少医疗救护站模型中确立的建立最少紧急医疗站的四种方案，将四个势为 8 的集合每个之中的个元素分别所能覆盖的乡村镇的总人口计算出来，然后分别进行排序，选出来覆盖人数最多的 7 个站点位置，当然要综合考虑，不能出现重叠覆盖某一地区人口的情况；根据这一算法我们得到了一些数据结果。

4.2 结果分析

根据上述算法得到了如下数据结果：

A 点覆盖的乡村镇是 A,B,O,R,1,35,34,33,31,29 覆盖的总人口是 17.63 万人；

D 点覆盖的乡村镇是 C,D,2,3,4,5,7 覆盖的总人口是 1.92 万人；

Q 点覆盖的乡村镇是 Q,R,32,30,29,28,27 覆盖的总人口是 2.18 万人；

9 点覆盖的乡村镇是 E,F,7,8,9,10,12 覆盖的总人口是 2.11 万人；

13 点覆盖的乡村镇是 G,H,I,J,11,12,13,14,18,19 覆盖的总人口是 3.05 万人；

18 点覆盖的乡村镇是 I,J,K,13,15,16,17,18,19,21,22 覆盖的总人口是 3.05 万人；

20 点覆盖的乡村镇是 J,K,L,M,N,6,7,21,20,19,23,25 覆盖的总人口是 4.02 万人；

27 点覆盖的乡村镇是 L,N,P,Q,28,27,26,24 覆盖的总人口是 2.81 万人；

30 点覆盖的乡村镇是 Q,28,29,30,31,38 覆盖的总人口是 1.76 万人;

31 点覆盖的乡村镇是 A,R,33,32,31,30,29 覆盖的总人口是 2.08 万人;

32 点覆盖的乡村镇是 Q,R,35,33,32,31,30 覆盖的总人口是 2.17 万人;

然后进行对每个集合涉及的八个元素进行 4 组排序，分别为：

W1：A,20,13,18,27,Q,9，D

W2：A,20,13,18,27,9，D,30

W3：A,20,13,18,27,9,31，D

W4：A,20,13,18,27,32，9，D

容易知道：Q 和 9 覆盖的总人口是 4.29 万人；

9 和 D 覆盖的总人口是 4.03 万人；

9 和 31 覆盖的总人口数是 4.19 万人；

9 和 32 覆盖的总人口数是 4.28 万人；

根据上述数据结果并对其进行分析，得到了如下建立站点的方案：当只有 7 辆救护车时，应该把紧急医疗救护站建立在 A,20,13,18,27,Q,9 这几个乡村镇，就能够保证覆盖的人数达到最多。

4.3 模型评估与该进方向

在建立这个模型的过程中，我们更多的采用了从建立最少紧急医疗救护站模型之中得到的数据，因此整个模型的合理性与正确性在很大程度上都过分依赖于最少紧急医疗站救护模型；其次计算过程有些粗糙，还存在着极大的改进空间。

虽然模型有很多不足之处，但是计算的结果还是相对准确的，而且那个计算能够覆盖最多人口的算法具有更加广泛的适用性，可以应用到其他模型之中去。

综合来看我们的模型仍需改进，以期得到更加完美的方案。

五、结论

根据建立的模型可知：

1，救护站可以设在各乡镇及村庄，那么最少需要设置 8 个救护站可以使得所有人口位于救护车 20 分钟车程能够覆盖的范围内，救护站点设置的位置方案有四种，分别是：①：设置在 A,D,9，13，18，20，27，Q 这几个乡村镇；②：设置在 A,D,9，13，18，20，27，31 这几个乡村镇；③：设置在 A,D,9，13，18，20，27，32 这几个乡村镇；④设置在 A,D,9，13，18，20，27，33 这几个乡村镇。如果救护站只能位于乡镇所在地，那么至少需要设置 10 个救护站可以使得所有人口位于救护车 20 分钟车程能够覆盖的范围内，救护站设置的位置方案有两种：①：设在 A,D,E,F,G,I,LN,P,Q 这几个乡镇；②：设在 A,D,E,F,H,I,LN,P,Q 这几个乡镇。

2，当有 10 辆救护车时，完全能够覆盖全县所有人口，站点的设置方案也有很多种，可以再任意一种只设置 8 个救护站的方案上任意添加两个站，现在给出任意 5 种方案：①：设置在 A,Q,P,N,D,E,19，10，I,H；②：A,Q,P,N,D,E,L，10，I,H③：设置在 A,Q,P,N,D,E,19，F，I,H；④：设置在 A,Q,P,N,D,E,19，9，I,H；⑤：设置在 A,P,N,D,E,19，10,30，I,H。当救护车只有 7 辆时，无法覆盖全县所有人口，将站点设置在 A,20,13,18,27,Q,9 这几个乡村镇能够保证覆盖到更多的人口。

六、参考文献

【1】李明哲，金俊等编著的《图论及其算法》，机械工业出版社。

【2】刘振宏，马绍汉编著的《离散最优化算法》，科学出版社。

附录

附录程序 1（C 源程序）：

```
#include<stdio.h>
#define  NMAX  200
#define  CITY  53
int  main(){
int  s[CITY][CITY],i,j,k,l,m,n,a,b,c;
```

```
char  p[CITY][3]={"A","B","C","D","E","F","G","H","I","J","K","L","M"
,"N","O","P","Q","R"};//,"1","2","3","4","5","6","7","8","9","10","11
","12","13","14","15","16","17","18","19","20","21","22","23","24","2
5","26","27","28","29","30","31","32","33","34","35"};
for(i=18,j=0;i<27;i++){
p[i][0]='0';
p[i][1]='1'+j++;
}
for(j=0;i<37;i++){
p[i][0]='1';
p[i][1]='0'+j++;
}
for(j=0;i<47;i++){
p[i][0]='2';
p[i][1]='0'+j++;
}
for(j=0;i<CITY;i++){
p[i][0]='3';
p[i][1]='0'+j++;
}
for(i=0;i<CITY;i++){
for(j=0;j<CITY;j++){
scanf("%d",&s[i][j]);
}
}
for(i=0;i<CITY;i++){
for(j=0;j<CITY;j++){
a=s[i][j];
if(a&&a<=NMAX){
printf("%s->%s    ==    %d\n",p[i],p[j],a);
m=j;
for(n=0;n<CITY;n++){
if(n==i)continue;
b=a+s[m][n];
if(b-a&&b<=NMAX){
printf("%s->%s->%s    ==    %d\n",p[i],p[j],p[n],b);
k=n;
for(l=0;l<CITY;l++){
if(l==m)continue;
c=b+s[k][l];
if(c-b&&c<=NMAX){
printf("%s->%s->%s->%s    ==    %d\n",p[i],p[j],p[n],p[l],c);
}
}
```

```
}
}
}
}

}
}
```

附录结果 1（程序 1 运行结果）：

程序运行结果.txt

附表 1：

P矩阵.txt

附录程序 2：

附录程序 2（C 源程序）：

```
#include<stdio.h>

#define NMAX 200

#define CITY 53

char

p[CITY][3]={"A","B","C","D","E","F","G","H","I","J","K","L","M","N"

,"O","P","Q","R"};

int q[CITY][CITY]={0};

int a[CITY]={0},b1[20],b2[20];

int all();

int t=0,flag,err=0;

int main(){

    int s[CITY][CITY],i,j,k,l,m,n,ai,b,c,t,flag,end[10000][20],a0[CITY];

    //,"1","2","3","4","5","6","7","8","9","10","11","12","13","14","15","
```

```
16","17","18","19","20","21","22","23","24","25","26","27","28","29","3
0","31","32","33","34","35"};
    for(i=18,j=0;i<27;i++){
        p[i][0]='0';
        p[i][1]='1'+j++;
    }
    for(j=0;i<37;i++){
        p[i][0]='1';
        p[i][1]='0'+j++;
    }
    for(j=0;i<47;i++){
        p[i][0]='2';
        p[i][1]='0'+j++;
    }
    for(j=0;i<CITY;i++){
        p[i][0]='3';
        p[i][1]='0'+j++;
    }
    for(i=0;i<CITY;i++){
        for(j=0;j<CITY;j++){
            scanf("%d",&s[i][j]);
        }
```

```
    }
    for(i=0;i<CITY;i++){
        for(j=0;j<CITY;j++){
            ai=s[i][j];
            if(ai&&ai<=NMAX){
                q[i][i]=1;
                q[i][j]=1;
                //printf("%s->%s    ==    %d\n",p[i],p[j],ai);
                m=j;
                for(n=0;n<CITY;n++){
                    if(n==i)continue;
                    b=ai+s[m][n];
                    if(b-ai&&b<=NMAX){
                        q[i][n]=1;
                        //printf("%s->%s->%s
==    %d\n",p[i],p[j],p[n],b);
                        k=n;
                        for(l=0;l<CITY;l++){
                            if(l==m)continue;
                            c=b+s[k][l];
                            if(c-b&&c<=NMAX){
                                q[i][l]=1;
```

```
                                    //printf("%s->%s->%s->%s
==    %d\n",p[i],p[j],p[n],p[l],c);
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
        for(j=0;j<CITY;j++){//读 a
            a[j]=q[0][j];
            a0[j]=q[0][j];
        }
        for(j=0;j<CITY;j++){
            printf("[%d]%s ",t,p[j]);
            all();
        }
}

int all(){
    int i,j,k,n=0;
    for(i=0;a[i]!=0&&i<CITY;i++);//i 为 a 空位
```

```
for(k=0;k<CITY;k++){
    if(i==CITY){
        for(j=0;j<CITY;j++){
            n+=a[j];
        }
        printf(" == %d\n",n);
        for(j=0;j<20;j++){
            b1[j]=b2[j];
        }
        t--;
        return 0;
    }
    if(q[k][i]!=0){
        b2[t]=k;
        t++;
        for(j=0;j<CITY;j++){
            a[j]+=q[k][j];
        }
        printf("[%d]%s ",t,p[k]);
        for(j=0;j<t-1;j++){
            if(b1[j]==k){
                printf("ERROR");
```

```
                flag=j;err=1;
                t--;
                return 0;
            }
        }
        all();
        for(j=0;j<CITY;j++){
            a[j]-=q[k][j];
        }
        if(t!=flag&&err){
            return 0;
        }
        err=0;
    }
}
t--;
return 0;
}
```

附录结果 2（程序 2 的执行结果）：

结果.txt

附表 2：


8方案验证过程.txt ，

9方案验证过程.txt