图论及其应用

北京邮电大学理学院

Ch2 最短路问题

Ch2 主要内容

- 最短路问题
- **Dijkstra 算法**
- **Bellman-Ford算法**
- **Floyd-Warshall算法**
- 最短路问题的应用

2.1 最短路问题与Dijkstra 算法

- 赋权图（weighted graph）G　(注：权 ≡ 1 时即为上文中所提的图。)
- 权(weight) w(e)　$\forall e \in E(G)$

记号：w(H) = $\sum_{e \in E(H)} w(e)$ ，　H ⊆ G .

- 路P的长 = w(P)，区别于前面1.4讲的路长
- 顶点u与v的 **距离** d(u, v) = 最短(u, v)-路的长。
- 最短路问题：求u到v的最短路d(u, v) 。

- 最短路问题中，若存在负圈，即，权重为负数的圈，不考虑。
- 无向图中，只要有边的权重是负数，就认为图中有负圈，不考虑。
- 无负权重的无向图/无负圈的有向图中允许：
  环（正权重）、 重边（弧）、
- 最短路只考虑简单图、严格有向图。

Dijkstra算法（狄克斯特拉算法）

- 1959年Dijkstra提出，目前被公认为是最好的方法
- 适用范围：权重都正w(e)>0；

若某个w(uv)=0，则合并uv为一个顶点

(所有与u和v关联的边都变成和新点关联，再减掉可能出现的重边，只保留权重最小的边）

若uv之间没有边，则w(uv)=∞。

Dijkstra算法（狄克斯特拉算法）

- 原理：

　若 $P = u_0 v_1 v_2 \cdots v_{k-1} v_k$ 是从点u_0到v_k 的最短路；则 $P' = u_0 v_1 v_2 \cdots v_{k-1}$ 必是点u_0到v_{k-1} 的最短路。

- 问题　求最短(u_0, v_0)-路。
- 转　求最短(u_0, **v**)-路，　$\forall$ v $\in$ V \ {u_0}，由近及远逐次求最短路d(u_0, **v**)

Dijkstra算法原理：

- 逐步求出顶点序列被定义为 u_1，u_2, ...u_{k-1}

使 $d(u_0, u_1) \leq d(u_0, u_2) \leq \ldots d(u_0, u_{k-1})$.

记 $S_0 = \{ u_0 \}$,

$S_k = \{ u_0, u_1, \ldots, u_k \}$,　　$\overline{S}_k = V \setminus S_k$ 。

P_i 为最短(u_0, u_i)-路,　　$i = 1, 2, \ldots k$

所以点 u_{k+1} 是点 u_0 到 $\overline{S}_k$ 中距离最短的顶点。

(1).求u_1： u_1是使

$w(u_0u_1) = \min\{ w(u_0v) \mid v \neq u_0 \}$者.

令 $S_1 = S_0 \cup \{u_1\}$， $P_1 = u_0u_1$.

(2). 若已求得S_k； $d(u_0, u_1)$， …， $d(u_0, u_k)$；

及最短 (u_0, u_j)路 P_i ， $i=1,2,\ldots,k$ 。

求u_{k+1} ：

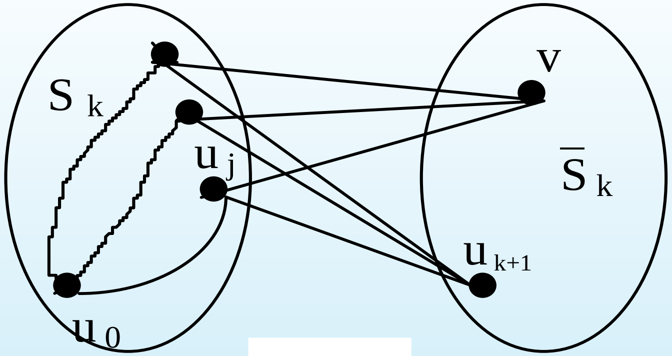


图 2-1

求u_{k+1} ：

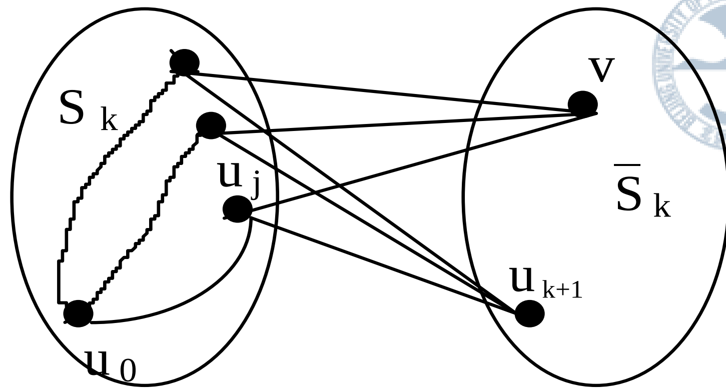


图 2-1

- $d(u_0, u_{k+1}) = \min\{\ d(u_0, v) \mid v \in \overline{S}_k\ \}$

  $= d(u_0, u_j) + w(u_j u_{k+1})$ 某 $j \in \{1, 2, \ldots, k\}$

  $= \min\{ d(u_0, u) + w(u u_{k+1}) \mid u \in S_k \}$

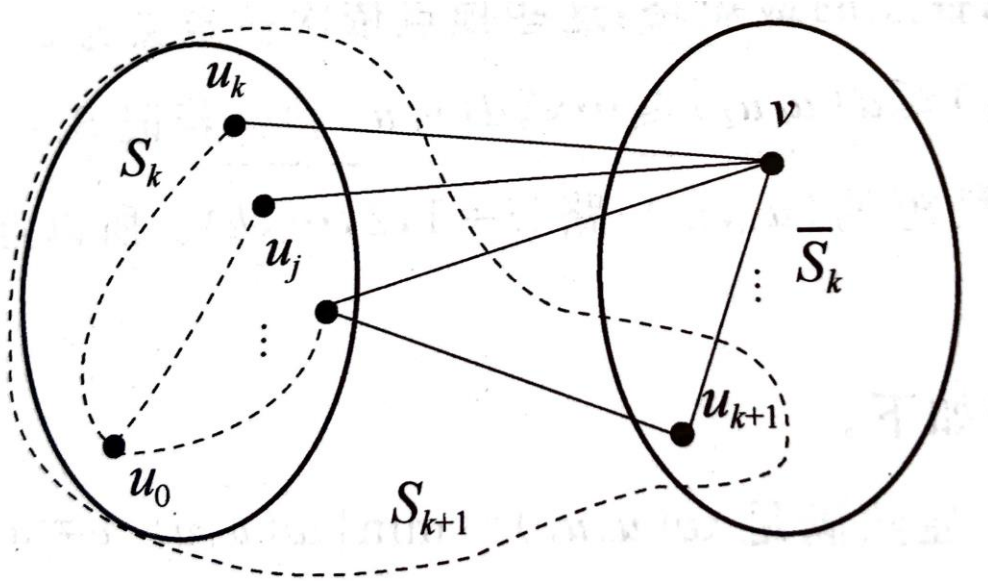
  $= \min\{ d(u_0, u) + w(uv) \mid u \in S_k, v \in \overline{S}_k \}$

  $= \min\{ l(v) \mid v \in \overline{S}_k \}$

其中， $l(v) = \min\{ d(u_0, u) + w(uv) \mid u \in S_k \}$

$(\ \therefore\ l(u_{k+1}) = d(u_0, u_{k+1})\ )$

- $S_{k+1} = S_k \cup \{ u_{k+1} \}$,
- $P_{k+1} = P_j u_j u_{k+1}$ 某 $j \in \{1, 2, \ldots, k\}$。



update 进行下一 步时，只要更新 $\overline{S_{k+1}}$ 中 点的l(v)即可：

$$l(v) \leftarrow \min\{\, l(v),\ l(u_{k+1}) + w(u_{k+1}v) \,\}$$

对 ∀ v ∈ $\overline{S_{k+1}}$。

会节省计算量

Dijkstra算法

Pr(v) :表示求得的（ u_0, v)-最短路上v点前的点；

- (1). 初始化：$l(u_0) \leftarrow 0$; $Pr(u_0) = u_0$, $P_0 = u_0$;

  $l(v) \leftarrow \infty$; $\forall v \neq u_0$; $Pr(v) = v_{v+1}$

  $S_0 \leftarrow \{u_0\}$; $k \leftarrow 0$.

- (2). （这时已有$S_k = \{u_0, u_1, \ldots, u_k\}$, $d(u_0, u_j)$, P_j ）

  $l(v) \leftarrow \min\{l(v), l(u_k) + w(u_k v)\}$ 和 **Pr(v) ?** $\forall v \in \overline{S}_k$

  再计算 $\min\{l(v)\}$，设其最小值点为 u_{k+1}，

  令 $S_{k+1} = S_k \cup \{u_{k+1}\}$。

- (3).若 $k=\nu-2$, 停止；不然，令$k \leftarrow k+1$,并回到(2)。

$\bar{S}$

计算复杂性

加法： $\nu(\nu-1)/2$

比较： $\nu(\nu-1)/2 \times 2$

$v \in \bar{S}$： 至多$(\nu-1)$^2

+) ______________________________

共 $O(\nu^2)$

凡是复杂性为 p(ν, ε) 的算法（ p(. , .) 为一多项式）称为 **“好算法”**（“good algorithm”------J.Edmonds）。这是相对于**指数型算法**而言的：在10^{-6}秒/步运算速度下：

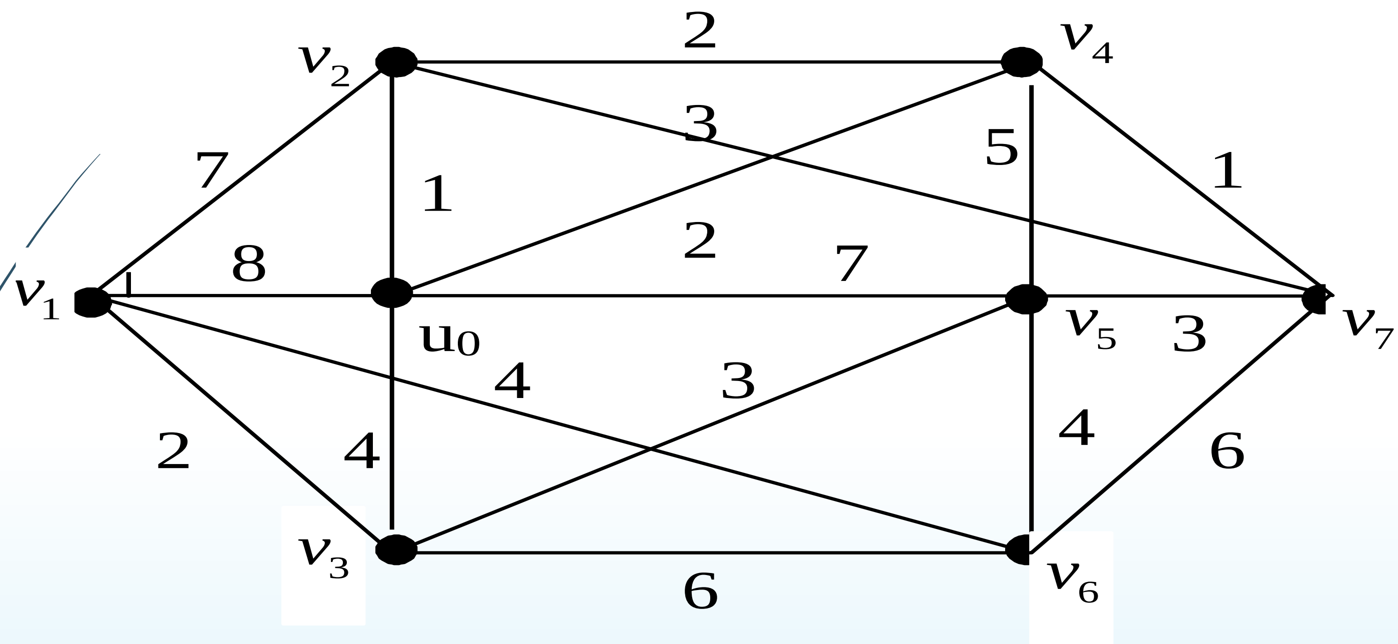
复杂性	n=10	20	30	40	50
n^3	.001sec	.008sec	.027sec	.064sec	.125sec
n^5	.1sec	3.2sec	24.3sec	1.7min	5.2min
2^n	.001sec	1.0sec	17.9min	12.7days	35.7 years

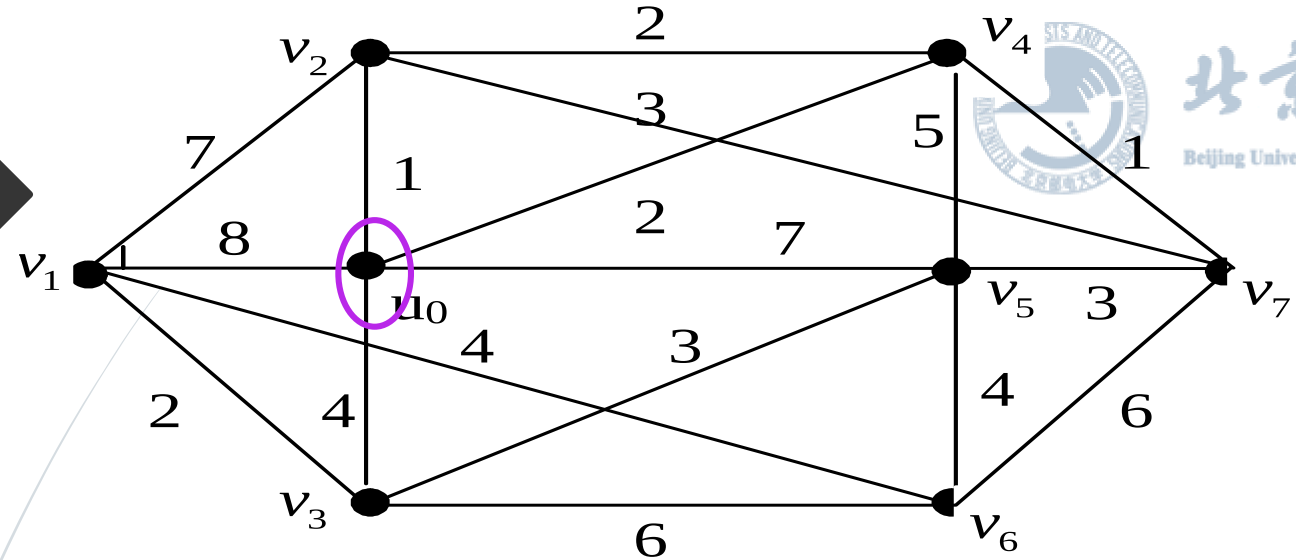
由上表可见，两种算法有天壤之别。

注

- 1.若只关心求$d(u_0,v_0)$, 则算法进行到$v_0 \in S_k$时停止。
- 2.计算过程中，每步所得子图都是一 棵树（？：每步都是往其上增加一条边及一 个顶点）。因此该过程称为 **tree growing procedure** 。在该树中的(u_0,v_0)-路，是原图中最短(u_0,v_0)-路 。
- 3.若要计录u_0到每个顶点u的最短路，只要记录该路中u的前一个顶点（即该树中u的父亲）即可。

例：求下面赋权图中点u_0 到其他各点的距离及最短路线。





解：（0）令 $l(u_0):=0$，$\Pr(u_0)=u_0$，$l(v_i):=\infty$，$\Pr(v_i)=v_9$（$i=1,2,\cdots,7$），$S_0:=\{u_0\}$

$i:=0$。

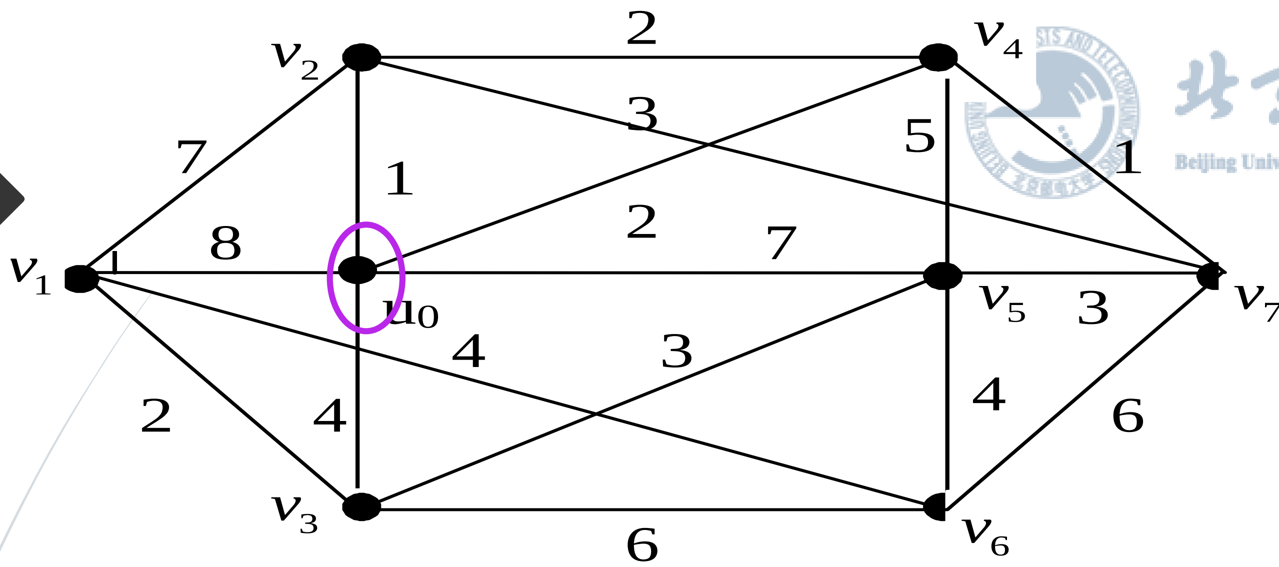
（1）对 $v\in\overline{S}_i$，计算 $l(v):=\min\{l(v),l(u_i)+w(u_iv)\}$ 和 $\Pr(v)$：

$$l(v_4):=\min\{l(v_4),l(u_0)+w(u_0v_4)\}=\min\{\infty,0+2\}=2\text{，}\quad \Pr(v_4)=u_0$$

$$l(v_5):=\min\{l(v_5),l(u_0)+w(u_0v_5)\}=\min\{\infty,0+7\}=7\text{，}\quad \Pr(v_5)=u_0$$

$$l(v_6):=\min\{l(v_6),l(u_0)+w(u_0v_6)\}=\min\{\infty,0+\infty\}=\infty\text{，}\quad \Pr(v_6)=v_9$$

$$l(v_7):=\min\{l(v_7),l(u_0)+w(u_0v_7)\}=\min\{\infty,0+\infty\}=\infty\text{，}\quad \Pr(v_7)=v_9$$

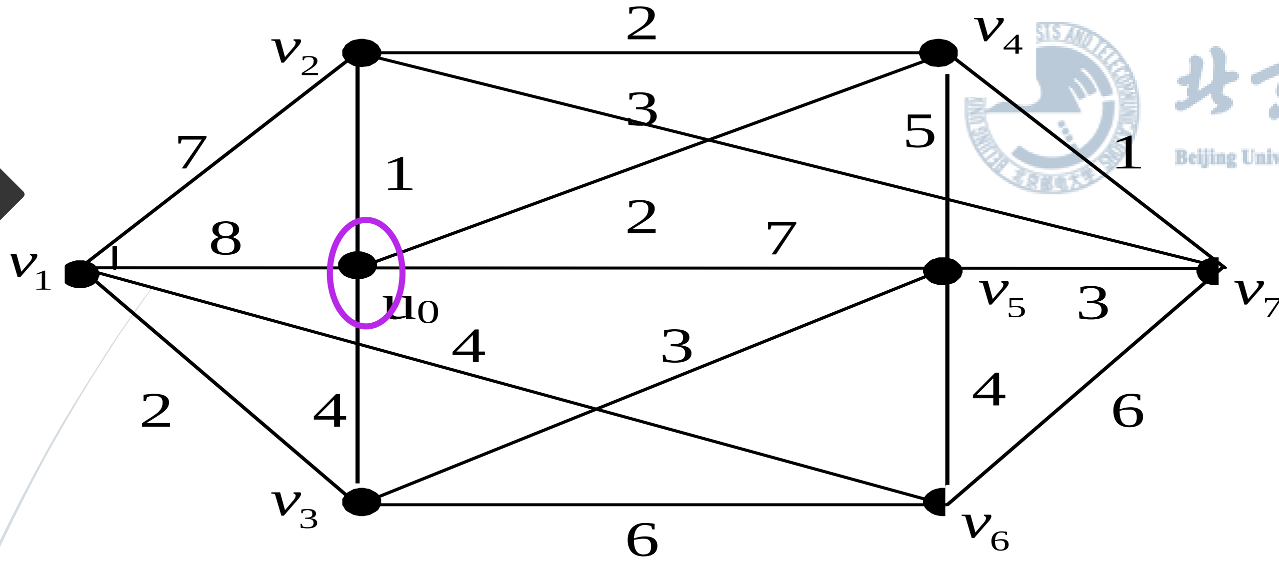


$$\min_{v \in \bar{S}_0}\{l(v)\} = \min\{8, 1, 4, 2, 7, \infty, \infty\} = 1 = l(v_2)\ ,$$

所以：

$$u_1 = v_2\ ,\ \ 令\ S_1 = S_0 \cup \{u_1\} = \{u_0, v_2\}\ ,\ \ P_1 = P_0 u_1 = u_0 u_1 = u_0 v_2$$

$$i = 0 < 6 = \nu - 2,\ 令\ i := 0 + 1 = 1\ 。$$



（2）对 $v \in \overline{S}_i$，计算 $l(v) := \min\{l(v), l(u_i) + w(u_i v)\}$ 和 $\Pr(v)$：

计算：

$$\min_{v \in \overline{S}_1}\{l(v)\} = \min\{8, 4, 2, 7, \infty, 4\} = 2 = l(v_4),$$

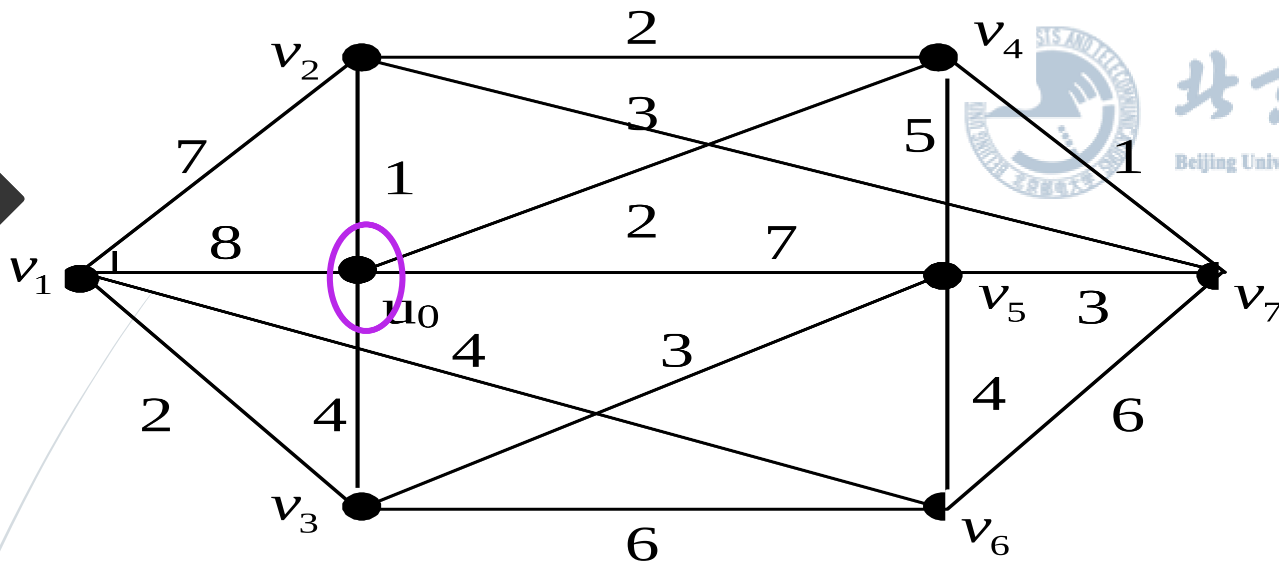
所以：

$$u_2 = v_4,\quad 令\ S_2 = S_1 \cup \{u_2\} = \{u_0, v_2, v_4\},\quad P_2 = P_0 u_2 = u_0 u_2 = u_0 v_4$$

$$i = 1 < 6 = \nu - 2,\ 令\ i := 1 + 1 = 2。$$

$$l(v_6) := \min\{l(v_6), l(u_1) + w(u_1 v_6)\} = \min\{\infty, 1 + \infty\} = \infty,\quad \Pr(v_6) = v_9$$

$$l(v_7) := \min\{l(v_7), l(u_1) + w(u_1 v_7)\} = \min\{\infty, 1 + 3\} = 4,\quad \Pr(v_7) = u_1$$



所有求得的短路线如图 2-4 中粗边所示（构成一棵树）。

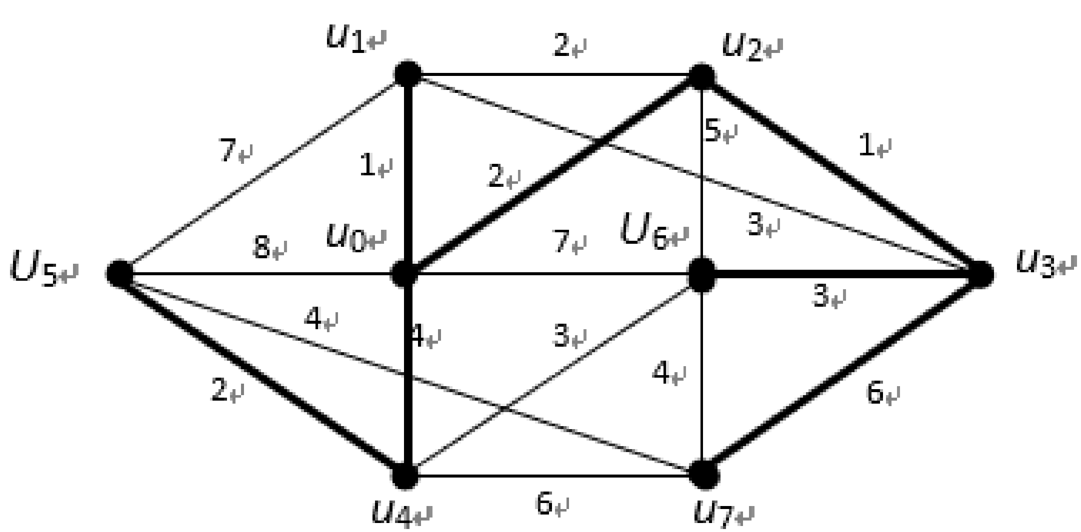
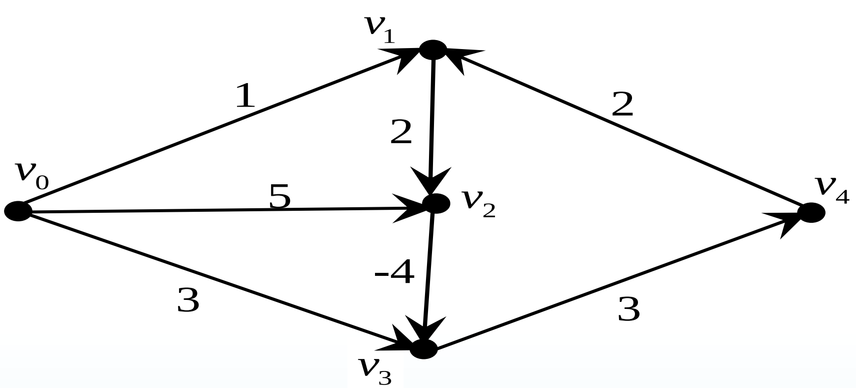
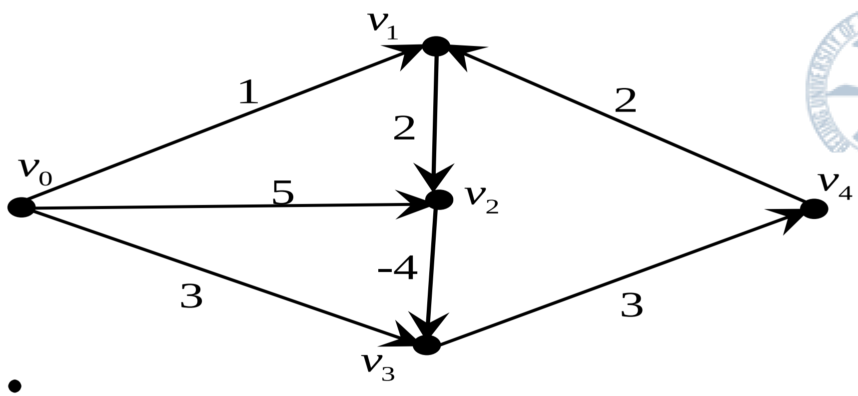


图 2-4

练

- 用Dijkstra算法求下面的赋权图中点v0到其他所有点的最短有向路的路长及路线。

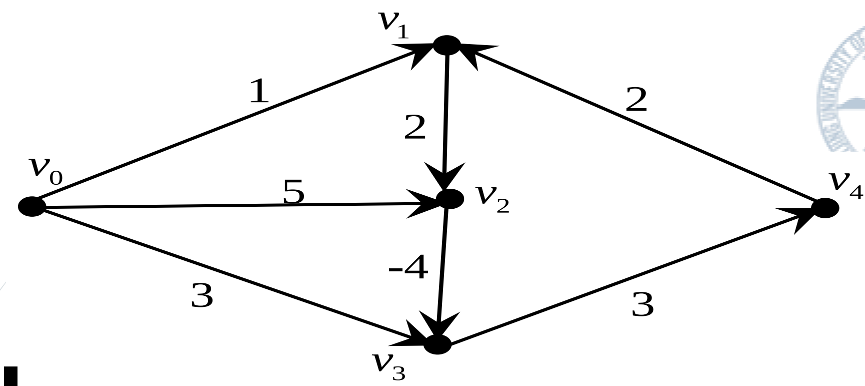




解法I:

v	$d(v_0, v_0)$	$d(v_0, v_1)$	$d(v_0, v_2)$	$d(v_0, v_3)$	$d(v_0, v_4)$
int	0	∞	∞	∞	∞
v_0	p	1	5	3	
v_1		p	3		
v_2			p	-1	
v_3				p	2
v_4					p

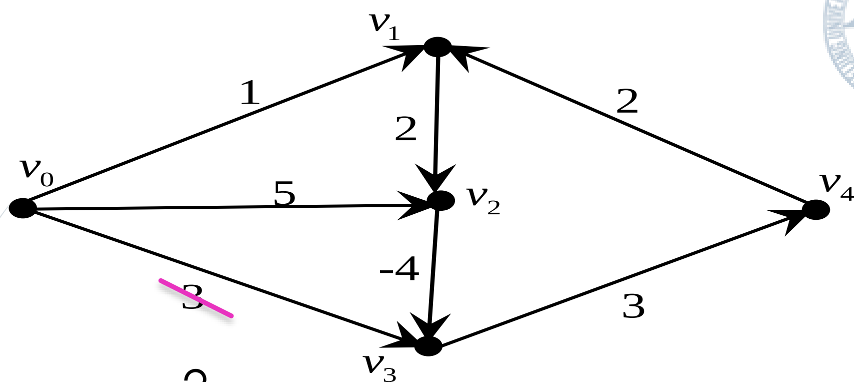
$$v_0 \to v_1 \to v_2 \to v_3 : \ -1$$



解法II：

v	$d(v_0,v_0)$	$d(v_0,v_1)$	$d(v_0,v_2)$	$d(v_0,v_3)$	$d(v_0,v_4)$
int	0	∞	∞	∞	∞
v_0	p	1	5	3	
v_1		p	3		
v_3				p	6
v_2			p		
v_4					p

$$v_0 \rightarrow v_3：\ 3$$



2

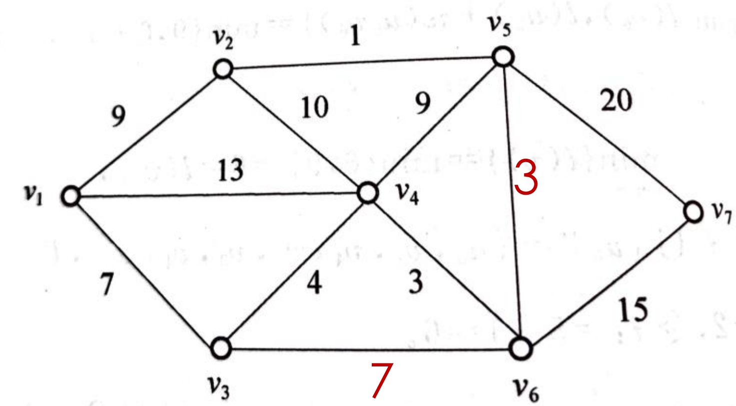
Dijkstra不适合求有负权重的图

v	$d(v_0,v_0)$	$d(v_0,v_1)$	$d(v_0,v_2)$	$d(v_0,v_3)$	$d(v_0,v_4)$
int	0	∞	∞	∞	∞
v_0	p	1	5	2	
v_1		p	3		
v_3				p	5
v_2			p	?	
v_4					p

Dijkstra求出：$v_0 \rightarrow v_3$ 的最短路长：**3**，路线是$v_0 \rightarrow v_3$；
但真实最短路是$v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow v_3$，长度**-1.**

习题2-1

- 1. 用**Dijkstra**算法求下图中从**v1**点到其他任意一点的最短路线及距离。



- 2. 要求出第1题中**v1**到每个点的所有最短路线，应该如何修改程序，复杂性如何？
- 3. 要求出第1题中**v1**到所有点的次短路线，应该如何修改程序，复杂性如何？

- **4.** 描述一个算法以确定

  (a). 判断一个图是否连通？

  （**b**）求出一个图的所有连通分支；

  (b). 求出一个图的围长（即最短圈的长）；