（1）求给定两结点间的最短通路—Dijkstra算法

如何求出简单无向赋权图G = <V, E>中从结点v_1到v_n的最短通路，目前比较好的算法是由Dijkstra在1959年提出的，称为Dijkstra算法，其基本思想是：

将结点集合V分为两部分：一部分称为具有P（永久性）标号的集合，另一部分称为具有T（暂时性）标号的集合。所谓结点v的P标号是指从v_1到v的最短通路的长度；而结点u的T标号是指从v_1到u的某条通路的长度。首先将v_1取为P标号，其余结点为T标号，然后逐步将具有T标号的结点改为P标号。当结点v_n也被改为P标号时，则找到了从v_1到v_n的一条最短通路。

算法9.3.1 Dijkstra算法

I. **初始化**：将v_1置为P标号，$d(v_1) = 0$，$P = \{v_1\}$，$v_i \in V$，$i \neq 1$，置v_i为T标号，即$T = V-P$且

$$d(v_i)=\begin{cases} w(v_1,v_i), & 若(v_1,v_i)\in E \\ \infty & 若(v_1,v_i)\notin E \end{cases}$$

II. **找最小**：寻找具有最小值的T标号的结点。若为v_k，则将v_k的T标号改为P标号，且$P = P\cup\{v_k\}$，$T = T-\{v_k\}$。

算法9.3.1 Dijkstra算法(续)

III. 修改：修改与v_k相邻的结点的T标号值。$v_i \in V$,

$$d(v_i)=\begin{cases} d(v_k)+w(v_k,v_i), & \text{若}\ d(v_k)+w(v_k,v_i)<d(v_i) \\ d(v_i) & \text{否则} \end{cases}$$

IV. 重复(II)和(III)，直到v_n改为P标号为止。

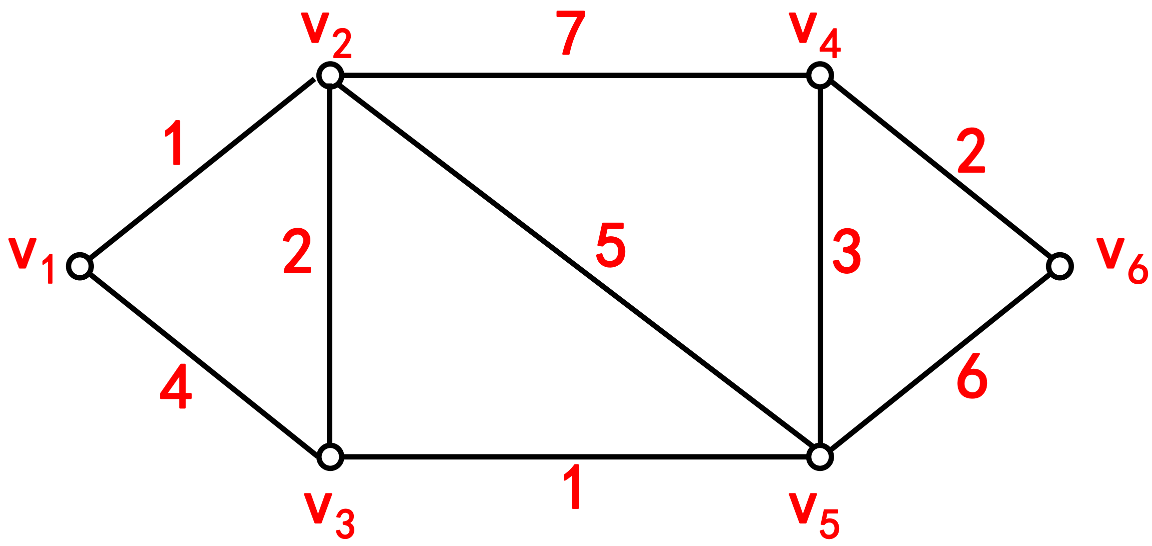
说明

当v_n归入P而正好P = V时，不仅求出了从v_1到v_n的最短通路，而且实际上求出了从v_1到所有结点的最短通路。

上述算法的正确性是显然的。因为在每一步，设P中每一结点的标号是从v_1到该结点的最短通路的长度（开始时，P = {v_1}，d(v_1) = 0，这个假设是正确的），故只要证明上述d(v_i)是从v_1到v_i的最短通路的长度即可。事实上，任何一条从v_1到v_i通路，若通过T的第一个结点是v_p，而$v_p \neq v_i$的话，由于所有边的长度非负，则这种通路的长度不会比d(v_i)小。

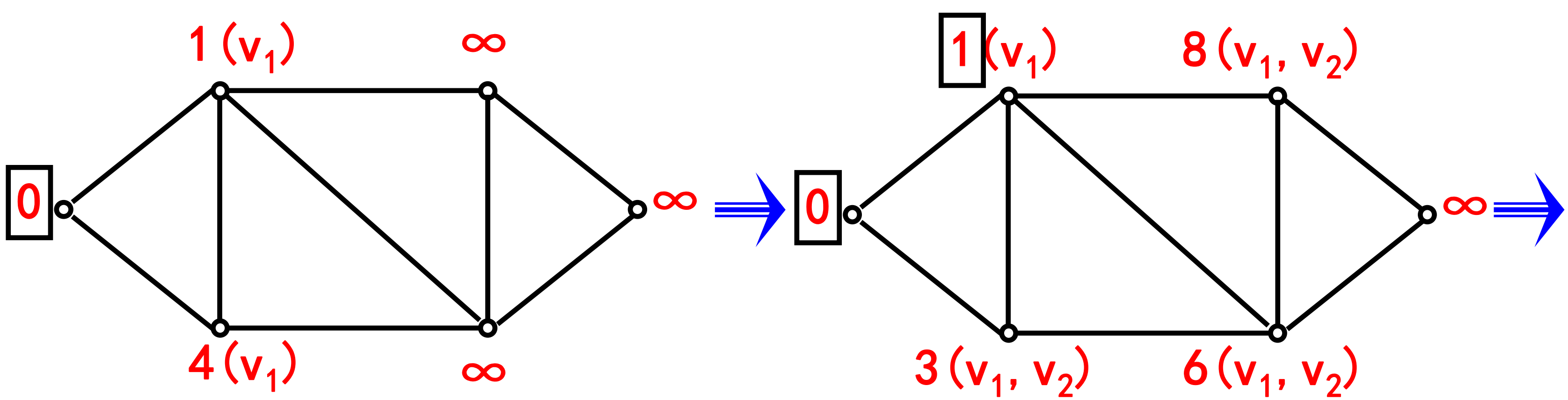
例9.3.9

试求简单无向赋权图中v_1到v_6的最短通路。

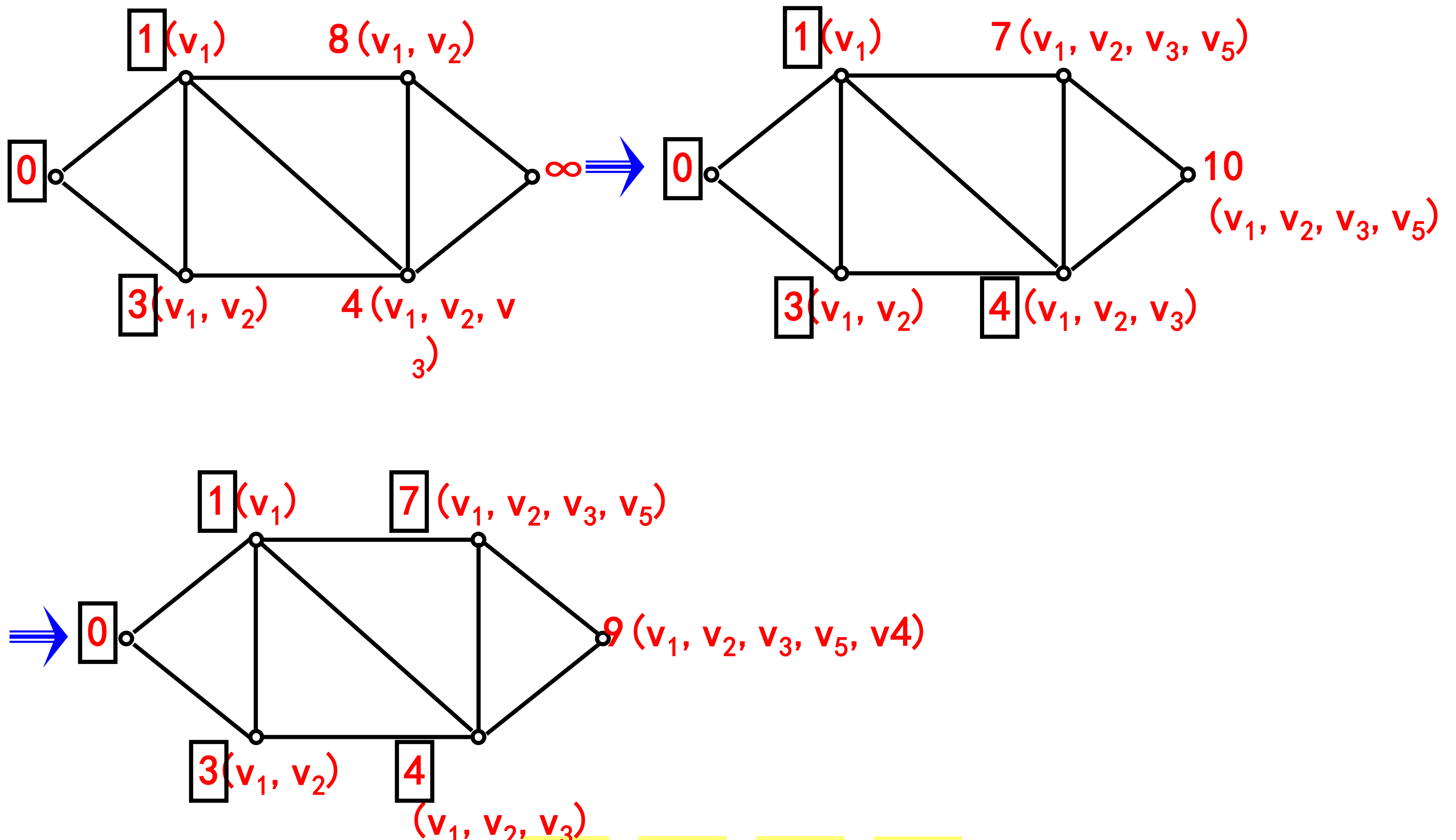


解

根据Dijkstra算法，有如下图所示的求解过程。故v_1到v_6的最短通路为$v_1v_2v_3v_5v_4v_6$，其长度为9。实际上，也求出了v_1到所有结点的最短通路，如v_1到v_5的最短通路为$v_1v_2v_3v_5$，其长度为4，等等。



解(续)


1 (v_1)
8 (v_1, v_2)
0
∞
3 (v_1, v_2)
4 (v_1, v_2, v_3)
1 (v_1)
7 (v_1, v_2, v_3, v_5)
0
10 (v_1, v_2, v_3, v_5)
3 (v_1, v_2)
4 (v_1, v_2, v_3)
1 (v_1)
7 (v_1, v_2, v_3, v_5)
0
9 (v_1, v_2, v_3, v_5, v4)
3 (v_1, v_2)
4 (v_1, v_2, v_3)