7.6　最短路径

最短路径

广度优先搜索

- 某一地区的一个公路网，给定了该网内的n个城市以及这些城市之间的相通公路的距离，能否找到城市A到城市B之间一条最近的通路呢?
- 从A地到B地换车次数最少的路径
- 从A地到B地最短的路径（距离最短，行驶时间最短）

最短路径

- 迪杰斯特拉算法----从一个源点到其它各点的最短路径
- 弗洛伊德算法----每一对顶点之间的最短路径

迪杰斯特拉算法

- 按路径长度递增的次序产生最短路径
- 设图G=(V,E), V={v_0, v_1,…, v_{n-1}},源点v_0, 求v_0到其余各点的最短路径。
- 分析： 设v_0到v_1,…, v_{n-1}的最短路径分别为P_1,P_2， …,P_{n-1}. 若这n-1条路径中最短的一条为P_i($1\le i\le n-1$), 那么它一定是弧< v_0, v_i>；
- 若这n-1条路径中第二短的一条为P_j($1\le i\ne j\le n-1$),那么它一定是弧< v_0, v_j>或者是路径< v_0, v_i> < v_i, v_j> ；
- …….

该算法只适用于静态网络

网络上边的权值不能为负数

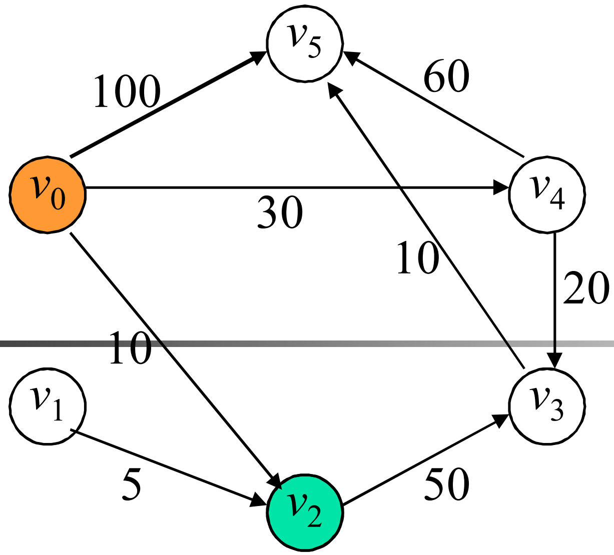
迪杰斯特拉算法

- 一种求单源点最短路径的算法，即从一个点到所有其他点的最短路径。
- 基本思想：设集合S中存放已找到最短路径的顶点，集合T＝V－S存放当前还未找到最短路径的顶点。

(1) 初态:S中只包含源点v_0，v_0到其余各点的弧为各点当前的“最短”路径。

(2) 从T中选取当前的“最短”路径长度最短的顶点u加入到S中。S每加入一个新的顶点u，都要修改顶点v_0到T中剩余顶点的最短路径长度，T中各顶点新的最短路径长度值为原来的最短路径长度值与顶点u的最短路径长度值加上u到该顶点的路径长度值中的较小值。

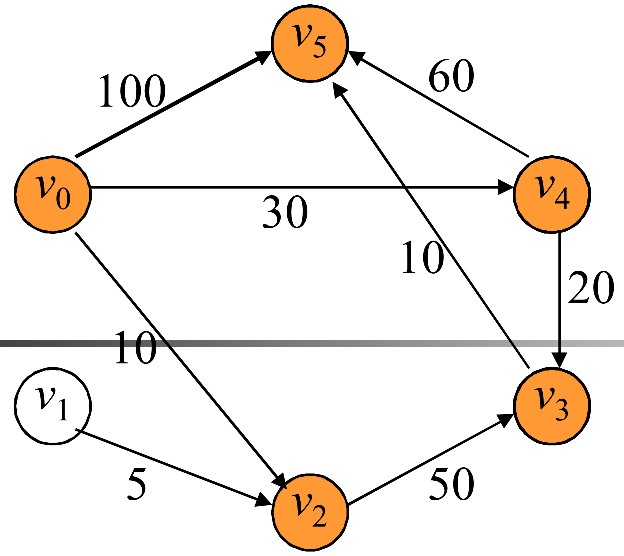
(3) 重复(2)，直到T的顶点全部加入到S中为止。



$$
\begin{bmatrix}
\infty & \infty & 10 & \infty & 30 & 100 \\
\infty & \infty & 5 & \infty & \infty & \infty \\
\infty & \infty & \infty & 50 & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & 10 \\
\infty & \infty & \infty & 20 & \infty & 60 \\
\infty & \infty & \infty & \infty & \infty & \infty
\end{bmatrix}
$$

初态	i=1	i=2	i=3	i=4	i=5
v_1	∞				
v_2	10 (v_0,v_2)				
v_3	∞				
v_4	30 (v_0,v_4)				
v_5	100 (v_0,v_5)				
选择点					
S	$\{v_0\}$				

迪杰斯特拉算法求解过程



v_0到v_1的最短路径:无穷
v_0到v_2的最短路径:10, $v_0 \rightarrow v_2$
v_0到v_3的最短路径:50, $v_0 \rightarrow v_4 \rightarrow v_3$
v_0到v_4的最短路径:30, $v_0 \rightarrow v_4$
v_0到v_5的最短路径:60, $v_0 \rightarrow v_4 \rightarrow v_3 \rightarrow v_5$

初态	i=1	i=2	i=3	i=4	i=5
v_1	∞	∞	∞	∞	∞
v_2	10 (v_0,v_2)				
v_3	∞	60 (v_0,v_2,v_3)	50 (v_0,v_4,v_3)		
v_4	30 (v_0,v_4)	30 (v_0,v_4)			
v_5	100 (v_0,v_5)	100 (v_0,v_5)	90 (v_0,v_4,v_5)	60 (v_0,v_4,v_3,v_5)	
选择点	v_2	v_4	v_3	v_5	
S	$\{v_0,v_2\}$	$\{v_0,v_2,v_4\}$	$\{v_0,v_2,v_3,v_4\}$	$\{v_0,v_2,v_3,v_4,v_5\}$	

迪杰斯特拉算法求解过程

迪杰斯特拉算法

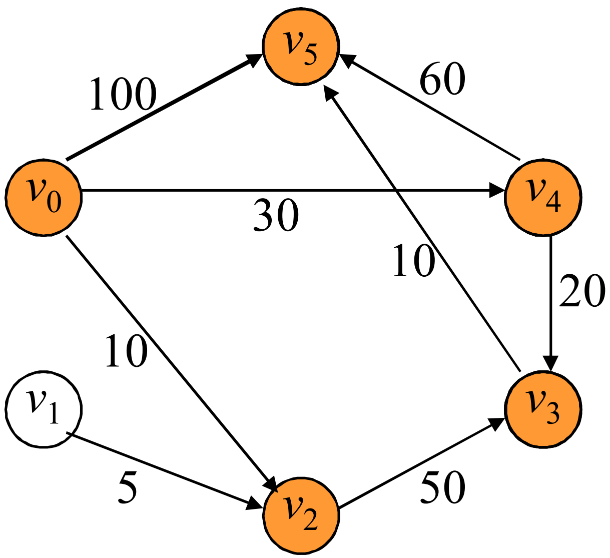
- 实现迪杰斯特拉算法尚需解决以下几个问题:
  - 图的存储?
  - 如何区分已经求出最短路径的点，即：如何区分S中的点和V-S中的点?
  - 如何表示源点到顶点i的最短路径?

迪杰斯特拉算法实现--图的存储

图的存储：邻接矩阵和邻接表都可以，下面以**邻接矩阵**为例给出算法实现

```
#define max 100
typedef struct {
            int arcs[max][max];
            int vexnum,arcnum;}AGraphs;

Agraphs G;
```



$$\begin{bmatrix} \infty & \infty & 10 & \infty & 30 & 100 \\ \infty & \infty & 5 & \infty & \infty & \infty \\ \infty & \infty & \infty & 50 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & 10 \\ \infty & \infty & \infty & 20 & \infty & 60 \\ \infty & \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

迪杰斯特拉算法实现--S中的点的表示

$$\begin{bmatrix} \infty & \infty & 10 & \infty & 30 & 100 \\ \infty & \infty & 5 & \infty & \infty & \infty \\ \infty & \infty & \infty & 50 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & 10 \\ \infty & \infty & \infty & 20 & \infty & 60 \\ \infty & \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

- 方法一:
  - 设一个一维数组int final[max];
  - final[i]=1表示从源点到顶点i的最短路径已经求出，i在S中
  - final[i]=0表示从源点到顶点i的最短路径尚未求出，i在V-S中
- 方法二:
  - 利用邻接矩阵主对角线的位置G.arcs[i][i]表示i是否在S中
  - G.arcs[i][i]=1表示从源点到顶点i的最短路径已经求出，i在S中
  - G.arcs[i][i]=0表示从源点到顶点i的最短路径尚未求出，i在V-S中

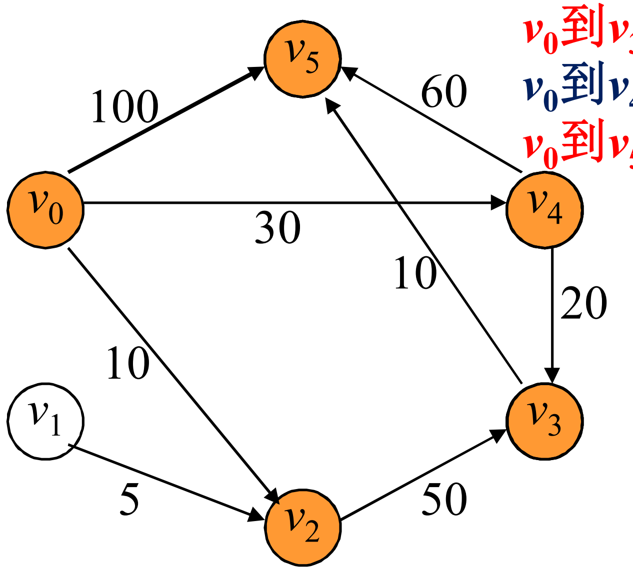
迪杰斯特拉算法实现--最短路径的表示

- 一维数组int D[max]表示最短路径的长度
  - D[i]： 从源点到点v_i 的最短路径的长度
  - 初态为：若从源点到v_i有弧，则D[i]为弧上的权值；否则置D[i]为∞，即：D[i]=G.arcs[k][i]; //k为源点
- 二维数组int p[max][max]表示最短路径包含的顶点
  - p[i][]： 从源点到点v_i 的最短路径
  - p[i][j]=0： v_j不在从源点到点v_i 的最短路径上
  - p[i][j]=1： v_j位于从源点到点v_i 的最短路径上。
  - 说明：1.书上的这种表示最短路径的方式只是给出了最短路径经过的顶点有哪些，没有给出这些顶点在这条路径上的顺序。2.也可采用其他的方式存储表示最短路径----例如利用最短路径定理，存放终点前一步的位置

迪杰斯特拉算法实现--最短路径的表示

- 二维数组int p[max][max]表示最短路径包含的顶点
  - p[i][]：从源点到点v_i的最短路径
  - p[i][j]=0：v_j不在从源点到点v_i的最短路径上
  - p[i][j]=1：v_j位于从源点到点v_i的最短路径上。

v_0到v_1的最短路径:无穷
v_0到v_2的最短路径:10, v_0 -> v_2
v_0到v_3的最短路径:50, v_0 -> v_4 -> v_3
v_0到v_4的最短路径:30, v_0 -> v_4
v_0到v_5的最短路径:60, v_0 -> v_4 -> v_3 -> v_5



p[5][0]=1;	p[3][0]=1;
p[5][1]=0;	p[3][1]=0;
p[5][2]=0;	p[3][2]=0;
p[5][3]=1;	p[3][3]=1;
p[5][4]=1;	p[3][4]=1;
p[5][5]=1;	p[3][5]=0;

算法描述(v_k为源点)

(1) 初始化操作:

D[i]=G.arcs[k][i]; D[k]=0;

final[i]=0 ($v_i \in V$); final[k]=1;

p[i][j]=0; $v_i,v_j \in V$;

p[i][i]=1; p[i][k]=1;

(2) 选择j，使得 D[j]=Min{D[i] | $v_i \in V-S$};

S＝S∪{ j }; // final[j]=1

(3) 修改从k出发到集合V-S上任一顶点的最短路径长度:

若D[j]+ G.arcs[j][w]<D[w]，则D[w]=D[j]+ G.arcs[j][w]

(4) 重复操作(2),(3)最多n-1次

```
void ShortestPath(AGraphs G,int k,int P[ ][ ], int D[ ])
{   int i,w, j,min;
   for (i=0;i<G.vexnum; i ++)
   {   final[i]=0; D[i]=G.arcs[k][i];
       for(w=0;w<G.vexnum; w ++) P[i][w]=0;
          if (D[i]<INFINITY) {   P[i][k]=1; P[i][i]=1; }
   }
   D[k]=0; final[k]=1;
   for(i=1; i<G.vexnum; i ++)
   {   min=INFINITY;
       for (w=0;w<G.vexnum; w ++)
          if (!final[w]&&D[w]<min) {j=w; min=D[w];}
       if(min== INFINITY) return;
       final[j]=1;
       for(w=0;w<G.vexnum; w ++)
        if(!final[w]&&(min+G.arcs[j][w]<D[w]))
            {   D[w]=min+G.arcs[j][w];
               P[w]=P[j]; P[w][w]=1;    }
   }
}
```

弗洛伊德算法

- 求每一对顶点之间的最短路径
- 从 v_i 到 v_j 的所有可能存在的路径中，选出一条长度最短的路径。
  - 若$\langle v_i,v_j \rangle$存在，则存在路径(v_i,v_j)
  - 若$\langle v_i,v_0 \rangle,\langle v_0, v_j \rangle$存在，则存在路径$(v_i,v_0, v_j)$
  - 若$(v_i,\ldots,v_1), (v_1,\ldots, v_j)$存在，则存在一条路径$(v_i, \ldots, v_1, \ldots v_j)$
  - …
  - 依次类推，则 v_i 至 v_j的最短路径应是上述这些路径中，路径长度最小者。

弗洛伊德算法--顶点v_i到顶点v_j的最短路径

- 如果<v_i,v_j>∈E(G)，则从v_i到v_j存在一条路径（v_i,v_j）；
- 该路径是否为最短路径尚需进行*n*次试探。
- 首先考虑路径（v_i,v_0,v_j），若其存在，比较路径（v_i,v_0,v_j）和路径（v_i,v_j）的长度，取其中较小者为从v_i到v_j的中间顶点序号不大于0的最短路径；
- 在路径上再加一个顶点v_1，若（v_i，…，v_1）和（v_1，…，v_j）分别是当前找到的中间顶点的序号不大于1最短路径，那么将（v_i，…，v_1，…，v_j）和已找到的中间结点的序号不大于0的最短路径比较，取其中较小的为从v_i到v_j的中间顶点序号不大于1的最短路径；
- 再增加一个顶点v_2，继续进行试探，依次类推。经过*n*次试探后可求得从顶点v_i到顶点v_j的最短路径。

弗洛伊德算法

图采用邻接矩阵形式存放

```
#define max 100
typedef struct {
            int arcs[max][max];
            int vexnum, arcnum;}AGraphs;

Agraphs G;
```

弗洛伊德算法

- 弗洛伊德算法递推地产生一个n阶矩阵序列:

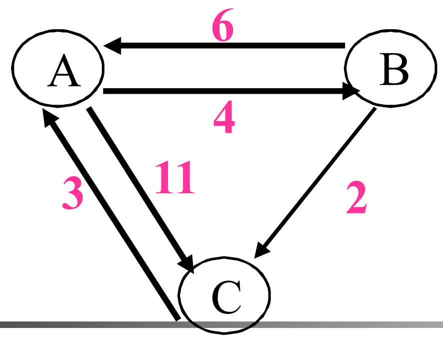
  $D^{(-1)}, D^{(0)}, \ldots, D^{(k)}, \ldots, D^{(n-1)}$

- 其中 $D^{(-1)}[i][j]=G.arcs[i][j];$

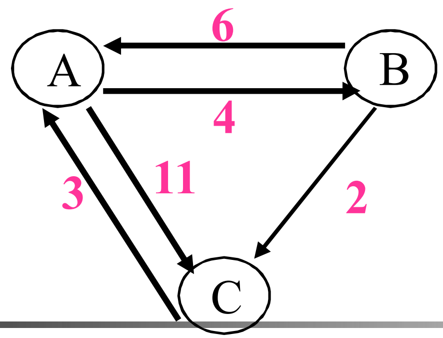
  $D^{(k)}[i][j]=\min\{D^{(k-1)}[i][j], D^{(k-1)}[i][k]+D^{(k-1)}[k][j]\}\ (0\leqslant k\leqslant n-1)$

- $D^{(k)}[i][j]$表示从v_i到v_j的中间顶点序号不大于k的最短路径的长度；$D^{(n-1)}[i][j]$表示从v_i到v_j的最短路径的长度
- P[i][j][]表示从v_i到v_j的路径.
  - P[i][j][x]=1表示从v_i到v_j的路径有顶点x
  - P[i][j][x]=0表示从v_i到v_j的路径没有顶点x

```
void s1(int D[ ][ ],int p[ ][ ][ ], Agraphs G)
{     int i,j,k;
      for(i=0;i<G.vnum;i++)
      for(j=0;j<G.vnum;j++)
      {     D[i][j]=G.arcs[i][j];
            for(k=0;k<G.vnum;k++) p[i][j][k]=0;
            if(D[i][j]<INFINITY)
              { p[i][j][i]=1;  p[i][j][j]=1;}
      for (k=0;k<G. vnum;k++)
        for (i=0;i<G. vnum;i++)
           for(j=0;j<G. vnum;j++)
           if(D[i][k]+D[k][j]<D[i][j])
            {D[i][j]=D[i][k]+D[k][j];
              for(w=0;w<G.vnum;w++)
              p[i][j][w]=p[i][k][w]||p[k][j][w];}
      }
```



$$\begin{bmatrix} 0 & 4 & 11 \\ 6 & 0 & 2 \\ 3 & \infty & 0 \end{bmatrix}$$



$$\begin{bmatrix} 0 & 4 & 11 \\ 6 & 0 & 2 \\ 3 & \infty & 0 \end{bmatrix}$$

$$D^{(-1)} = \begin{bmatrix} 0 & 4 & 11 \\ 6 & 0 & 2 \\ 3 & \infty & 0 \end{bmatrix} \quad D^{(0)} = \begin{bmatrix} 0 & 4 & 11 \\ 6 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix} \quad D^{(1)} = \begin{bmatrix} 0 & 4 & 6 \\ 6 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix} \quad D^{(2)} = \begin{bmatrix} 0 & 4 & 6 \\ 5 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix}$$

$$P^{(-1)} = \begin{bmatrix} & AB & AC \\ BA & & BC \\ CA & & \end{bmatrix} \quad P^{(0)} = \begin{bmatrix} & AB & AC \\ BA & & BC \\ CA & CAB & \end{bmatrix} \quad P^{(1)} = \begin{bmatrix} & AB & ABC \\ BA & & BC \\ CA & CAB & \end{bmatrix} \quad P^{(2)} = \begin{bmatrix} & AB & ABC \\ BCA & & BC \\ CA & CAB & \end{bmatrix}$$