基于广度优先算法的钢管订购和运输

【摘要】

本文对钢管的订购和运输进行研究，根据各钢厂钢管的生产数量、铁路和公路的运输费用规则、各主管道铺设的钢管数量等因素，以总费用最小为目标，建立优化模型。

首先，本文考虑到求铺设管道的花费包括：钢管从钢管厂运输到铺设点费用、从钢管厂购买钢管费用、铺设铁路时铺设点间运输费用。结合题目给出的钢厂产能限制、运输收费规则、路线图等条件，可以求出以上三个类别费用，从而求出铺设管道的总花费。

其次，为了求出铺设管道总花费最小，需要通过**二次优化**，先求出管道从钢管厂运输到铺设点的最少运费。根据题目钢管需要从钢厂运输到铺设地点，可以通过铁路或公路进行运输。有单位钢管的铁路运价表和公路每单位钢管运价再结合题目所给路线图，我们将路线图进行标记点处理，共计 39 个点位，转换成数据结构内的图，采用 **BFS 算法**求解各钢管厂运输到各铺设点的最少运费及运输路径。最终结果为 7*15 的钢厂到铺设点的最少运费矩阵，具体结果见正文矩阵。

最终目标是找到一种订购和运输计划，使总费用最小化。已经求出了各钢管从钢管厂运输到铺设点最少费用，再求出各钢管厂运输到各铺设点的钢管数量。本文确定可以从铺设点向两端铺设钢管的铺设方案，结合钢厂产能，路径图等条件约束建立钢厂是否运出钢管的“**0-1”变量**数学模型。利用 **Lingo 求解**，得从 S1 钢铁厂订购 800 个单位的钢管，从 S2 钢铁厂订购 800 个单位的钢管，从 S3 钢铁厂订购 1000 个单位的钢管,从 S5 钢铁厂订购 1352 个单位的钢管，从 S6 钢铁厂订购 1219 个单位的钢管，从 S4 和 S7 钢铁厂订购的钢管量为 0，最小总费用为 1278632 万元。详细方案见表 3 到表 8。

综上所述，主管道钢管订购和运输问题需要综合考虑钢厂产能、销售价、运输费用等因素，并通过建立数学模型和优化求解算法，找到一个最优的订购和运输方案，以实现总费用的最小化，利用 LINGO 求解。这可以帮助企业在满足需求的同时降低成本，提高效益。

关键词：订购和运输计划 最优化模型 BFS 算法

一、问题重述

1.1 引言

近年来，在国家大力发展建设的促进下，全国各地发展迅速，为了更好地输送天然气，需要对输送天然气的主管道进行铺设，因钢厂离铺设点存在一定的距离，如何制定合理的主管道钢管订购和运输计划使得总费用最小成为急需解决的问题之一。

1.2 问题提出

根据附件中给出的题目，结合所给条件和相关的数据，建立数学模型，解决以下问题：某地需对 A_1 到 A_{15} 这条路上铺设管道，经筛选后该地周围共有 7 个钢厂可向 15 个铺设点运送钢管，据此以总费用最小为目标制定一个主管道钢管的订购和运输计划。

二、问题分析

本题主要对钢管的订购和运输进行研究。需满足以总费用最小为目标，制定主管道钢管的订购和运输计划。

2.1 问题分析

在满足钢厂生产条件和铁路、公路运输条件的情况下，以总费用最小为目标，制定主管道钢管的订购和运输计划，需要从以下几个方面进行分析：

① 铺设管道的流程：铺设天然气运输管道分为 2 步，首先要将管道从钢厂运输到铺设点，然后再铺设点展开铺设管道工作。可以分析出铺设管道的总花费包括：钢管从钢管厂运输到铺设点费用、从钢管厂购买钢管费用、铺设铁路时铺设点间运输费用。故此求解以上三部分和的最小值即可求出总费用的最优解。

② 钢管从钢管厂运输到铺设点的最小费用：根据题目给出的铁路运输价格表和规则，以及公路的运输价格和路线规划图等信息，采用最优化算法求解。

③ 确定铺设方案：当钢管运输到铺设点时，钢管在铺设过程中还需要运输，采用不同的铺设方案，影响铺设的效率与花费。本文采用从铺设点向两端进行铺设管道。

④ 约束条件：本文的约束条件有钢厂的产能，及路线图所给出的点点间距离。

结合以上几点，我们建立得出求解该问题的步骤，具体步骤流程图如下：

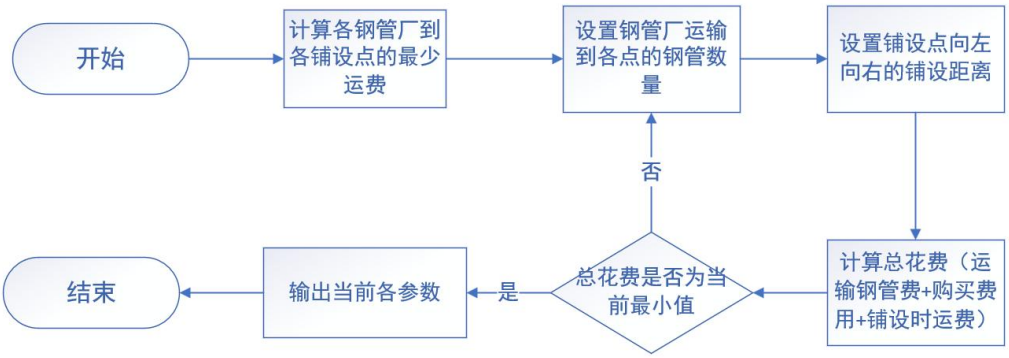

图 1 问题分析流程图

三、符号说明

符号	描述说明
x_{ij}	第i厂运输到第j点的钢管数量
a_{ij}	每单位钢管从第i厂运输到第j点的运费
p_i	第i厂销价 1 单位钢管的售价
s_i	在第i厂购买的钢管数量
L_j	由第j点向左侧铺的长度
R_j	由第j点向右侧铺的长度
b_k	第k段路段的长度
f_i	“0-1”变量

四、模型假设

1、假设钢管价格不会随市场行情波动。

2、假设运输和铺设途中不会造成钢管的浪费

3、假设钢厂生产的钢管长度精确无误，质量过关。

五、模型建立与求解

本题主要对钢管的订购和运输进行制定。根据附件中给出的条件和数据，以总费用最小为目标，建立数学优化模型。

5.1 问题 ：主管道钢管的订购和运输计划

需满足各钢厂钢管的生产数量、铁路和公路的运输费用规则、各铺设点

铺设的钢管数量等条件下，以总费用最小为目标，制定主管道钢管的订购和运输计划。

5.1.1 各钢厂到各铺设点运输 1km 的最少费用矩阵

为了能制定最优的主管道钢管的订购和运输计划，根据附件中已知的钢厂生产、铁路和公路运输的价格等条件，进行编程求解出各钢厂到各铺设点运输 1 km 钢管的最少费用。

（1）单位钢管铁路运价

根据附件中给出的 1 单位钢管的铁路运价表，对表中的数据进行研究，可求解出单位钢管铁路运价 $f(x)$ 关于里程 x 的分段函数如下所示：

$$f(x)=\begin{cases}20 & ,x\le 300\\ 20+3\left\lceil\dfrac{x-300}{50}\right\rceil & ,301\le x\le 500\\ 37 & ,501\le x\le 600\\ 44 & ,601\le x\le 700\\ 50 & ,701\le x\le 800\\ 50+5\left\lceil\dfrac{x-800}{100}\right\rceil & ,801\le x\le 1000\\ 60+5\left\lceil\dfrac{x-1000}{100}\right\rceil & ,x>1000\end{cases}$$

（2）各钢厂到各铺设点运输 1 km 的最少费用矩阵

根据以上的单位钢管铁路运价，为了方便数据进行计算，编程求解各钢厂到各铺设点运输 1 km 钢管的最少费用矩阵 A 表示为：

$$A=\begin{bmatrix}170.7 & 160.3 & 140.2 & 98.6 & 38 & 20.5 & 3.1 & 21.2 & 64.2 & 92 & 96 & 106 & 121.2 & 128 & 142\\ 215.7 & 205.3 & 190.2 & 171.6 & 111 & 95.5 & 86 & 71.2 & 114.2 & 142 & 146 & 156 & 171.2 & 178 & 192\\ 230.7 & 220.3 & 200.2 & 181.6 & 121 & 105.5 & 96 & 86.2 & 48.2 & 82 & 86 & 96 & 111.2 & 118 & 132\\ 260.7 & 250.3 & 235.2 & 216.6 & 156 & 140.5 & 131 & 116.2 & 84.2 & 62 & 51 & 61 & 76.2 & 83 & 97\\ 255.7 & 245.3 & 225.2 & 206.6 & 146 & 130.5 & 121 & 111.2 & 79.2 & 57 & 33 & 51 & 71.2 & 73 & 87\\ 265.7 & 255.3 & 235.2 & 216.6 & 156 & 140.5 & 131 & 121.2 & 84.2 & 62 & 51 & 45 & 26.2 & 11 & 28\\ 275.7 & 265.3 & 245.2 & 226.6 & 166 & 150.5 & 141 & 131.2 & 99.2 & 76 & 66 & 56 & 38.2 & 26 & 2\end{bmatrix}$$

5.1.2 确定目标

根据题目要求，为了从 A_1 到 A_{15} 铺设一条主管道，并且使得整个过程总费用最小。设 x_{ij} 为第 i 厂运输到第 j 点的钢管数量，a_{ij} 为每单位钢管从第 i 厂运输到第 j 点的运费，p_i 为第 i 厂销价 1 单位钢管的售价，L_j 为由第 j 点向左侧铺的长度，R_j 为由第 j 点向右侧铺的长度，以满足各钢厂钢管的生产数量、

铁路和公路的运输费用规则、各铺设点铺设的钢管数量的情况下，以整个过程的总费用最小为目标，可确定目标如下：

$$Min=\sum_{i=1}^{7}\sum_{j=1}^{15}x_{ij}a_{ij}+\sum_{i=1}^{7}\sum_{j=1}^{15}x_{ij}p_i+0.1\sum_{j=1}^{15}\frac{\left(L_j\left(L_j+1\right)+R_j\left(R_j+1\right)\right)}{2}$$

5.1.3 约束条件

为了满足本体对钢管订购和运输的基本要求，需对各个条件进行如下约束：

（1）“0-1”变量的约束

根据钢厂生产数量的限制条件，钢厂是否进行生产钢管，可设 f_j 进行“0-1”约束，当 f_i 等于 1 时表示第 i 个钢厂生产铺设点所需的钢管，当 f_i 等于 0 时表示第 i 个钢厂不进行生产，具体约束如下：

$$f_i\in\left(0,1\right)\qquad,\left(i=1,2,...,7\right)$$

（2）钢厂生产钢管数量的约束

根据钢厂生产数量要求的规则，每个钢厂在指定期限内需至少生产 500 个单位，且每个钢厂生产的钢厂数量不能超过生产的最大数量，设 x_{ij} 为第 i 厂运输到第 j 点的钢管数量，s_i 为第 i 厂在指定期限内能生产钢管的最大数量，具体约束如下：

$$500f_i\le\sum_{j=1}^{15}x_{ij}\le s_if_i\ ,\left(i=1,2,...,7\right)$$

（3）铺设点铺设钢管数量的分配

从钢厂运输到铺设点的钢管数量分为向左铺设的数量 L_i 和向右铺设的数量 R_j，且两个铺设点之间铺设的钢管总数量 $b_{(j,j+1)}$ 应为铺设点 $j+1$ 向左铺设的数量 L_{j+1} 与铺设点 j 向右铺设的数量 R_j 之和，具体如下：

$$L_j+R_j=\sum_{i=1}^{7}x_{ij}\qquad,\left(j=1,2,...,15\right)$$
$$L_{j+1}+R_j=b_{(j,j+1)}\qquad,\left(j=1,2,...,14\right)$$

（4）铺设费用计算

根据附件所知，沿管道运输的管道铺设费用与公路运输收费规则一致，但以铺设点为起点进行铺设，且不足整公里部分按公里计算，故从铺设点出发每铺设 1 km，所运输的钢管数量应相应减 1，符合等差数列，即：

$$\alpha=0.1\sum_{j=1}^{15}\frac{\left(L_j\left(L_j+1\right)+R_j\left(R_j+1\right)\right)}{2}$$

具体分析图如下：

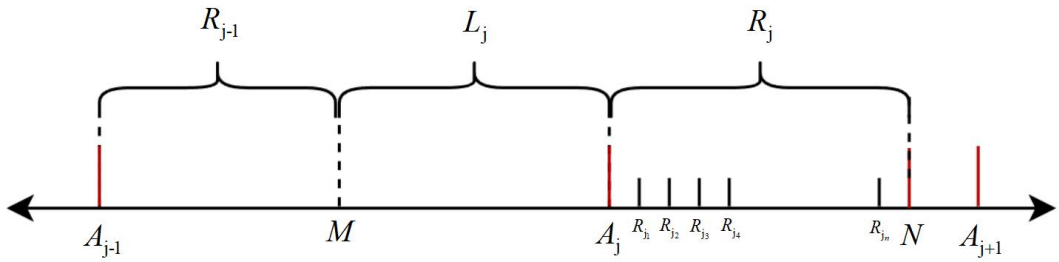

图 2 铺设点铺设费用分析图

A_j 到 N 点的距离为 R_j，从 A_j 点铺设管道到 N 点的花费为：

$$\phi = 0.1*(\mathrm{R_j}+\mathrm{R_j}-1+\mathrm{R_j}-2+\ldots+2+1)=(R_j+1)*\mathrm{R_j}/2$$

5.1.4 总费用最小模型的建立与求解

综合以上目标模型和约束条件分析，以求解总费用最小为目标，建立优化模型，具体模型如下：

$$\text{目标：}Min=\sum_{i=1}^{7}\sum_{j=1}^{15}x_{ij}a_{ij}+\sum_{i=1}^{7}\sum_{j=1}^{15}x_{ij}p_i+0.1\sum_{j=1}^{15}\frac{\left(L_j\left(L_j+1\right)+R_j\left(R_j+1\right)\right)}{2}$$

$$\begin{cases}500f_i\le\sum_{j=1}^{15}x_{ij}\le s_if_i & ,\left(i=1,2,...,7\right)\\ L_{j+1}+R_j=b_{(j,j+1)} & ,\left(j=1,2,...,14\right)\\ L_j+R_j=\sum_{i=1}^{7}x_{ij} & ,\left(j=1,2,...,15\right)\\ f_{ij}\in(0,1) & ,\left(i=1,2,...,7,\ j=1,2,...,15\right)\\ R_j\in N^* & ,\left(j=1,2,...,15\right)\\ L_j\in N^* & ,\left(j=1,2,...,15\right)\\ x_{ij}\ge 0 & ,\left(i=1,2,...,7,\ j=1,2,...,15\right)\\ a_{ij}\ge 0 & ,\left(i=1,2,...,7,\ j=1,2,...,15\right)\end{cases}$$

5.1.5 BFS 算法的具体描述

根据附件中已知的钢厂生产、铁路和公路运输的价格等条件，进行编程求解，具体步骤如下：

*Step*1：初始化图的每条边及其权重 $\mathrm{d_{ij}}$。

*Step*2：处理连续铁路，初始化矩阵 te_{ij} 为无穷大，其中 te_{ij} 表示从 i 点到 j 点只经过铁路的最短路。

枚举每一个节点分别使用一次 *BFS* 算法

（1）每次先将起始节点 *S* 加入队列

（2）接着取出队头节点 u，遍历剩余的每个节点 v

（3）如果从起始节点 S 到节点 u 的最短路长度加上节点 u 和节点 v 的直接边边权小于起始节点 S 到节点 v 的最短路长度，那么更新节点 S 到节点 v 的最短路长度并且将节点 v 加入队尾

（4）如果队列为不为空重复步骤（2）（3）（4），否则结束算法

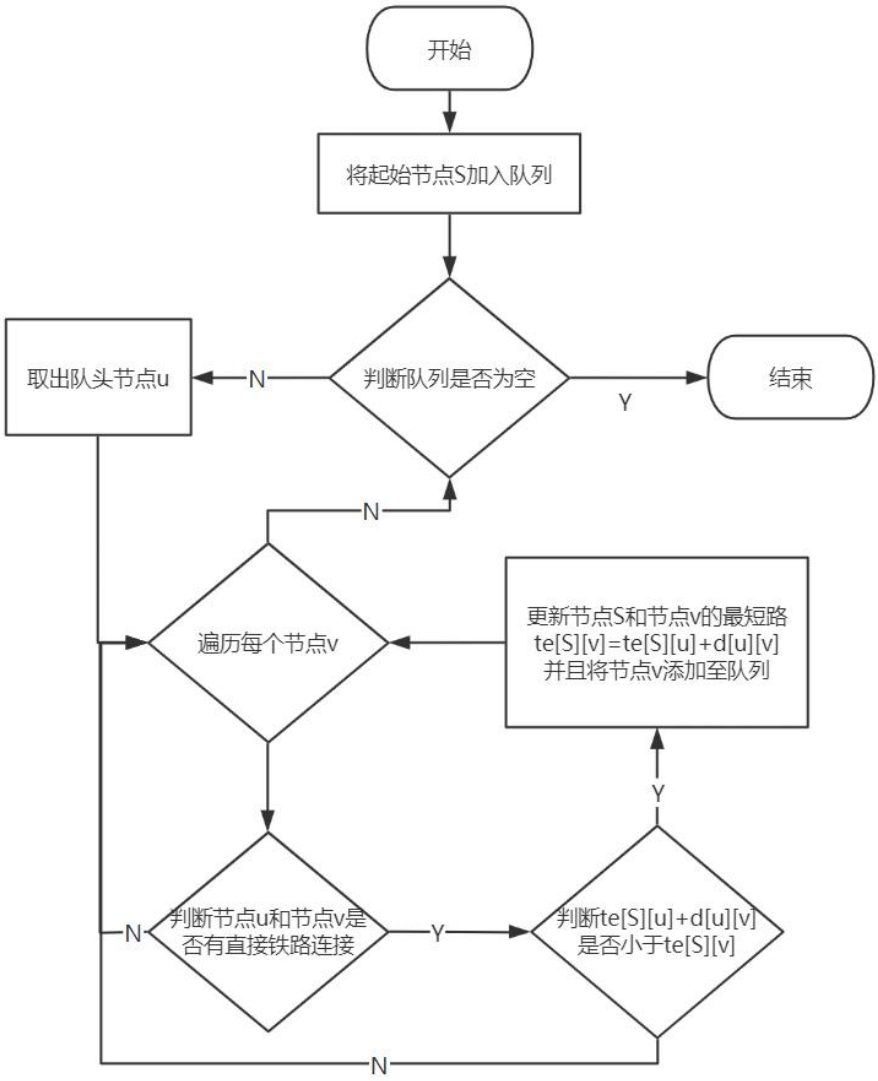

图 3 算法步骤二流程图

*Step*3：求解矩阵 a_{ij}

枚举每一个工厂节点 S 分别使用一次 *BFS* 算法

定义矩阵 cost_i 为起始节点 S 到节点 i 的最少运输费用

（1）每次先将起始节点 S 加入队列，并且初始化 cost 数组为无穷大

（2）接着取出队头节点 u，遍历剩余的每个节点 v

（3）如果从起始节点 S 到节点 u 的最少费用加上节点 u 到节点 v 的最少费用

(其中费用分为铁路最短路运输费用或者直接公路运输费用)少于节点 S 到节点 v 的最小运输费用，那么更新节点 S 到节点 v 的最少费用 $cost_v$ 并且将节点 v 加入队尾

（4）如果队列为不为空重复步骤（2）（3）（4），否则结束算法

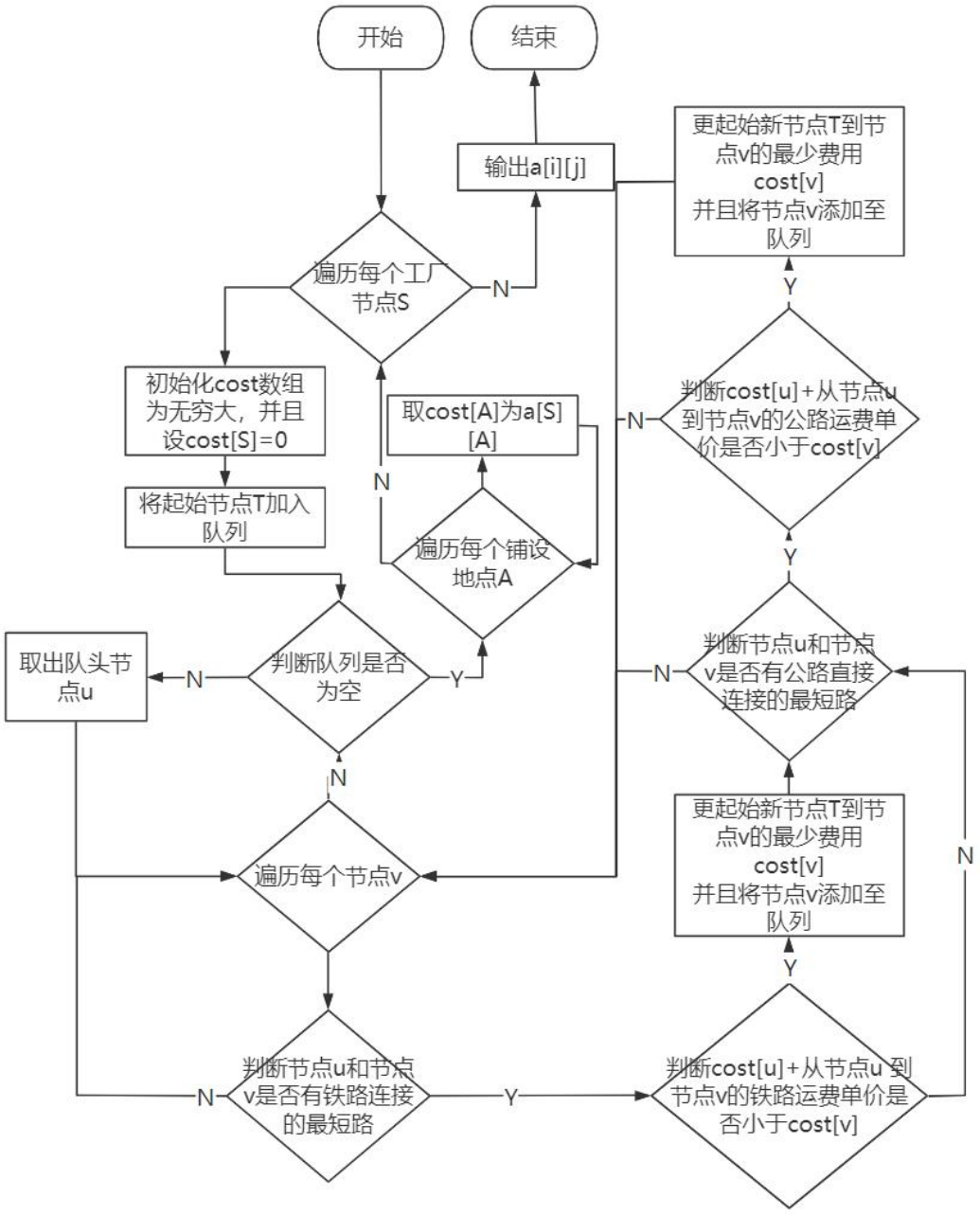

图 4 算法步骤三流程图

5.1.6 具体结果

综上所述，对 5.1.4 建立的最优化模型进行 *Lingo* 求解，得到各钢厂向各铺设点运输的钢管数量及最小总费用，具体如下表 1 所示：

表 1：各钢厂向各铺设点运输的钢管数量

	S1	S2	S3	S4	S5	S6	S7
A1	0	0	0	0	0	0	0
A2	0	179	0	0	0	0	0
A3	0	206	70	0	232	0	0
A4	122	52	167	0	127	0	0
A5	212	63	99	0	241	0	0
A6	201	0	0	0	0	0	0
A7	265	0	0	0	0	0	0
A8	0	300	0	0	0	0	0
A9	0	0	664	0	0	0	0
A10	0	0	0	0	255	96	0
A11	0	0	0	0	415	0	0
A12	0	0	0	0	0	89	0
A13	0	0	0	0	0	330	0
A14	0	0	0	0	0	621	0
A15	0	0	0	0	0	165	0

根据以上表格各钢厂向各铺设点运输的钢管数量，求解得出总费用最小为 1278632 万元，相应的订购和运输计划，具体如下：

（1）订购计划

对上表进行统计，可得到需向各钢厂订购的钢管数量和相应的价格，具体如下表 2 所示：

表 2：向各钢厂订购的钢管数量、价格

钢厂	订购钢管数量（单位：km）	价格（单位：万元）
S_1	800	128000
S_2	800	124000
S_3	1000	155000
S_4	0	0
S_5	1270	196850
S_6	1301	195150
S_7	0	0

从 S_1 钢铁厂订购 800 个单位的钢管，花费 128000 万元，从 S_2 钢铁厂订购 800 个单位的钢管，花费 124000 万元，从 S_3 钢铁厂订购 1000 个单位的钢管，花费

155000 万元，从 S_5 钢铁厂订购 1270 个单位的钢管，花费 196850 万元，从 S_6 钢铁厂订购 1301 个单位的钢管，花费 195150 万元，从 S_4 和 S_7 钢铁厂订购的钢管量为 0。

（2）运输计划

根据以上数据，可绘制出各钢厂向各铺设点运输的路线及钢管数量，具体如下：

① 钢厂 S_1 的运输路线

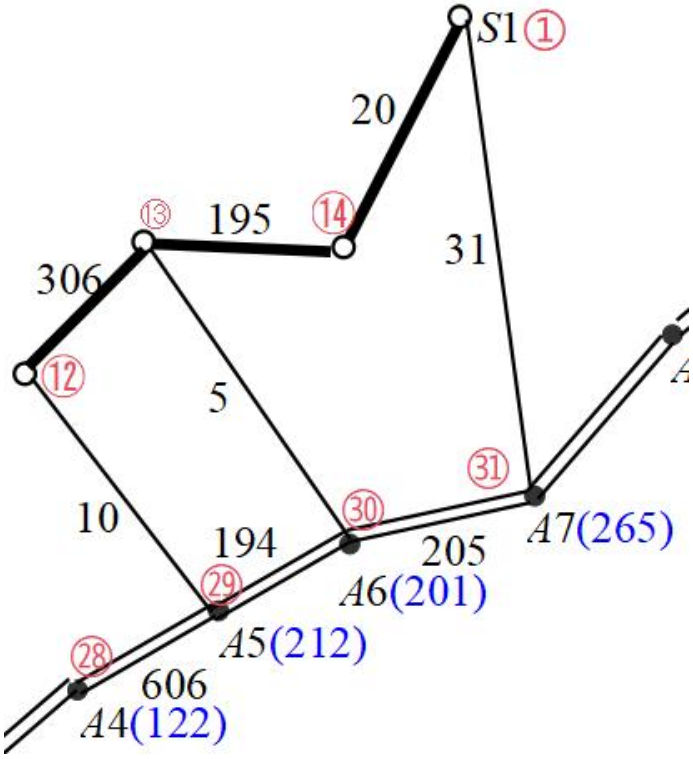

图 5 钢厂 S_1 的运输路线

具体数据如下表 3 所示：

表 3：钢厂 S_1 的运输路线及钢管数

钢厂->铺设点	路线	运输钢管数（单位：km）
$S_1 \to A_4$	$S_1 \to 12 \to 29 \to A_4$	122
$S_1 \to A_5$	$S_1 \to 12 \to A_5$	212
$S_1 \to A_6$	$S_1 \to 13 \to A_6$	201
$S_1 \to A_7$	$S_1 \to A_7$	265

② 钢厂 S_2 的运输路线

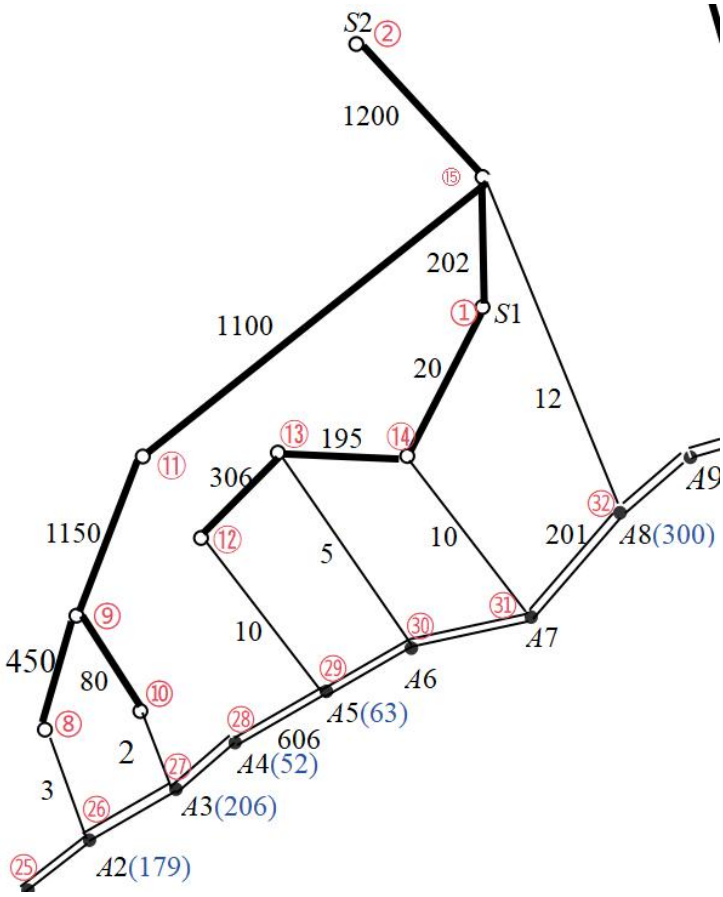

图 6 钢厂 S_2 的运输路线

具体数据如下表 4 所示：

表 4：钢厂 S_2 的运输路线及钢管数

钢厂->铺设点	路线	运输钢管数（单位：km）
$S_2 \to A_2$	$S_2 \to 8 \to A_2$	179
$S_2 \to A_3$	$S_2 \to 10 \to A_3$	206
$S_2 \to A_4$	$S_2 \to 12 \to 29 \to A_4$	52
$S_2 \to A_5$	$S_2 \to 12 \to A_5$	63
$S_2 \to A_8$	$S_2 \to 15 \to A_8$	300

③ 钢厂 S_3 的运输路线

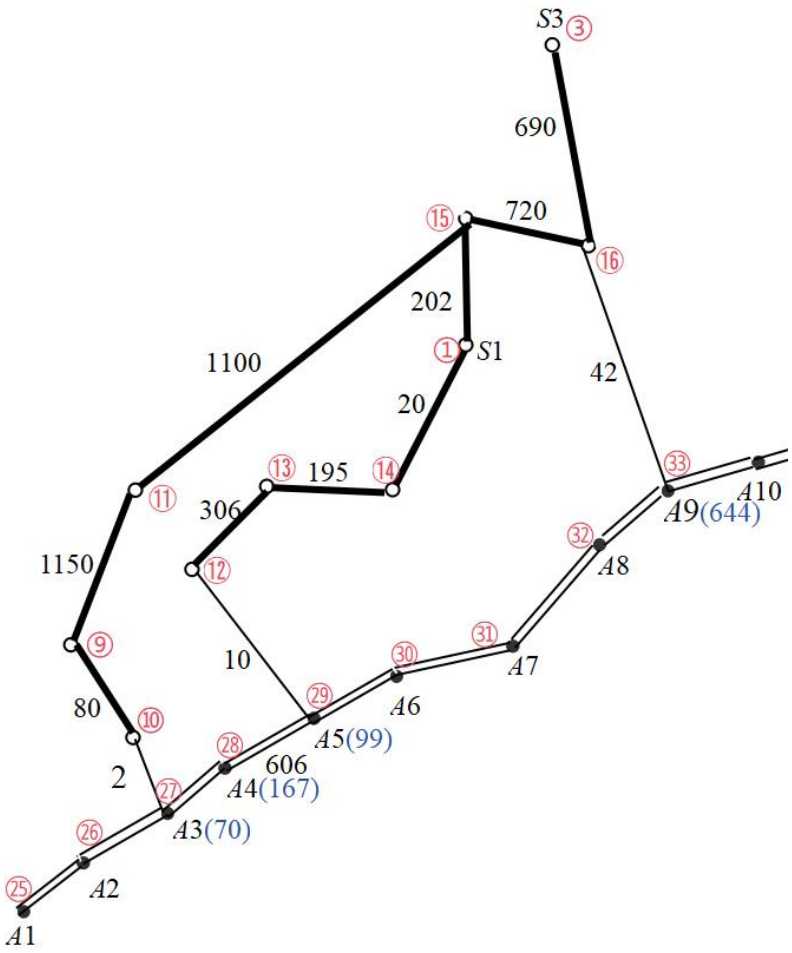

图 7 钢厂 S_3 的运输路线

具体数据如下表 5 所示：

表 5：钢厂 S_3 的运输路线及钢管数

钢厂->铺设点	路线	运输钢管数（单位：km）
$S_3 \rightarrow A_3$	$S_3 \rightarrow 10 \rightarrow A_3$	70
$S_3 \rightarrow A_4$	$S_3 \rightarrow 12 \rightarrow 29 \rightarrow A_4$	167
$S_3 \rightarrow A_5$	$S_3 \rightarrow 12 \rightarrow A_5$	99
$S_3 \rightarrow A_9$	$S_3 \rightarrow 16 \rightarrow A_9$	644

④ 钢厂 S_5 的运输路线

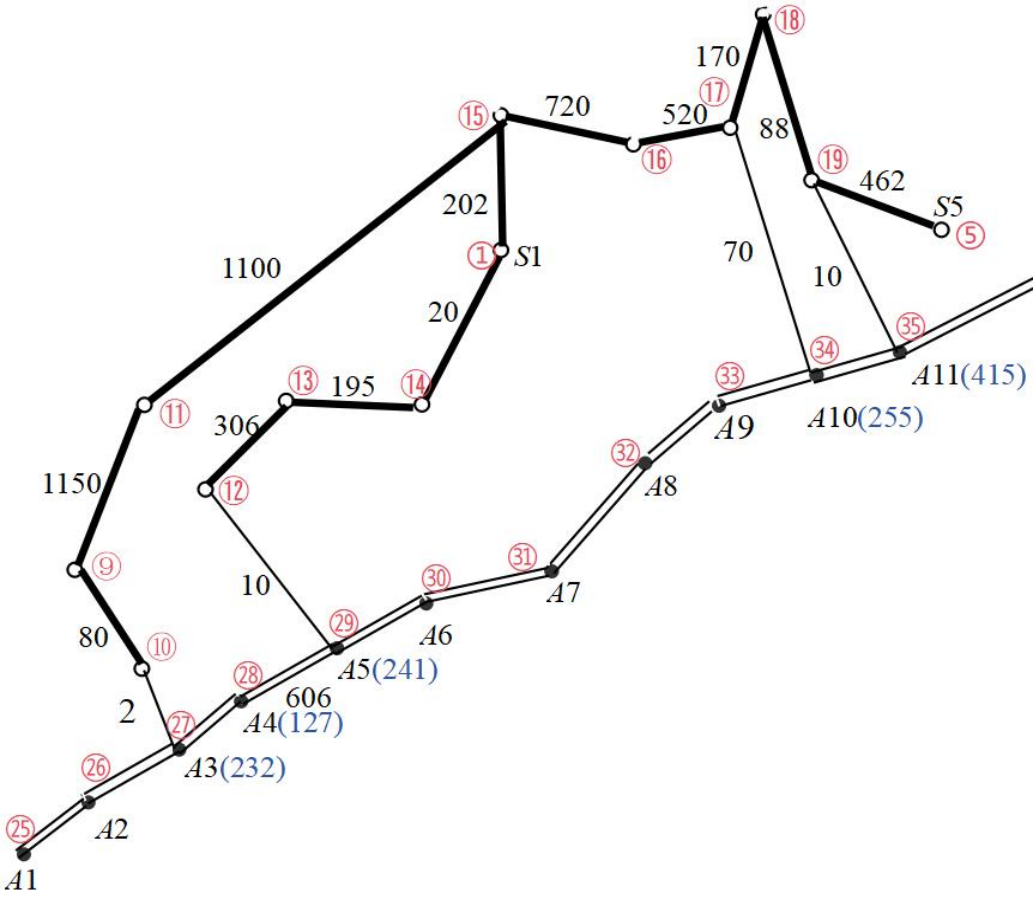

图 8 钢厂 S_5 的运输路线

具体数据如下表 6 所示：

表 6：钢厂 S_5 的运输路线及钢管数

钢厂->铺设点	路线	运输钢管数（单位：km）
$S_5 \to A_3$	$S_5 \to 10 \to A_3$	232
$S_5 \to A_4$	$S_5 \to 12 \to 29 \to A_4$	127
$S_5 \to A_5$	$S_5 \to 12 \to A_5$	241
$S_5 \to A_{10}$	$S_5 \to 17 \to A_{10}$	255
$S_5 \to A_{11}$	$S_5 \to 19 \to A_{11}$	415

⑤ 钢厂 S_6 的运输路线

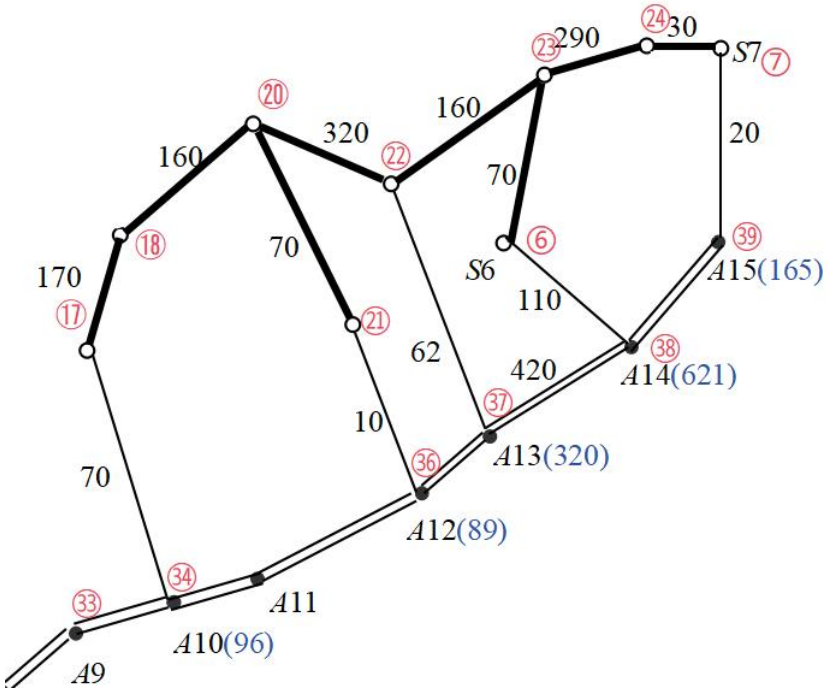

图 9 钢厂 S_6 的运输路线

具体数据如下表 7 所示：

表 7：钢厂 S_6 的运输路线及钢管数

钢厂->铺设点	路线	运输钢管数（单位：km）
$S_6 \to A_{10}$	$S_6 \to 17 \to A_{10}$	96
$S_6 \to A_{12}$	$S_6 \to 21 \to A_{12}$	89
$S_6 \to A_{13}$	$S_6 \to 22 \to A_{13}$	320
$S_6 \to A_{14}$	$S_6 \to A_{14}$	621
$S_6 \to A_{15}$	$S_6 \to 7 \to A_{15}$	165

（3）铺设计划

为了完成管道的铺设，从各铺设点出发需分成向左铺设和向右铺设，即从钢厂运输到的各铺设点的钢管量分成两部分，具体分配见下表 8：

表 8：各铺设点向两端铺设的钢管数量

铺设点	向左铺设的钢管数量（单位：km）	向右铺设的钢管数量（单位：km）
A_1	0	0
A_2	104	75
A_3	226	282
A_4	468	0
A_5	606	10
A_6	184	16
A_7	189	76
A_8	125	175
A_9	505	159
A_{10}	321	30
A_{11}	270	145
A_{12}	75	11
A_{13}	199	134
A_{14}	286	335
A_{15}	165	0

六、模型评价与推广

6.1 模型优点

1）本模型满足钢管生产和运输提供的各类约束条件

2）本模型满足总费用最小

3）本模型符合显示中多生产点与多点之间的订购和运输，具有较大的实用性

6.2 模型推广

钢管订购和运输制定是一类比较常见的事件，与现实中多个生产点与多个接收点的订购和运输计划较为类似，满足钢管生产和运输提供的限制，接收点需要的总费用最少，本模型能合理地安排多个生产点与多个接收点的订购和运输计划。

七、参考文献

[1]崔海洋,蓝可芸,丁培菲,等.基于广度优先算法的出港路径优化[J].中国科技信息,2022(07):26-28.

[2]刘晓丹,胡颖,左正康.图搜索问题算法推导及形式化证明[J].江西师范大学学报(自然科学版),2021,45(06):642-651.DOI:10.16357/j.cnki.issn1000-5862.2021.06.14.

[3]万国华,孙磊.批量运输的二层供应链系统的生产和订购计划:模型与算法[J].系统管理学报,2012,21(06):729-735.

附录

一、支撑材料的文件列表

1、cost.xlsx

2、road.xlsx

3、main.cpp

4、lingo.lg4

二、程序代码

1、Lingo 代码

```
model:
sets:
aa/1..7/:s,p,f;
bb/1..15/:L,R,T;
cc(aa,bb):w,x;
endsets

data:
w=@ole('D:\jianmo\大一\第一道：钢管运输问题\road.xlsx','s');
s=800 800 1000 2000 2000 2000 3000;
p=160 155 155 160 155 150 160;
T=104 301 750 606 194 205 201 680 480 300 220 210 420 500 0;
enddata

min=@sum(aa(I):@sum(bb(J):p(I)*x(I,J)))+@sum(cc(I,J):w(I,J)*x
(I,J))+@sum(bb(J): (R(J)*(R(J)+1)/20+L(J)*(L(J)+1)/20));

@for(aa(I):@sum(bb(J):x(I,J))>=500*f(I));
@for(aa(I):@sum(bb(J):x(I,J))<=s(I)*f(I));
@for(bb(J):R(J)+L(J)=@sum(aa(I):x(I,J)));
@for(bb(J)|J#LT#15:(R(J)+L(J+1))=T(J));
R(15)=0;
L(1)=0;
```

```
    @for(aa:@bin(f));
    @for(bb:@gin(R));
    @for(bb:@gin(L));
    @for(cc:@gin(x));
```

2、程序代码

```
#include <iostream>
#include <queue>
#include <algorithm>
#include <tuple>
#include <cstring>
#include <limits>
#include <vector>
#include <fstream>
using namespace  std;
const int N = 200;
const int INF = 0x3f3f3f3f;//无穷大
double a[N][N];//图的带权值的邻接矩阵
bool te[N][N];//铁路标记
int te_a[N][N];//铁路最短路费用
double ans[N][N];//运费矩阵 (每单位钢管从第 i 厂运输到第 j 点的运费)
double cost[N];//花费数组
int road[N];//路径数组
bool vis[N];//访问标记
void init_f(int u,int v,int w){a[u][v] = a[v][u] = w;}//初始化图的函数(初始化节点 u 到节点 v 的边权为 w)(无向图)
void init_t(int u,int v){te[u][v] = te[v][u] = true;}//初始化铁路标记函数(记节点 u 到节点 v 的路为铁路)
void init();//初始化
void solve();
void put_file();//输出文件

int main() {
    init();//初始化
    solve();//解决问题
```

```
    put_file();//输出文件
    return 0;
}

double cost_te(double x){//铁路花费函数(传入铁路长度 x,输出该长度铁路费用)
    if(x<=300)return 20;
    if(x<=350)return 23;
    if(x<=400)return 26;
    if(x<=450)return 29;
    if(x<=500)return 32;
    if(x<=600)return 37;
    if(x<=700)return 44;
    if(x<=800)return 50;
    if(x<=900)return 55;
    if(x<=1000)return 60;
    return 60 + ( int((x-1000-1)/100) + 1) *  5;
}
double cost_go(double x){//公里花费函数(传入铁路长度 x,输出该长度公路费用)
    return x*0.1;
}
void bfs(int start){

    for(int i=1;i<=39;i++){
        cost[i] = numeric_limits<double>::max();//初始化为无穷大
    }
    cost[start] = 0;
    queue<int> q;
    q.push(start);
    while(!q.empty()){
        int u = q.front();
        q.pop();
        for(int i=1;i<=39;i++){
```

```
            if(te_a[u][i] != INF  && cost[u]+cost_te(te_a[u][i]) <
cost[i]){//如果是有铁路连接，并且走铁路花费小
                road[i] = u;//记录从哪一个节点来的
                cost[i] = cost[u]+cost_te(te_a[u][i]);//更新费用
                q.push(i);
            }
            if(a[u][i] > 0 && !te[u][i] && cost[u]+cost_go(a[u][i])
< cost[i]){//如果是有公路连接，并且走公路花费小
                road[i] = u;//记录从哪一个节点来的
                cost[i] = cost[u]+cost_go(a[u][i]);//更新费用
                q.push(i);
            }
        }
    }
}
vector<int>v;
void dfs(int start,int x){
    if(x == start){//走到了起始点
        v.push_back(x);
        return ;
    }
    v.push_back(x);//记录节点
    dfs(start,road[x]);//找从哪一个节点来的
}

void solve(){

    std::ofstream file("road.xls"); // 创建 ofstream 对象并打开文件
    if (!file.is_open()) {
        std::cerr << "无法打开文件" << std::endl;
    }

    for(int i=1;i<=7;i++){
        bfs(i);
```

```
            for(int j=25;j<=39;j++){
                ans[i][j] = cost[j]; // 更新运费矩阵
            }

            for(int j=25;j<=39;j++){
                dfs(i,j);//求终点到起点的路径
                reverse(v.begin(),v.end());//将终点到起点的路径反转为
起点到终点的路径
                file << i << "->" << j-24 << '\t';
                for(auto x:v){
                    file << x << '\t';//输出路径到文件
                }
                file << '\n';
                v.clear();//清除路径
            }

        }
        for(int i=1;i<=7;i++){
            for(int j=25;j<=39;j++)
                cout << ans[i][j] << ' ';//输出运费矩阵
            cout << '\n';
        }
        cout << '\n';

}
void put_file(){
    std::ofstream file("cost.xls"); // 创建 ofstream 对象并打开文件
    if (!file.is_open()) {
        std::cerr << "无法打开文件" << std::endl;
    }

    for(int i=1;i<=7;i++){
        for(int j=25;j<=39;j++)
```

```
            file << ans[i][j] << '\t';//输出文件
        file << '\n';
    }
    cout << '\n';

    file.close(); // 关闭文件
}

void init_so(){
    //初始化铁路最短路长度

    for(int i=1;i<=39;i++)
        for(int j=1;j<=39;j++)
            te_a[i][j] = INF;

    for(int i=1;i<=39;i++){
        memset(vis,false,sizeof(vis));

        queue<pair<int,int> >q;
        q.push(make_pair(i,0));
        while(!q.empty()){
            int u,k;
            tie(u,k) = q.front();
            q.pop();
            for(int j=1;j<=39;j++){
                if(j==u)continue;
                if(te[u][j]){
                    if(a[u][j] + k < te_a[i][j]) // 判断起始节点 i 到节点 u 的最短路长度加上节点 u 和节点 j 的直接边边权是否小于起始节点 i 到节点 j 的最短路长度
                        q.push(make_pair(j,a[u][j] + k));
                    te_a[i][j] = min( k + (int)a[u][j] , te_a[i][j]);//更新最短路长度
                }
```

```
                }
            }
        }
    }
    void init(){
        //初始化每条边以及边权
        init_f(25,26,104);
        init_f(26,8,3);
        init_f(8,9,450);
        init_f(9,10,80);
        init_f(26,27,301);
        init_f(27,28,750);
        init_f(28,29,606);
        init_f(29,30,194);
        init_f(30,31,205);
        init_f(31,32,201);
        init_f(9,11,1150);
        init_f(12,13,306);
        init_f(11,28,600);
        init_f(10,27,2);
        init_f(8,26,3);
        init_f(13,14,195);
        init_f(11,15,1100);
        init_f(1,15,202);
        init_f(1,14,20);
        init_f(1,31,31);
        init_f(31,14,10);
        init_f(13,30,5);
        init_f(12,29,10);
        init_f(32,31,201);

        /////////
        init_f(2,15,1200);
        init_f(3,16,690);
```

```
init_f(16,33,42);
init_f(15,16,720);
init_f(16,17,520);
init_f(17,34,70);
init_f(17,18,170);
init_f(4,18,690);
init_f(18,19,88);
init_f(19,5,462);
init_f(19,35,10);
init_f(18,20,160);
init_f(20,21,70);
init_f(21,36,10);
init_f(20,22,320);
init_f(22,37,62);
init_f(22,23,160);
init_f(23,6,70);
init_f(6,38,110);
init_f(23,38,30);
init_f(23,24,290);
init_f(24,39,20);
init_f(24,7,30);
init_f(7,39,20);
init_f(15,32,12);


///
init_f(25,26,104);
init_f(26,27,301);
init_f(27,28,750);
init_f(28,29,606);
init_f(29,30,194);
init_f(30,31,205);
init_f(31,32,201);
init_f(32,33,680);
```

```
init_f(33,34,480);
init_f(34,35,300);
init_f(35,36,220);
init_f(36,37,210);
init_f(37,38,420);
init_f(38,39,500);

//初始化铁路标签
init_t(8,9);
init_t(9,11);
init_t(9,10);
init_t(11,15);
init_t(2,15);
init_t(15,1);
init_t(1,14);
init_t(13,14);
init_t(12,13);
init_t(15,16);
init_t(3,16);
init_t(16,17);
init_t(17,18);
init_t(4,18);
init_t(19,18);
init_t(5,19);
init_t(18,20);
init_t(20,21);
init_t(20,22);
init_t(22,23);
init_t(23,6);
init_t(23,24);
init_t(24,7);


init_so();//初始化铁路最短路长度
```

```
}
```