Prim（普里姆）算法与Dijkstra（迪杰斯特拉）算法分析比较

孟庆伟 / 大连海洋大学职业技术学院

摘　要：数据结构中，普里姆算法与迪杰斯特拉算法分析考虑的均是带权图的造价最小问题，而这两种方法在生活的诸多领域应用相当广泛，但是学生往往把这两种算法混为一谈，似乎认为这两种算法求得的结果是一样的，而不明白为什么一个称为最小生成树，另一个则是最短路径。本文从算法的思想入手，通过示意图进行对比分析，突出不同点，使学生在今后的学习中加深对知识点的理解与认识。

关键词：普里姆；迪杰斯特拉；示意图

1 Prim算法与Dijstra算法思想分析比较

Prim算法是在由n 个顶点组成的无向连通图中，取图中任意一个顶点V作为生成树的根，之后往生成树上添加新的顶点W。在添加的顶点W和已经在生成树上的顶点V之间必定存在一条边，并且该边的权值在所有连通顶点V和W之间的边中取值最小。之后继续往生成树上添加顶点，直至生成树上含有n个顶点为止。一般情况下所添加的顶点应满足下列条件：在生成树的构造过程中，图中n个顶点分属两个集合：已落在生成树上的顶点集U和尚未落在生成树上的顶点集V-U，则应在所有连通U中顶点和V-U中顶点的边中选取权值最小的边。直到U中包含n个顶点，而V-U中为空集。

迪杰斯特拉（Dijkstra）算法是在由n个顶点组成的有向图中，从指定一个顶点出发，提出按路径长度的递增次序，逐步产生最短路径的算法，是单源点多目标点最短路径。最短路径的求解过程中，图中的顶点也是分属两个集合：指定出发点的顶点为一个集合V，其余顶点为一个集合W。从W中选择一个离V中顶点最近的一个顶点加入到V中，求出了长度最短的一条最短路径，并把刚选择加入最短路径的顶点作为中间点，对其它顶点到源点的路径长度进行修改，求出长度次短的一条最短路径，依次类推，直到从顶点v到其它各顶点的最短路径全部求出为止。

2 Prim算法与Dijstra算法处理步骤比较

Prim算法构造最小生成树的步骤是每次都从生成树外顶点与生成树内顶点相连的边中选择代价最小的加入到生成树，也就是每次按权值局部最小的方法，直到加入所有顶点和n-1条边为止。对应的步骤如下：

G=（V，E）是连通网，TE是G上最小生成树中边的集合。

初态：U={u_0|u_0∈V}，TE={}开始，重复执行下述操作：

（1）在所有u∈U，v∈V-U的边（u，v）∈E中找一条带权最小的边（u_0，v_0）并入集合TE，v_0并入U。

（2）重复（1）直至U=V为止。此时TE中必有n-1条边，则T=（V，TE）为N的最小生成树。

迪杰斯特拉提出了一个按路径长度递增的次序产生最短路径的算法。首先，引进一个辅助向量D，它的每个分量D[i]表示当前找到的从始点v到每个终点vi的最短路径的长度。它的初态为：若从v到v_i有弧，则D[i]为弧上的权值；否则置D[i]为∞，显然，长度为：

D[j]=Min{D[i]|v_i∈V}

的路径就是从v出发的长度最短的一条最短路径。此路径为（v，v_j）。那么，下一条长度次短的最短路径的终点是vk，则可想而知，这条路径或者是（v，v_k），或者是（v，v_j，v_k）。它的长度或者是从v到vk的弧上的权值，或者是D[j]和从v_j到vk的弧上的权值之和。具体步骤如下：

（1）S和T=V-S，S中存放已找到最短路径的顶点，T存放当前还未找到最短路径的顶点。初态，S中只包含源点v_0。（2）不断从T中选取到顶点v0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的长度，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。

3 示意图比较

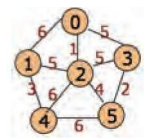

图1

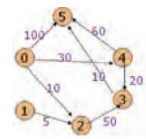

图2

由上图图1所示，用普里姆算法构造最小生成树，其过程如下表3.1.1～3.1.6：

表1 初始化时指定从0结点出发

表3.1.1

序号	0	1	2	3	4	5
顶点	0	0	0	0	0	0
权值	0	6	1	5	∞	∞
U	0					
TE	Φ					

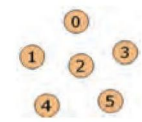

图3.1.1

表3.1.2选择离0结点最近的顶点2加入到生成树中

表3.1.2

序号	0	1	2	3	4	5
顶点	0	2	0	0	2	2
权值	0	5	0	5	6	4
U	0，2					
TE	（0，2）1					

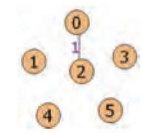

图3.1.2

中图分类号：TP301.6

表3.1.3选择离U最近的顶点5加入

表3.1.3

序号	0	1	2	3	4	5
顶点	0	2	0	5	2	2
权值	0	5	0	2	6	0
U	0，2，5					
TE	（0，2）1；（2，5）4					

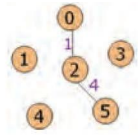

图3.1.3

表3.1.4选择离U最近的顶点3加入

表3.1.4

序号	0	1	2	3	4	5
顶点	0	2	0	5	2	2
权值	0	5	0	0	6	0
U	0，2，5，3					
TE	（0，2）1；（2，5）4；（5，3）2					

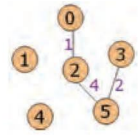

图3.1.4

表3.1.5选择离U最近的顶点1加入

表3.1.5

序号	0	1	2	3	4	5
顶点	0	2	0	5	1	2
权值	0	0	0	0	3	0
U	0，2，5，3，1					
TE	（0，2）1；（2，5）4；（5，3）2；（2，1）5					

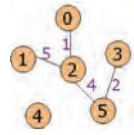

图3.1.5

表3.1.6选择离U最近的顶点4加入

表3.1.6

序号	0	1	2	3	4	5
顶点	0	2	0	5	1	2
权值	0	0	0	0	0	0
U	0，2，5，3，1，4					
TE	（0，2）1；（2，5）4；（5，3）2；（2，1）5；（1，4）3					

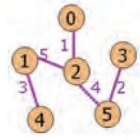

图3.1.6

以上分析的是最小生成树的建立过程，下面以图二为例分析最短路径的构造过程，过程如表3.2.1～3.2.5

表3.2.1 初始化时（选择0为源点）

顶点序号	0	1	2	3	4	5
当前最短路径终点S	0	0	0	0	0	0
当前最短路径长度D	0	∞	10	∞	30	100
当前最短路径P	0		{0，2}		{0，4}	{0，5}

表3.2.2 选择离源点0最近的终2加入

顶点序号	0	1	2	3	4	5
当前最短路径终点S	0	0	0	0	0	0
当前最短路径长度D	0	∞	10	60	30	100
当前最短路径P	0		{0，2}	{0，2，3}	{0，4}	{0，5}

表3.2.3 选择离源点0最近的终点4加入

顶点序号	0	1	2	3	4	5
当前最短路径终点S	0	0	0	0	0	0
当前最短路径长度D	0	∞	10	50	30	60
当前最短路径P	0		{0，2}	{0，4，3}	{0，4}	{0，4，5}

表3.2.4 选择离源点0最近的终点3加入

顶点序号	0	1	2	3	4	5
当前最短路径终点S	0	0	0	0	0	0
当前最短路径长度D	0	∞	10	50	30	90
当前最短路径P	0		{0，2}	{0，4，3}	{0，4}	{0，4，3，5}

表3.2.5 选择离源点0最近的终点5加入

顶点序号	0	1	2	3	4	5
当前最短路径终点S	0	0	0	0	0	0
当前最短路径长度D	0	∞	10	50	30	90
当前最短路径P	0	无	{0，2}	{0，4，3}	{0，4}	{0，4，3，5}

4 Prim（普里姆）算法和Dijkstra（迪杰斯特拉）的区别

通过以上的分析我把这两种算法的主要区别规纳总结如下：

（1）分析对象不同，前者为无向图，后者为有向图。

（2）连通的顶点不同，前者连通所有顶点，且总的代价最低，后者为单源点多目标点最短路径，所求得的是两顶点之间代价最小。

（3）普里姆算法生成最小生成树需重复n-1次，迪杰斯特拉算法则需重复n-2次。

5 总结

知识并不是孤立的，即存在着联系又有区别，只有把相近的内容加以对比分析，找出不同点，才能真正的理解和掌握所学的内容，从而提高我们的学习效率。

参考文献：

[1]杨剑.数据结构[M].北京：清华大学出版社,2011.

[2]严蔚敏.数据结构（C语言版）[M].北京:清华大学出版社,2006.

作者简介：孟庆伟（1979.8.9-），女，满族，辽宁铁岭人，教师，讲师，硕士，研究方向：计算机技术。

作者单位：大连海洋大学职业技术学院，辽宁大连 116300