第十四章　路径算法

§14.1　导言

在生产管理，交通运输和通讯领域，经常会碰到这样的问题：沿着哪条路线可以最短的时间或最少的费用把货物运往目的地？沿着哪条路线传送信息最可靠或最快捷？如何组织生产可使生产成本最低？如何制定投资计划可使利润最大？这些都可看成是：在给定的加权图中，求最短路径的问题。

这一章的目的是要大家掌握求最短路径的 Dijkstra 算法，了解 Floyd 算法的由图形直觉思维转化为矩阵操作的算法思想。进一步熟悉用 MATLAB 语言编写非数值计算问题的编程技巧，学会用多重循环和选择结构来实现较复杂的穷举。学会如何建立实际问题的图论模型，希望能举一反三。

§14．2　引例

14．2．1 引例一：最短运输路线问题

如图 14.1 的交通网络，每条弧上的数字代表车辆在该路段行驶所需的时间，有向边表示单行道，无向边表示可双向行驶。若有一批货物要从 1 号顶点运往 11 号顶点，问运货车应沿哪条线路行驶，才能最快地到达目的地？

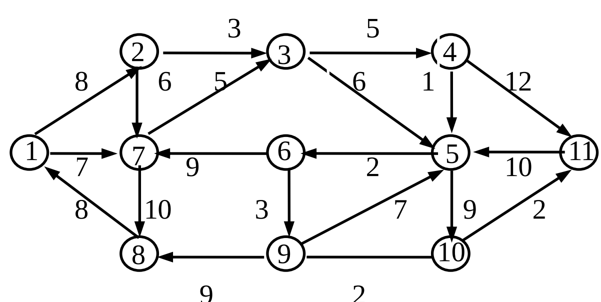

图 14.1

给定一个有向图 G，每条边上都有一个数字代表边的长度。在实际问题中这个长度可以代表费用，时间，可靠度或其他性能指标。

普通长度（ordinary path length）：路径长度定义为该路径所包含的全体

边的长度之和。对图中任意给定的两点 u, v, 在它们之间可能存在多条路径。

普通型最短路径问题(ordinery shortest-path problem): 求从 u 到 v 的路径中普通长度最短的路径。该路径称为从 u 到 v 的**最短路径**（shortest-path）。

14．2．2 引例二：最廉价航费表的制定

某公司在六个城市 C_1,C_2,C_3,C_4,C_5,C_6 都有分公司，公司成员经常往来于它们之间，已知从 C_i 到 C_j 的直达航班票价由下述矩阵的第 i 行，第 j 列元素给出（∞表示无直达航班），该公司想算出一张任意两个城市之间的最廉价路线表。

$$\begin{bmatrix} 0 & 50 & \infty & 40 & 25 & 10 \\ 50 & 0 & 15 & 20 & \infty & 25 \\ \infty & 15 & 0 & 10 & 20 & \infty \\ 40 & 20 & 10 & 0 & 10 & 25 \\ 25 & \infty & 20 & 10 & 0 & 55 \\ 10 & 25 & \infty & 25 & 55 & 0 \end{bmatrix}$$

该问题实际上是求以上述矩阵为带权邻接矩阵的加权图中，任意两点之间的最短路径及其长度的问题。

14．2．3 引例 3：数据的最可靠传输线路问题

设有十分重要的数据需要在一个通讯网的两个节点间传输，假设每条通讯链路(边)完好且不出现错码（即正常运行）的概率已知，各链路运行是相互独立的。应沿哪条线路传输才能使该信息到达目的地的可靠性最高。线路的可靠性为其上所有链路都正常运行的概率。这时，可定义边上的权为该边正常运行的概率，路径的权定义为该路径所包含的全体边的概率之积，即该路径的可靠性。两点间的最大权路径即为所求。

§14．3　最短路径问题和算法的类型

按路径长度的不同定义可将最短路径问题分为两大类：普通路径长度和一般路径长度。后者是指路径权被定义为其上边权的其他函数，如路径的权为其包含的所有边权之积，边权的最大值或其他更复杂的函数。再如，在交

通网络中，在道路的交叉口转弯时，可能会增加一个“转弯罚数”（turn penalty）。详细的分类见表 14.1。

表 14.1　最短路径问题的分类

一、普通路径长度
A.　无约束
1）最短路径
a.两个指定顶点间的最短路径
b.一个指定顶点到其余各顶点的最短路径
c.任意两顶点间的最短路径
2）第二，第三，…，第 k 短路径
B.　带约束
1）包含一些指定顶点的最短路径
2）包含一些指定边的最短路径
二、一般路径长度
A.带转弯罚数；
B.路径权为其上边权的其他函数形式，如边权之积。

我们先介绍求解类型一.A.1.: 具有普通路径长度的最短路径问题的算法。

§14．4　最短路径算法

图论问题的求解与数值问题的求解（如方程式求根，插值计算，数值积分和函数逼近等）有很大的不同，前者是“非数值性问题”，涉及到的数据结构更为复杂，数据元素之间的相互关系一般无法用数学方程式来描述。解决此类问题的关键已不再是分析数学和计算方法，而是能设计出合适的数据结构。所谓数据结构是指数据（信息的载体，能够被计算机识别，存储和加工处理）之间的相互关系。在这一章，我们将从粗略描述开始，逐步精细化的算法设计过程展示出来。并给出经调试通过的 MATLAB 程序。在这些程序中，你将会注意到，主要是进行判断，比较，而不是进行算术运算。

14．4. 1 固定起点到其余各点的最短路径算法

寻求从一固定起点 v_0 到其余各点的最短路径的最有效算法之一是 Dijkstra 算法，它是一种迭代算法。为叙述方便，我们把从起点 v_0 到顶点 v 的最短路径简称为 v 的最短路径。

要求：加权图中无负权。

出发点：最短路径上的任何子段仍是最短路径，距 v_0 远的顶点的最短路径必经过距 v_0 近的顶点。因此可按与 v_0 的距离由近及远地逐个求出各顶点的

最短路径和长度。

算法思路：设置一个集合 *S*，存放已求出其最短路径长度的顶点。

1）$S \leftarrow \{v_0\}$

2）求出 $\overline{S} = V - S$ 中与 v_0 距离最近的顶点 *u*，将 *u* 加入到 *S* 中

3）重复 2）直到 $\overline{S} = \Phi$。

例 14．1　一个简单例子

求图 14.2 中从顶点 1 到顶点 6 的最短路径及其长度。

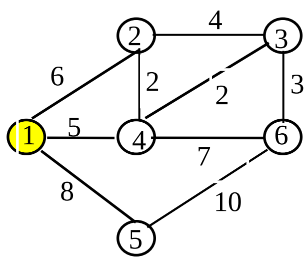

图 14.2

以 1 号顶点为起点，首先，*S*={1}，与 1 号顶点距离最近的顶点为 4 号顶点，将其加入到 *S* 中，*S*={1,4}，1 到 4 的最短路径已求出，见图 14.3 中的粗线所示，并在顶点 4 旁边标上该路径的权，*S* 中的顶点在图中为实心点。下一个与顶点 1 距离最近的顶点一定是 *S* 的邻点 2，5，3 或 6，经过 *S* 中的顶点直接到达，显然这当中，顶点 1 到顶点 2 的距离最近，其最短路径在图 14.4 中用粗线表示，将顶点 2 加入到 *S* 中，*S*={1,4,2}。

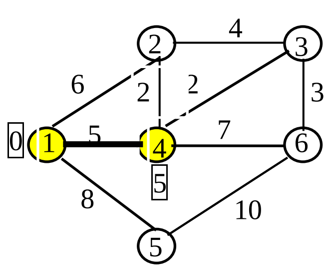

图 14.3

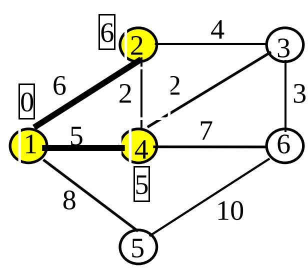

图 14.4

此时，在 $\overline{S}$ 中与顶点 1 最近的顶点为 3，经顶点 4 到达 3，距离为 5+2=7，已求出的最短路径在图 14.5 中用粗线表示，顶点 1 到各顶点的最短距离标在顶点的旁边，此时 *S*={1,4,2,5}。与前面类似，下一个离顶点 1 最近的顶点为顶点 5，接着便是顶点 6，其最短路径和距离如图 14.6 所示，该图的粗线是一棵树，树上任意两点间有唯一路径，这些路径均为最短路径。该树称为最短路径树。

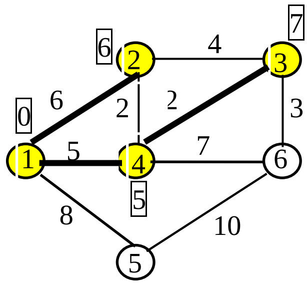

图 14.5

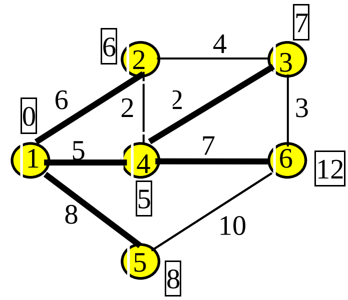

图 14.6

在这个简单例子里，第 2 步，找出 $\overline{S}$ 中离顶点 1 最近的顶点用手工操作较容易，计算机如何才能实现呢？

为直观，想象把集合 S 中的顶点涂成红色，$\overline{S}$ 中的顶点为白色。如何在白点集 $\overline{S}$ 中找出最短路径长度最小的顶点 u，加入到红点集 S 呢？

对于图中每个顶点 v，引入一个标记 $l(v)$来记录从 v_0 到 v 的，且中间只经过红点，不经过白点的路径中的最短路径长度（当从 v_0 出发，经过红点集 S 中的顶点不能到达 v 时，$l(v)$取∞）。

1）当 $v \in S$ 时，$l(v)$ 是 v_0 到 v 的最短路径长度；

2）当 $v \in \overline{S}$ 时，$l(v)$ 不小于 v_0 到 v 的最短路径长度；

3）若 $l(u) = \min\limits_{v \in \overline{S}} \{l(v)\}$，则 u 是 $\overline{S}$ 中距离 v_0 最近的顶点，且 $l(v)$ 是 u 的最短路径长度。

上述结论成立吗？为什么？

最初 $l(v_0)=0$, $\forall v \neq v_0, l(v) = \infty$,标记最小的顶点为 $u = v_0$，将其涂红加入到 S 中，从而 $S = \{v_0\}$。

当新红点 u 加入 S 后，S 改变，红点的标记不会改变，白点 v 的标记将怎样变化呢？

从起点 v_0 出发，中间只经红点到 v 的最短路径只可能是如下两种之一，

1）v 的前一个点为老红点；

2）v 的前一个点为新红点。

第一种情形 v 的最短路径长度为 $l(v)$，第二种情形 v 的最短路径长度为 $l(u) + w(u,v)$（其中 $w(u,v)$为边(u,v)的权）。因此 $l(v)$应更改为

$$\min\{l(v), l(u)+w(u, v)\}。$$

上述算法只求出了最短路径的长度，如要求出最短路径，还需记下路径。为此，对每个顶点 v，引入一个父亲点 $f(v)$，记录在 v 的只经过红点的最短路径上，v 的前一个顶点。与 $l(v)$ 一样，$f(v)$ 将随着 S 的变化而不断更新。$f(v)$ 最终的取值就可以确定从起点 v_0 出发到其余各点的最短路径。怎么确定？为什么？

因此，算法可进一步细化为

Dijkstra 算法：

输入加权图的带权邻接矩阵 $w=[w(v_i, v_j)]_{n\times n}$，所求路径的起点为 v_0

1）初始化

令 $l(v_0)\leftarrow 0$,　$\forall v\neq v_0, l(v)\leftarrow\infty$，$u\leftarrow v_0$，$S\leftarrow\{v_0\}$；

2）更新 $l(v)$, $f(v)$

$\forall v\in\bar{S}$，若 $l(v)>l(u)+w(u\ ,)$，则 $l(v)\leftarrow l(u)+w(u\ ,)$，$f(v)\leftarrow u$；

3）求出使 $l(u)=\min\limits_{v\in\bar{S}} l(v)$ 的 u, $S\leftarrow S\cup\{u\}$；

4）重复 2)，3）直到 $\bar{S}=\Phi$。

提示

1）由于任何一条最短路径的子段也是最短路径，因此在 v_0 到 v 的最短路径上，v 的前一个顶点为 $f(v)$，$f(v)$ 的前一个顶点为 $f[f(v)]$，这过程继续直到追踪到 v_0 为止，这样便可得到 v_0 到 v 的最短路径。

2）若最终某顶点 v 的标记为∞，则表明从 v_0 到 v 没有路径，即从 v_0 出发不能到达 v。

下面用一个例子来说明 Dijkstra 算法的迭代过程。

例 14．2　一个计算例子

用 Dijkstra 算法求图 14.1 从 1 号顶点到 5 号顶点的最短路径。

初始化：$l(v)$的初值如表 14.2, u=1，S={1}

表 14. 2

v	1	2	3	4	5	6	7	8	9	10	11
$l(v)$	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

第一次迭代：$\bar{S}$ 中顶点的新标号由公式 $l(v)=\min\{l(v),l(u)+w(u,v)\}$确定，修正 $l(v)$,并为 $f(v)$赋值，见表 14.3。

表 14. 3

v	1	2	3	4	5	6	7	8	9	10	11
$l(v)$	∞	8	∞	∞	∞	∞	7	∞	∞	∞	∞
$f(v)$		1					1				

对 $\bar{S}$ 中所有顶点的标号 $l(v)$进行比较，得出具有最小标号 $l(7)=7$ 的顶点 $u=7$，将 7 加入 S 得：$S=\{1,7\}$。从 1 号顶点到 S 中顶点的最短路径（由 $f(v)$ 记录）及其长度（由 $l(v)$记录）已经求得，由 $f(v)$记录的当前最短路径树见图 14.7，连接红点与白点的边用虚线表示，因为以后可能会改变，且虚线所指的顶点的标记也可能会改变。红点用实心圆表示，白点用空心圆表示。方框内的数为顶点标记，没画出的顶点的标记为∞。

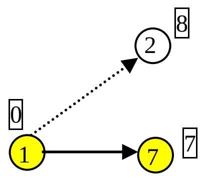

图 14.7

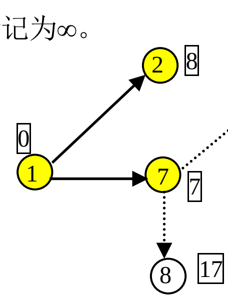

图 14.8

第二次迭代： $l(v)$, $f(v)$经第二次迭代后的值如表 14.4。

表 14. 4

v	1	2	3	4	5	6	7	8	9	10	11
$l(v)$	∞	8	12	∞	∞	∞	7	17	∞	∞	∞
$f(v)$		1	7				1	7			

对 $\bar{S}$ 中所有顶点的标号进行比较，得出具有最小标号 $l(2)=8$ 的顶点 $u=2$，将 2 加入 S 得：$S=\{1,7, 2\}$。从 1 号顶点到 S 中顶点的最短路径（由 $f(v)$记录）及其长度（由 $l(v)$记录）已经求得，当前的最短路径树见图 14.8，方框内的数为顶点标记。

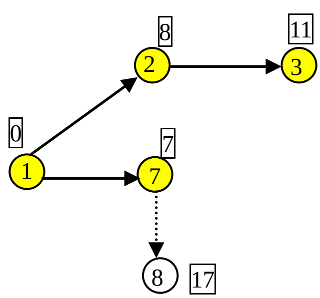

图 14.9

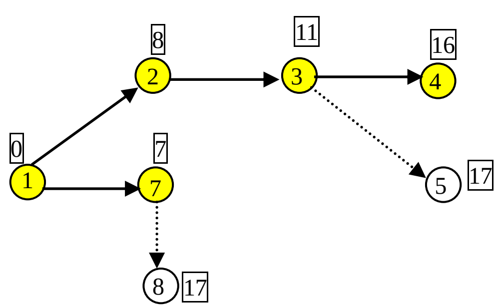

图 14.10

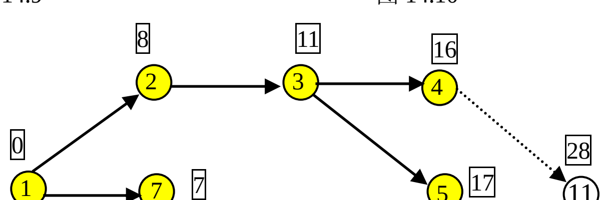

图 14.11

以后的各次迭代类似，对应的最短路径树见图 14.9——图 14.11，表 14.5 列出了 $l(v)$, $f(v)$在各次迭代的取值。

表 14.5

迭代	$l(v)$ $(f(v))$											u
	1	2	3	4	5	6	7	8	9	10	11	
0	$\boxed{0}$	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	1
1	$\boxed{0}$	8(1)	∞	∞	∞	∞	$\boxed{7(1)}$	∞	∞	∞	∞	7
2	$\boxed{0}$	$\boxed{8(1)}$	12(7)	∞	∞	∞	$\boxed{7(1)}$	17(7)	∞	∞	∞	2
3	$\boxed{0}$	$\boxed{8(1)}$	$\boxed{11(2)}$	∞	∞	∞	$\boxed{7(1)}$	17(7)	∞	∞	∞	3
4	$\boxed{0}$	$\boxed{8(1)}$	$\boxed{11(2)}$	$\boxed{16(3)}$	17(3)	∞	$\boxed{7(1)}$	17(7)	∞	∞	∞	4
5	$\boxed{0}$	$\boxed{8(1)}$	$\boxed{11(2)}$	$\boxed{16(3)}$	$\boxed{17(3)}$	∞	$\boxed{7(1)}$	17(7)	∞	∞	28 (4)	5

方框“□”里的数表示 S 中顶点的标记 $l\ (v)$和父亲点 $f\ (v)$，在以后的迭代中不会改变。各次迭代对应的最短路径树在图 14.7 到图 14.11 中依次绘出，方框内的数为顶点标记。到第五次迭代，5 号顶点已在 S 中，因此，对应的最短路径树中从 1 到 5 的唯一路径 1-2-3-5，就是 1 到 5 的最短路径，$l(5)=17$ 便是其长度。该最短路径也可由父子关系追踪而得。由 $f(5)=3$ 知，5 的前一个点为 3；由 $f(3)=2$ 知，3 的前一个点为 2；由 $f(2)=1$ 知，2 的前一个点为 1。因此得到 5-3-2-1，倒过来就是 1 到 5 的最短路径。

注意

1）若想求出从 1 到其余各顶点的最短路径，则继续上述过程直到 S 包含全部顶点。

2）Dijkstra 算法的运算时间是 $O(n^2)$ 级的。该法既适合于有向图又适合于无向图，只是要求所有权都必须非负。

1）试分析 Dijkstra 算法的时间复杂度，证实为 $O(n^2)$。
2）在 MATLAB 环境下，编制用 Dijkstra 算法求最短路径的 M 文件函数，求解 14.2.1 所给出的“最短运输路线问题”。

1）如果想要求出两点间的全体最短路径应怎么办？当最短路径被占用或出现故障时，就必须改用次短，再次短的路径，如何求出第 k 短的最短路径？
2）如果带有负权，如何求最短路径？

14.4.2 每对顶点间的最短路径算法

显然此问题可由重复 Dijkstra 算法来解决，每次取定一个顶点作起点，但这需要大量重复计算，效率不高。Floyd 另辟蹊径，提出了比这更好的算法，可一次性地求出任意两点间的最短路径和距离，其思想方法很有创意，与 Dijkstra 算法截然不同。

Floyd 算法的基本思路： 从图的带权邻接矩阵 $A=[a(i,\ j)]_{n\times n}$ 开始，递归地进行 n 次更新，即由矩阵 $D^{(0)}=A$, 按一个公式，构造出矩阵 $D^{(1)}$；又用同样的公式由 $D^{(1)}$构造出矩阵 $D^{(2)}$；……；最后又用同样的公式由 $D^{(n-1)}$构造出矩阵 $D^{(n)}$。矩阵 $D^{(n)}$的 i 行 j 列元素便是 i 号顶点到 j 号顶点的最短路径长度，称 $D^{(n)}$为图的距离矩阵，同时还可引入一个后继点矩阵 path 来记录两点间的最短路径。

递推公式为

$D^{(0)}=A$;

$D^{(1)}=[d_{ij}^{(1)}]_{n\times n}$, 其中 $d_{ij}^{(1)}=\min\{ d_{ij}^{(0)},\ d_{i1}^{(0)}+ d_{1j}^{(0)}\}$;

$D^{(2)}=[d_{ij}^{(2)}]_{n\times n}$, 其中 $d_{ij}^{(2)}=\min\{ d_{ij}^{(1)},\ d_{i2}^{(1)}+ d_{2j}^{(1)}\}$;

……

$D^{(n)}=[d_{ij}^{(n)}]_{n\times n}$, 其中 $d_{ij}^{(n)}=\min\{ d_{ij}^{(n-1)},\ d_{i,n-1}^{(n-1)}+ d_{n-1,j}^{(n-1)}\}$;

$d_{ij}^{(1)}$：中间只允许经过 1 号顶点，从 i 到 j 的路径中，最短路径的长度；

$d_{ij}^{(2)}$：中间只允许经过 1，2 号顶点，从 i 到 j 的路径中，最短路径的长度；

……

$d_{ij}^{(k)}$：中间只允许经过 1，2 … k 号顶点，从 i 到 j 的路径中，最短路径的长度；

……

$d_{ij}^{(n)}$：中间允许经过 1，2 … n 号顶点（即任何顶点），从 i 到 j 的路径中，最短路径的长度，此即为 i 到 j 的最短路径长度。

上述矩阵序列{$D^{(k)}$}可递归地产生，利用循环迭代便可简便求出。算法的详细步骤如下

Floyd 算法步骤：

d(i, j)：$d_{ij}^{(k)}$；

path(i, j)：对应于 $d_{ij}^{(k)}$的路径上 i 的后继点，最终的取值为 i,到 j 的最短路径上 i 的后继点。

输入带权邻接矩阵 $A=[a(i, j)]_{n\times n}$

1）赋初值

对所有 i, j, d(i, j)=a(i, j); 当 a(i, j)=∞ 时, path(i, j)=0, 否则 path(i, j)=j; k=1。

2）更新 d(i, j),path(i, j)

对所有 i, j，若 d(i,k)+d(k,j)>=d(i,j), 则转 3)；否则 d(i,j)=d(i,k)+d(k,j)，path(i,j)=path(i,k)，k=k+1, 继续执行 3)。

3）重复 2）直到 k=n+1。

例 14．2　一个编程例子

借助 MATLAB 软件，用 Floyd 算法求图 14.12 所示的加权有向图中任意两点间的最短路径及距离。

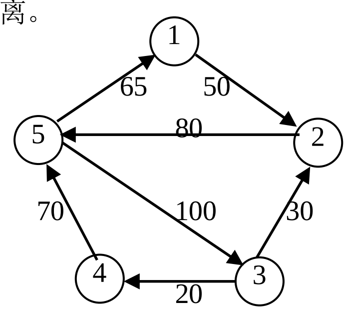

图14.12

加权有向图的存储结构采用带权邻接矩阵[a(i, j)]。

MATLAB程序：

```
% Floyd's Algorithm
function [D,path]=floyd1(a)
n=size(a,1);
%设置D和path的初值
D=a; path=zeros(n,n);
```

```
for i=1:n
    for j=1:n
        if D(i,j)~=inf
            path(i,j)=j;        %j是i的后继点
        end
    end
end
%做n次迭代，每次迭代均更新D(i,j)和path(i,j)
for k=1:n
    for i=1:n
        for j=1:n
            if D(i,k)+D(k,j)<D(i,j)
                D(i,j)=D(i,k)+D(k,j);       %修改长度
                path(i,j)=path(i,k);        %修改路径
            end
        end
    end
end
```

在 MATLAB 命令窗键入：

```
a=[0 50 inf inf inf; inf 0 inf inf 80; inf 30 0 20 inf; inf inf inf 0 70;65 inf …
100 inf 0];
[D,path]=floyd1(a)
```

运行结果：

```
D =
     0    50   230   250   130
   145     0   180   200    80
   155    30     0    20    90
   135   185   170     0    70
    65   115   100   120     0
path =
     1     2     2     2     2
     5     2     5     5     5
     4     2     3     4     4
     5     5     5     4     5
     1     1     3     3     5
```

因此，由最短距离矩阵D和最短路径矩阵path，容易得出任意两点之间的最短路径及其长度。如，顶点1到顶点3的最短路径长度：D(1,3)=230，最短路径：1-->2-->5-->3。这是因为，path(1,3)=2,意味着顶点1的后继点为2，又path(2,3)=5，从而顶点2的后继点为5，同理，因path(5,3)=3，从而顶点5的后继点为3，故1-->2-->5-->3便是顶点1到顶点3的最短路径。

在该程序运行过程中，每次循环所得到的D和path的变化情况在表14.6中给出。

表14.6　Floyd算法求解例14.2的动态执行情况

K=0	D =						path=					
		0	50	Inf	Inf	Inf		1	2	0	0	0
		Inf	0	Inf	Inf	80		0	2	0	0	5
		Inf	30	0	20	Inf		0	2	3	4	0
		Inf	Inf	Inf	0	70		0	0	0	4	5
		65	Inf	100	Inf	0		1	0	3	0	5
k=1	D =						path=					
		0	50	Inf	Inf	Inf		1	2	0	0	0
		Inf	0	Inf	Inf	80		0	2	0	0	5
		Inf	30	0	20	Inf		0	2	3	4	0
		Inf	Inf	Inf	0	70		0	0	0	4	5
		65	115	100	Inf	0		1	1	3	0	5
k=2	D =						path=					
		0	50	Inf	Inf	130		1	2	0	0	2
		Inf	0	Inf	Inf	80		0	2	0	0	5
		Inf	30	0	20	110		0	2	3	4	2
		Inf	Inf	Inf	0	70		0	0	0	4	5
		65	115	100	Inf	0		1	1	3	0	5
k=3	D =						path=					
		0	50	Inf	Inf	130		1	2	0	0	2
		Inf	0	Inf	Inf	80		0	2	0	0	5
		Inf	30	0	20	110		0	2	3	4	2
		Inf	Inf	Inf	0	70		0	0	0	4	5
		65	115	100	120	0		1	1	3	3	5
k=4	D =						path=					
		0	50	Inf	Inf	130		1	2	0	0	2
		Inf	0	Inf	Inf	80		0	2	0	0	5
		Inf	30	0	20	90		0	2	3	4	4
		Inf	Inf	Inf	0	70		0	0	0	4	5
		65	115	100	120	0		1	1	3	3	5
k=5	D =						path=					
		0	50	230	250	130		1	2	2	2	2
		145	0	180	200	80		5	2	5	5	5
		155	30	0	20	90		4	2	3	4	4
		135	185	170	0	70		5	5	5	4	5
		65	115	100	120	0		1	1	3	3	5

提示

1）Floyd 算法中记录最短路径的矩阵Path，与Dijkstra算

法中记录最短路径的向量*f*有何区别和联系？如何由矩阵Path得到任意两点间的最短路径？为什么？在m文件floyd1.m中加入一些语句，使其直接输出全部顶点间的最短路径及其长度，而不只是D和path 。

2）两算法的基本思路完全不同，是图论中颇具代表性的两类算法，它们各自妙在何处？受此启发，构思不同于Floyd算法的求任意两点间最短路径方法。

＊§14．5　一般型最短（或最长）路径问题

上一节讨论的最短路径问题，其路径权为路径上各边权之和。但在实际问题中常有这样的情况，路径的权是边权的其他函数。例如，在14.2.3中介绍的“数据的最可靠传输线路问题”，这样的问题颇具代表性，电线网，交通网中也有类似的问题。每条边有可靠度$\mathrm{p}_k: 0 \le \mathrm{p}_k \le 1$作为边权。可靠度可以是通道正常运转的概率，也可以是此通道空闲的概率，我们的目的是要找出两顶点间最可靠的路径传送物流。显然，路径的可靠度为路径上每条边的可靠度之积，此时路径的权可定义为路径上所有边权的乘积，不再是边权之和。

14．5．1 最可靠线路问题

在图G中，每条边的权为0与1之间的数，表示线路的可靠性，*P*是从s到t的路径，定义路径*P*的长度　$w(P)=\prod_{e\in P} w(e)$，“$\prod$ ”是连乘号。称s到t的路径中路径长度*w*(*P*)最大的路径为s到t的最可靠路径。

如何求出两点间的最可靠路径呢？

方法一：转化为最短路径问题。将图G上每条边的权*w*(*e*)换为–ln(*w*(*e*))，得到与G结构相同，但权不一样的加权图G’，求出G’的最短路径，该最短路径必为原图G的最可靠路径，反之亦然。

方法二：可将Dijkstra算法稍作修改来求最可靠路径。

算法：

输入加权图G的带权邻接矩阵$w=[w(v_i,v_j)]_{n\times n}$，($0\le w(v_i, v_j)\le 1$)，所求最可靠路径的起点为v_0

1）初始化

令$l(v_0)=1$,　$\forall v \ne v_0, l(v)=0$，$u=v_0$，$S=\{v_0\}$；

2）更新 $l(v), f(v)$

$\forall v \in \overline{S}$ ，若 $l(v) < l(u) * w(u,v)$ ，则 $l(v) \leftarrow l(u) * w(u,v)$ ，$f(v) \leftarrow u$；

3）求出使 $l(u) = \max_{v \in \overline{S}} l(v)$ 的 u, $S \leftarrow S \cup \{u\}$；

4）重复 2），3）直到 $\overline{S} = \Phi$。

14．5．2 最小爬高度路径问题

在图 G 中，设边权 $w(e)$表示沿 e 的爬高度，则从 s 到 t 沿路径 P 的爬高度为 $w(P) = \max_{e \in P} \{w(e)\}$

我们把从 s 到 t 的路径中爬高度最小的路径称为从 s 到 t 的最小爬高度路径。

同样，可将求最短路径的算法修改来求最小爬高度路径。

修改 Dijkstra 算法，使其适合于求最小爬高度路径，写出该修正算法的步骤。

14．5．3 更一般的路径问题

一般地，只要路径的长度函数和路径问题属于下述两种情况，均可由 Dijkstra 算法稍作修改之后求解。

情形一：对任意路径 $P = v_0 v_1 v_2 \cdots v_k, P' = v_0 v_1 v_2 \cdots v_k v_{k+1}$ 均有 $w(P) \geq w(P')$，求两点间路径中长度最大的路径。

情形二：对任意路径 $P = v_0 v_1 v_2 \cdots v_k, P' = v_0 v_1 v_2 \cdots v_k v_{k+1}$ 均有 $w(P) \leq w(P')$，求两点间路径中长度最小的路径。

最可靠线路问题属于情形一，最小爬高度问题属于情形二。

§14．6　范例：设备更新问题

14．6．1 问题

设备的更新和改造是一项系统工程，并且是一种动态系统。从系统内部分析，由于设备在使用过程中处于经常的运行状态，必然产生有形磨损；从系统所处的外部环境来看，由于科学技术的发展，新一代设备的问世，将使

原有设备的使用价值和价值降低，产生无形磨损。设备更新则是彻底消除这两种磨损的手段。一台设备究竟使用多长时间就需更新？

14．6．2 函数优化模型及其求解

考虑一台设备使用多久更换，可使平均每年的成本最低。

决策变量：设备的使用年限 t；

目标函数：一年的成本 Y。成本包括：设备的维持费用（即维修费，能耗，事故及效率下降损失等），设备折旧费，投资利息。

假设

1）设备的维持费用按等差级数逐年增大，即每年增加的费用差额为常数λ；

2）设备更新投资额为 K_0，设备残值为 E，年利率为 Z。

提示

1）假设 2 是合理的，但假设 1 把问题大大地理想化了，使问题变得非常简单，容易处理。该假设的合理性需通过实际的数据来检验。若与实际偏差大，则需作更一般化的假设，当然，模型也更复杂。

2）假设实际上就是给出问题的已知条件，假设不同，当然解决方法不同，模型就不一样。后面第二个模型在不同的假设下，导出了不同的模型。

由假设 1，按等差级数求和公式，易求出 t 年的维持费用

$$S = t(t-1)\lambda/2$$

则设备每年平均维持费为

$$C = S/t = (t-1)\lambda/2$$

由假设 2，得年折旧费 A 和年投资利息 B 分别为

$$A = (\mathrm{K}_0 - \mathrm{E})/t$$

$$\mathrm{B} = (\mathrm{K}_0 + \mathrm{E})\mathrm{Z}/2$$

因此设备平均每年的总成本为

$$\begin{aligned} Y = C + \mathrm{B} + A &= (t-1)\lambda/2 + (\mathrm{K}_0 + \mathrm{E})\mathrm{Z}/2 + (\mathrm{K}_0 - \mathrm{E})/t \\ &= (\mathrm{K}_0 - \mathrm{E})/t + [(t-1)\lambda + (\mathrm{K}_0 + \mathrm{E})\mathrm{Z}]/2 \end{aligned}$$

确定设备经济寿命的问题就归结为下述模型

$$\min Y = (\mathrm{K}_0 - \mathrm{E})/t + [(t-1)\lambda + (\mathrm{K}_0 + \mathrm{E})\mathrm{Z}]/2$$

其中 t 为决策变量。

为求使 Y 值最小的 t^*值，

令 $\dfrac{dY}{dt}=0$，即 $\dfrac{\lambda}{2}-\dfrac{\mathrm{K}_0-\mathrm{E}}{t^2}=0$

解出 $t^*=\sqrt{\dfrac{2(\mathrm{K}_0-\mathrm{E})}{\lambda}}$，$t^*$ 即为设备的最佳更新周期。

将 t^* 代入 Y，可得设备的最低年均总成本为

$$Y^*=\sqrt{\frac{2(\mathrm{K}_0-\mathrm{E})}{\lambda}}+\frac{\mathrm{K}_0\mathrm{Z}-\lambda}{2}$$

14．6．3 网络模型及其求解

考虑车辆的更新策略。一辆汽车从购进到更新，使用年限不同，更新速度不同，其总成本有明显差别。问题是一辆汽车使用几年更新，其运营总成本最少。

假设：

1）只在每年年初考虑是否更新车辆；车辆最多能用 4 年。

2）考虑一辆车的 8 年规划。

3）运营成本只考虑投资，维持费用和车辆的残值。

4）第 i 年初一辆新车的价格为$\mathrm{P_i}$；车辆在第 i 年初到第 j 年初这段时间的维持费用为$\mathrm{m_{ij}}$（包括维修费，能耗，事故及运营收入损失）；第 i 年购进，第 j 年出售的旧车价格为$\mathrm{s_{ij}}$。其中 i, j=1,2,3,…,8,9。

提示 显然假设 1，2，3 都比较合理。假设 4 里的各参数需通过市场调查和过去的运营状况来进行预测，这些参数的获取也需建立数学模型。这里假设它们已知，是把问题分解，我们只考虑其中的一个子问题。

由假设 3，4，可确定：车辆于第 i 年购进，第 j 年出售，这期间车辆的使用成本为

$$\mathrm{C_{ij}}=\mathrm{P_i}+\mathrm{m_{ij}}-\mathrm{s_{ij}}$$

由此可计算出各种更新方案的单元成本估计值 C_{ij}。

建立一个加权有向图 G：用 9 个顶点表示 9 个年头的年初，有向边(i, j)表示第 i 年购进一辆车，第 j 年出售该车这个阶段，其上的权为这阶段的单元成本估计值 C_{ij}，作出的加权有向图 G 如图 14.1 3 所示。顶点 1 到顶点 9 的任何一条有向路径都对应了一个车辆更新方案，且有向路径的长度就是对应的更新方案的总成本。例如有向路径 1→3→6→9 就表示第一年购进，用到第 3 年更新，然后再在第 6 年更新，用到第 9 年，该方案的总成本为 $C_{13}+C_{36}+C_{69}$。

求最佳车辆更新策略，使总成本最低的问题便转化为：求加权有向图 G 从顶点 1 到顶点 9 的最短路径，长度对应于费用。

假设各阶段的单元成本估计值 C_{ij} 分别都已求出，在图 14.13 中标出。用 Dijkstra 算法，借助于 MATLAB 软件可求得顶点 1 到顶点 9 的最短路径为 1→5→9。因此，车辆的最优更新策略即为第一年购进，第 5 年更新，再用到第 9 年，总成本为 11250。

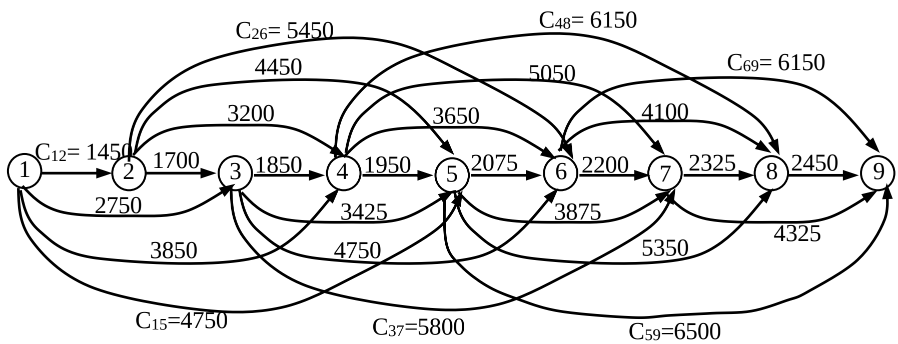

图 14.13

按照第二个模型的假设，能否建立整数规划模型？哪些整数规划问题可转化为网络的最短路径问题？比如，整数背包问题能转化为网络的最短路径问题吗？

§14.8 实验

14．8．1 实验一：计算机网络布线

表 14.7 主干网连接一览表（表中数据为距离）

	总部	子公司 1	子公司 2	子公司 3	电视台	电信部门
总部	——	5	——	10	15	7
子公司 1		——	12	——	——	11
子公司 2			——	——	20	4
子公司 3				——	10	——
电视台					——	——
电信部门						——

某公司下属多家单位均建有局域网，其中有几家已连上公司的主干网，连接情况见表 14.7，现有 A,B 两家单位平时业务往来较多，欲通过与其他单位连接，通往主干网，表 14.8 给出了 A, B 两单位与可连接的单位的距离。问应如何连接能使 A,B 之间的距离最短？

表 14.8 A,B 两单位与可连接的单位（表中数据为距离）

	总部	子公司 1	子公司 2	子公司 3	电视台	电信部门
A 单位	——	——	3	1	——	3
B 单位	3	5	——	——	8	——

14．8．2 实验二：矿厂选址

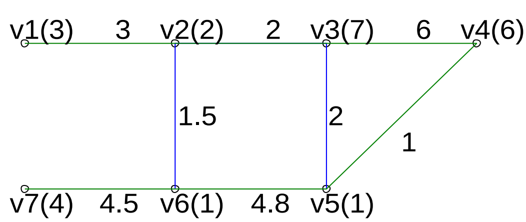

图 14.15

矿石在若干个采矿点被采下后，需集中运输到统一的矿厂进行处理。某矿区有七个采矿点，其线路图如图 14.15，已知各矿点每天的产矿量 $q(v_i)$(标在图中的各顶点上)。现要从这七个采矿点中选一个来建造选矿厂。问应把选矿厂建在哪个采矿点处，才能使各采矿点所产的矿运到选矿厂所在地的总运

力（千吨公里）最小。

参考文献

[1] 龚劬，图论与网络最优化算法，重庆大学应用数学系，1998。

[2] 姚健钢等，长方体材料截断切割的优化设计，数学的实践与认识，1998.1. Vol.28, No.1, p88-93。

[3] 李人厚等译，精通 MATLAB 综合辅导与指南，西安交通大学出版社，1998。