算法是一种使用全局信息的算法。说它是分布式的，是因为每个结点都要从一个或多个直接相连邻居接收某些信息，执行计算，然后将其计算结果分发给邻居。说它是迭代的，是因为此过程一直要持续到邻居之间无更多信息要交换为止。（有趣的是，此算法是自我终止的，即没有计算应该停止的信号，它就停止了。）说它是异步的，是因为它不要求所有结点相互之间步伐一致地操作。我们将看到一个异步的、迭代的、自我终止的、分布式的算法比一个集中式的算法要有趣得多。

在我们给出 DV 算法之前，有必要讨论一下存在于最低费用路径的费用之间的一种重要关系。令 $d_x(y)$ 是从结点 x 到结点 y 的最低费用路径的费用。则该最低费用与著名的 Bellman-Ford 方程相关，即

$$d_x(y) = \min_v\{c(x,v) + d_v(y)\} \tag{4-1}$$

方程中的 $\min_v$ 是对于 x 的所有邻居的。Bellman-Ford 方程是相当直观的。实际上，从 x 到 v 遍历之后，如果我们接下来取从 v 到 y 的最低费用路径，则该路径费用将是 $c(x, v) + d_v(y)$。因此我们必须通过遍历某些邻居 v 开始，从 x 到 y 的最低费用是对所有邻居 v 的 $c(x, v) + d_v(y)$ 的最小值。

但是对于那些可能怀疑该方程正确性的人，我们核查在图 4-27 中的源结点 u 和目的结点 z。源结点 u 有 3 个邻居：结点 v、x 和 w。通过遍历该图中的各条路径，容易看出 $d_v(z) = 5$、$d_x(z) = 3$ 和 $d_v(z) = 3$。将这些值连同费用 $c(u, v) = 2$、$c(u, x) = 1$ 和 $c(u, w) = 5$ 代入方程（4-1），得出 $d_u(z) = \min\{2 + 5, 5 + 3, 1 + 3\} = 4$，这显然是正确的，并且对同一个网络来说，这正是 Dijskstra 算法为我们提供的结果。这种快速验证应当有助于消除你可能具有的任何怀疑。

Bellman-Ford 方程不止是一种智力上的珍品，它实际上具有重大的实践重要性。特别是对 Bellman-Ford 方程的解为结点 x 的转发表提供了表项。为了理解这一点，令 v^* 是取得方程（4-1）中最小值的任何相邻结点。接下来，如果结点 x 要沿着最低费用路径向结点 y 发送一个分组，它应当首先向结点 v^* 转发该分组。因此，结点 x 的转发表将指定结点 v^* 作为最终目的地 y 的下一跳路由器。Bellman-Ford 方程的另一个重要实际贡献是它提出了将在 DV 算法中发生的邻居到邻居通信的形式。

其基本思想如下。每个结点 x 以 $D_x(y)$ 开始，对在 N 中的所有结点，估计从它自己到结点 y 的最低费用路径的费用。令 $\boldsymbol{D}_x = [D_x(y): y \in N]$ 是结点 x 的距离向量，该向量是从 x 到在 N 中的所有其他结点 y 的费用估计的向量。使用 DV 算法，每个结点 x 维护下列路由选择信息：

- 对于每个邻居 v，从 x 到直接相连邻居 v 的费用为 $c(x, v)$。
- 结点 x 的距离向量，即 $\boldsymbol{D}_x = [D_x(y): y \in N]$，包含了 x 到 N 中所有目的地 y 的费用的估计值。
- 它的每个邻居的距离向量，即对 x 的每个邻居 v，有 $\boldsymbol{D}_v = [D_v(y): y \in N]$。

在该分布式、异步算法中，每个结点不时地向它的每个邻居发送它的距离向量副本。当结点 x 从它的任何一个邻居 v 接收到一个新距离向量，它保存 v 的距离向量，然后使用 Bellman-Ford 方程更新它自己的距离向量如下：

$$D_x(y) = \min_v\{c(x,v) + D_v(y)\} \quad \text{对 } N \text{ 中的每个结点}$$

如果结点 x 的距离向量因这个更新步骤而改变，结点 x 接下来将向它的每个邻居发送其更新后的距离向量。令人惊奇的是，只要所有的结点继续以异步方式交换它们的距离向