8.6 最短路径

8.6.1 路径的概念

考虑带权有向图，把一条路径（仅仅考虑简单路径）上所经边的权值之和定义为该路径的路径长度或称带权路径长度。

$$v \xrightarrow{c_1} v_1 \xrightarrow{c_2} v_2 \xrightarrow{c_3} \cdots \xrightarrow{c_m} u$$

路径长度=$c_1 + c_2 + \cdots + c_m$

路径：（v，v_1，v_2，$\cdots$，u）

从源点到终点可能不止一条路径，把路径长度最短的那条路径称为最短路径。

8.6.2 从一个顶点到其余各顶点的最短路径

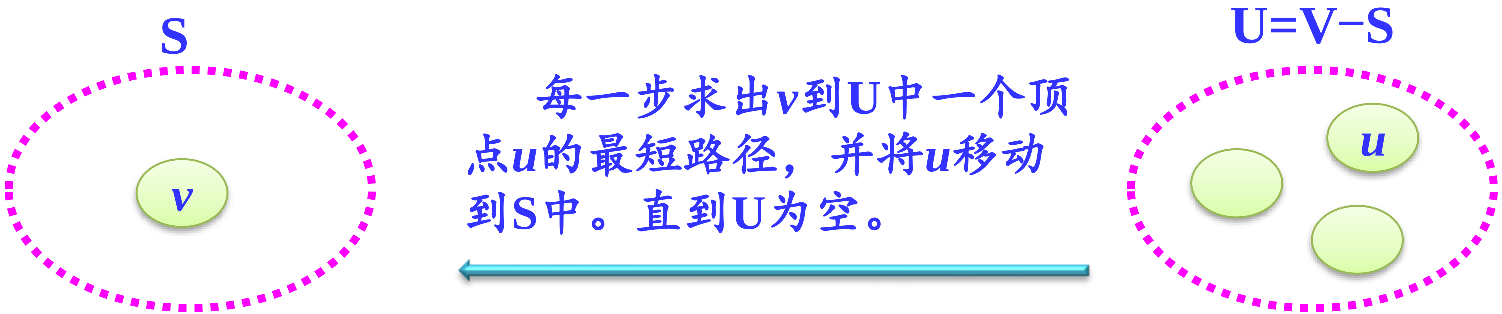
问题描述：给定一个带权有向图G与源点v，求从v到G中其他顶点的最短路径，并限定各边上的权值大于或等于0。

单源最短路径问题：Dijkstra算法

狄克斯特拉（Dijkstra）求解思路

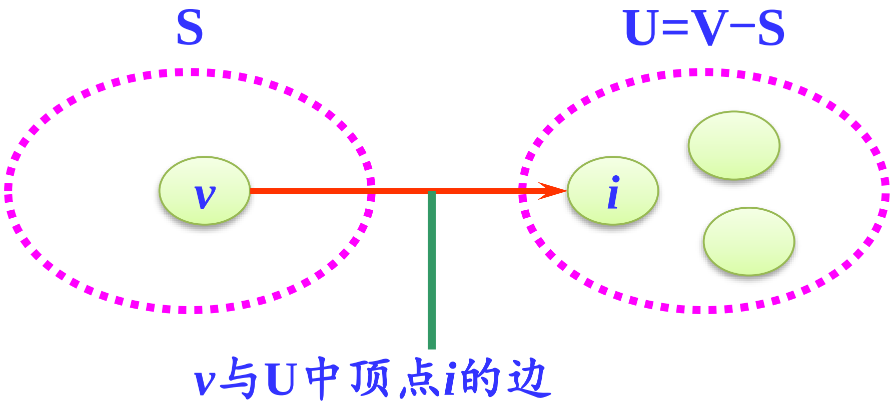
设G=(V，E)是一个带权有向图，把图中顶点集合V分成两组：

- 第1组为已求出最短路径的顶点集合（用S表示，初始时S中只有一个源点，以后每求得一条最短路径*v*，…，*u*，就将*u*加入到集合S中，直到全部顶点都加入到S中，算法就结束了）。
- 第2组为其余未求出最短路径的顶点集合（用U表示）。

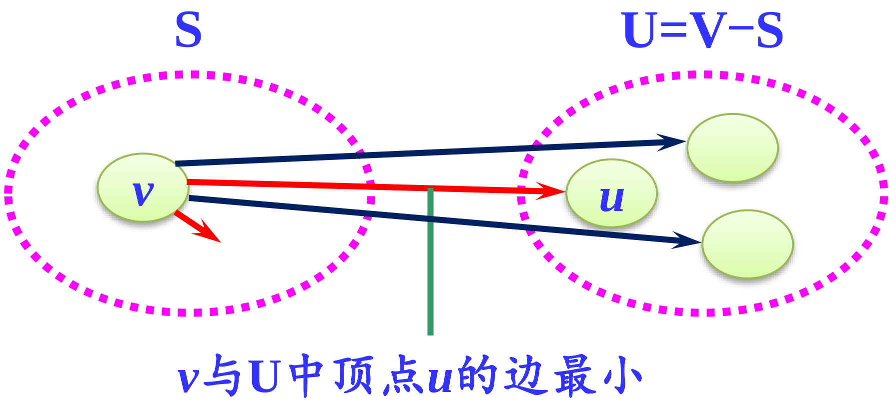


狄克斯特拉算法的过程

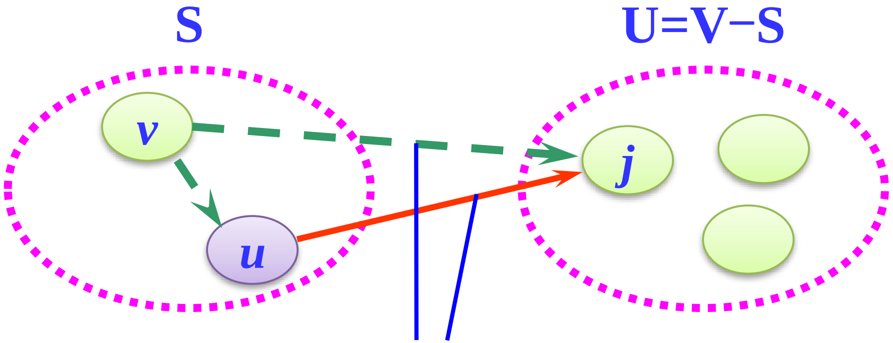
（1）初始化：S只包含源点即S={*v*}，*v*的最短路径为0。U包含除*v*外的其他顶点，U中顶点*i*距离为边上的权值（若*v*与*i*有边<*v*，*i*>）或∞（若*i*不是*v*的出边邻接点）。



（2）从U中选取一个距离*v*最小的顶点*u*，把*u*加入S中（该选定的距离就是*v* ⇨ *u*的最短路径长度）。



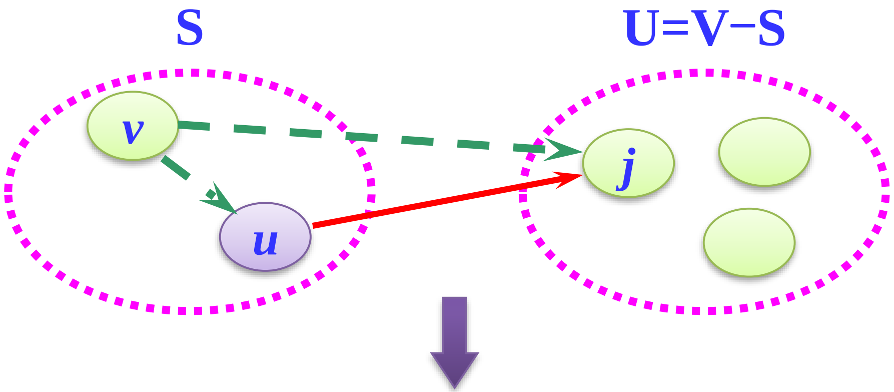
（3）以u为新考虑的中间点，修改U中各顶点j的最短路径长度：若从源点v到顶点j（$j \in U$）的最短路径长度（经过顶点u）比原来最短路径长度（不经过顶点u）短，则修改顶点j的最短路径长度。



两条路径进行比较：

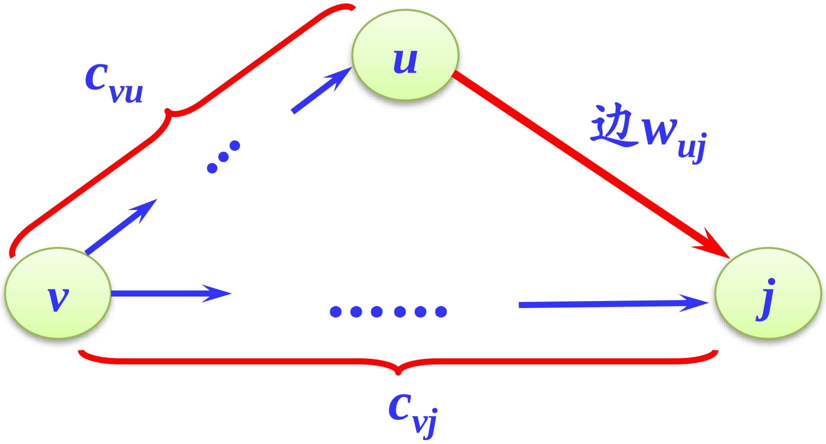
若经过u的最短路径长度更短，则修正

修改方式



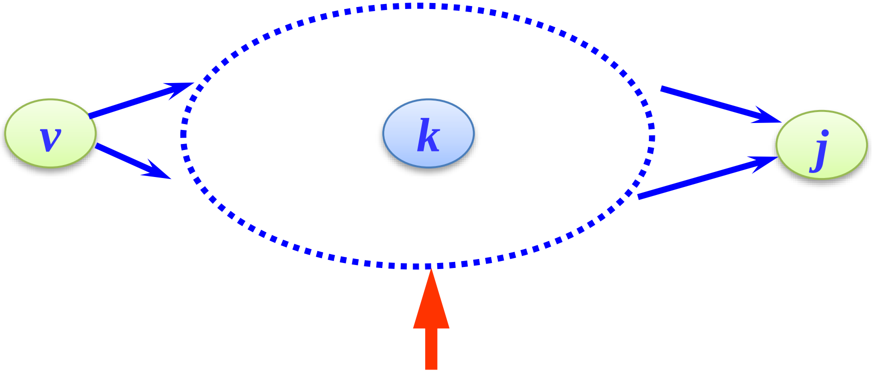
v ⇨ j的路径：

- 不经过顶点u
- 经过顶点u



顶点v ⇨ j的最短路径长度＝MIN($c_{vk}+w_{kj}$， c_{vj})

（4）重复步骤（2）和（3）直到所有顶点都包含在S中。



考虑中间其他所有顶点*k*，通过比较得到*v* ⇨ *j*的最短路径

算法设计（解决2个问题）

- 如何存放最短路径长度：

用一维数组dist[*j*]存储！

源点*v*默认，dist[*j*]表示源点 ⇨ 顶点*j*的最短路径长度。如dist[2]=12表示源点 ⇨ 顶点2的最短路径长度为12。

- 如何存放最短路径：

从源点到其他顶点的最短路径有*n*−1条，一条最短路径用一个一维数组表示，如从顶点0 ⇨ 5的最短路径为0、2、3、5，表示为

path[5]={0,2,3,5}。

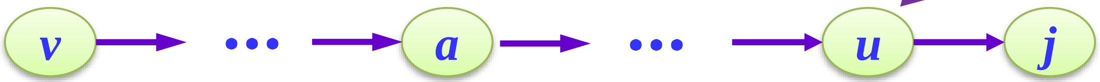
所有*n*−1条最短路径可以用二维数组path[][]存储。

?

改进的方法是采用一维数组**path**来保存：

若从源点v ⇨ j的最短路径如下：

v ⇨ j最短路径中j的前一个顶点

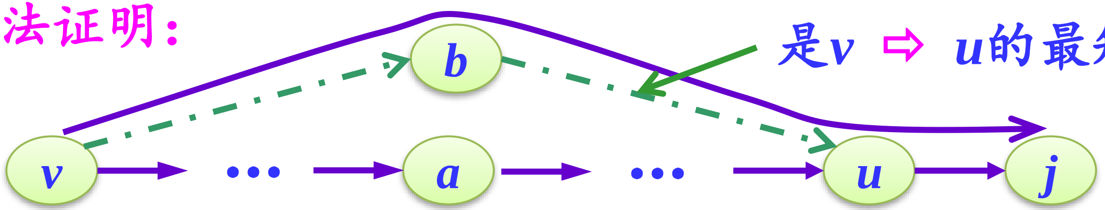


则

一定是从源点v ⇨ u的最短路径 ？

反证法证明：

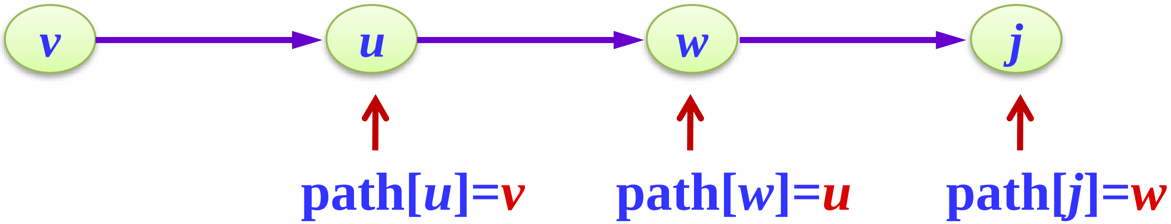
是v ⇨ u的最短路径



而通过b的路径更短，则$v \to \cdots a \to \cdots \ u \to j$不是最短路径

与假设矛盾，问题得到证明。

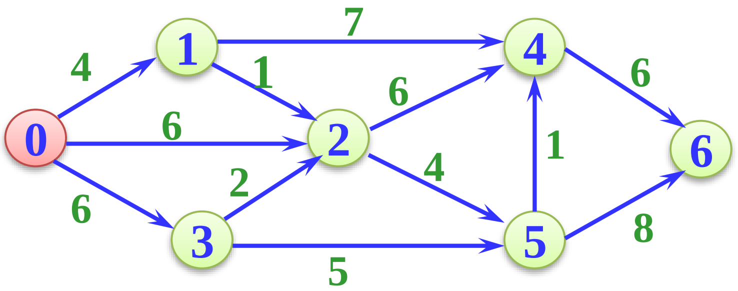
v ⇨ j的最短路径:



从path[j]推出的逆路径：j, w, u, v

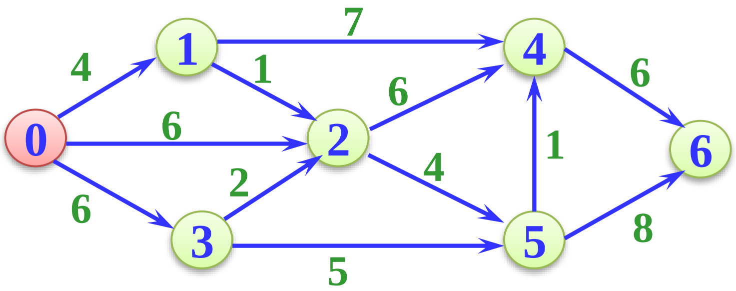
对应的最短路径为：$v \rightarrow u \rightarrow w \rightarrow j$

Dijkstra算法示例演示



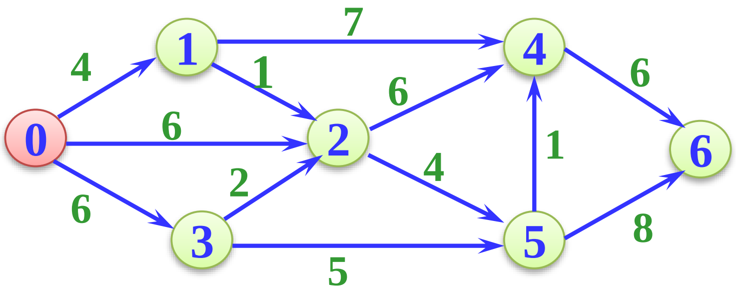
S	U	dist[]	path[]
		0 1 2 3 4 5 6	0 1 2 3 4 5 6
{0}	{1,2,3,4,5,6}	{0, 4, 6, 6, ∞, ∞, ∞}	{0, 0, 0, 0, -1, -1, -1}
		最小的顶点：1	
{0,1}	{2,3,4,5,6}	{0, 4, 5, 6, 11, ∞, ∞}	{0, 0, 1, 0, 1, -1, -1}
		最小的顶点：2	
{0,1,2}	{3,4,5,6}	{0, 4, 5, 6, 11, 9, ∞}	{0, 0, 1, 0, 1, 2, -1}

Dijkstra算法示例演示



S	U	dist[]	path[]
		0 1 2 3 4 5 6	0 1 2 3 4 5 6
{0,1,2}	{3,4,5,6}	{0, 4, 5, 6, 11, 9, ∞}	{0, 0, 1, 0, 1, 2, -1}
		最小的顶点：3	
{0,1,2,3}	{4,5,6}	{0, 4, 5, 6, 11, 9, ∞}	{0, 0, 1, 0, 1, 2, -1}
		最小的顶点：5	
{0,1,2,3,5}	{4,6}	{0, 4, 5, 6, 10, 9, 17}	{0, 0, 1, 0, 5, 2, 5}

Dijkstra算法示例演示



S	U	dist[]	path[]
		0 1 2 3 4 5 6	0 1 2 3 4 5 6
{0,1,2,3,5}	{4,6}	{0, 4, 5, 6, 10, 9, 17}	{0, 0, 1, 0, 5, 2, 5}
		最小的顶点：4	
{0,1,2,3,5,4}	{6}	{0, 4, 5, 6, 10, 9, 16}	{0, 0, 1, 0, 5, 2, 4}
		最小的顶点：6	
{0,1,2,3,5,4,6}	{}	{0, 4, 5, 6, 10, 9, 16}	{0, 0, 1, 0, 5, 2, 4}

最终结果

狄克斯特拉算法如下（v为源点编号）：

```
void Dijkstra(MGraph g,int v)
{    int dist[MAXV],path[MAXV];
     int s[MAXV];
     int mindis,i,j,u;

     for (i=0;i<g.n;i++)
     {      dist[i]=g.edges[v][i];         //距离初始化
            s[i]=0;                         //s[]置空
            if (g.edges[v][i]<INF)          //路径初始化
                 path[i]=v;                 //顶点v到i有边时
            else
                 path[i]=-1;                //顶点v到i没边时
     }

     s[v]=1;                                //源点v放入S中
```

dist和path数组初始化

```
for (i=0;i<g.n;i++)                          //循环n-1次
{     mindis=INF;
      for (j=0;j<g.n;j++)
          if (s[j]==0 && dist[j]<mindis)
          {      u=j;
                 mindis=dist[j];
          }
      s[u]=1;                                  //顶点u加入S中
      for (j=0;j<g.n;j++)                      //修改不在s中的顶点的距离
          if (s[j]==0)
              if (g.edges[u][j]<INF && dist[u]+g.edges[u][j]<dist[j])
              {     dist[j]=dist[u]+g.edges[u][j];
                    path[j]=u;
              }
}
Dispath(dist,path,s,g.n,v);                    //输出最短路径
}
```

找最小路径长度顶点*u*

调整

算法的时间复杂度为$O(n^2)$。

利用dist和path求最短路径长度和最短路径

❶ 求0 ⇨ 6的最短路径长度：

0	1	2	3	4	5	6

dist={0, 4, 5, 6, 10, 9, 16}

从顶点0 ⇨6的最短路径长度为16

❷ 求0 ⇨ 6的最短路径：

0	1	2	3	4	5	6

path={0, 0, 1, 0, 5, 2, 4}

path[6]=4

path[4]=5

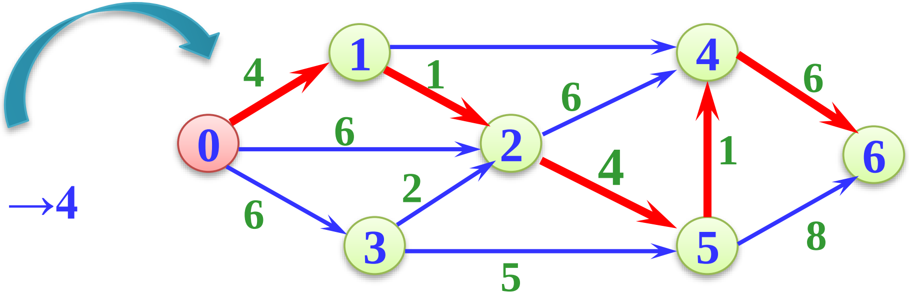
path[5]=2

path[2]=1

path[1]=0到源点

最短路径为：

0→1 →2 →5 →4



观察求解结果

源点v=0

S={0, 1, 2, 3, 5, 4, 6}

4

5

6

9

10

16

源点到各个顶点的最短路径长度

递增

0	1	2	3	4	5	6

dist={0, 4, 5, 6, 10, 9, 16}

结论：

❶ 按顶点进入S的先后顺序，最短路径长度越来越长。

❷ 一个顶点一旦进入S后，其最短路径长度不再改变（调整）。

思考题：

Dijkstra算法为什么不适合负权值的情况？

思考题

狄克斯特拉算法可以用于带权无向图求最短路径吗？

——本讲完——