d-子树划分问题

蔡延光[1)] 章 云[1)] 钱积新[2)]

[1)](广东工业大学自动化学院 广州 510006)

[2)](浙江大学工业控制技术国家重点实验室 杭州 310027)

摘 要 边赋非负权无向简单图的一簇顶点两两不相交的子树称为它的一个 d-子树划分(d 为非负实数),如果这些子树的顶点集的并等于此图的顶点集,且每棵子树的直径不超过 d. 图的 d-子树划分问题就是求它的一个含子树数最少的 d-子树划分及其所含的子树数. d-子树划分问题在有线通信网络、道路交通网络、城市供水网络、电力传输网络等网络的运行管理、维护与测试中具有很强的应用背景. 文中证明了对任意正实数 d,边赋非负权二分平面图的 d-子树划分问题是 NP-完全问题;提出了求解边赋非负权树 d-子树划分问题的一个线性时间算法,详细地讨论了算法的实现策略. 所提出的算法具有简明易实现、耗费时间少等特点.

关键词 d-子树划分;子树划分;NP-完全;树;算法;复杂度

中图法分类号 TP301 **DOI 号**: 10.3724/SP.J.1016.2010.00652

The d-Subtree Partition Problem

CAI Yan-Guang[1)] ZHANG Yun[1)] QIAN Ji-Xin[2)]

[1)](*Faculty of Automation, Guangdong University of Technology, Guangzhou* 510006)

[2)](*National Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou* 310027)

Abstract A d-subtree partition of undirected simple graph with nonnegative edge-weights is a collection of pairwise vertex-disjoint subtrees such that the union of vertex sets of these subtrees is equal to the vertex set of the graph and each subtree has diameter of no more than d, where d is a nonnegative real. The d-subtree partition problem of a graph is to find a d-subtree partition that has the smallest cardinality among all d-subtree partitions of the graph and the cardinality. The d-subtree partition problem has found some applications in operation management, maintenance and test of network such as wire communication network, road traffic network, urban water supply network and power transmission network. In this paper, it is proved that the d-subtree partition problem is NP-complete for bipartite planar graphs with nonnegative edge-weights for any positive real d. A linear time algorithm is presented to solve the d-subtree partition problem for trees with nonnegative edge-weights. Realization strategies for the algorithm are discussed in detail. The algorithm presented can be realized easily and requires less time.

Keywords d-subtree partition; subtree partition; NP-complete; tree; algorithm; complexity

---

收稿日期:2008-06-12;最终修改稿收到日期:2009-06-30. 本课题得到国家自然科学基金(60374062)、广东省自然科学基金团队项目(8351009001000002)和广东省科技计划项目基金(2007B010200070,2008B010200005)资助.**蔡延光**,男,1963 年生,博士,教授,主要研究领域为网络控制与优化、组合优化、智能优化、智能决策支持系统. E-mail: caiyg99@163.com. **章 云**,男,1963 年生,博士,教授,博士生导师,主要研究领域为网络控制与优化、控制网络集成、智能控制与信息处理技术. **钱积新**,男,1939 年生,教授,博士生导师,主要研究领域为复杂系统建模、控制与优化.

1 引　言

定义 1.　设 $G=(V,E)$ 是一个边赋非负权的无向简单图，顶点数为 $n=|V(G)|$，$d\geq 0$ 为实数. 如果 G 中 m 棵顶点两两不相交的子树 $T_1,T_2,\cdots,T_m$ 满足 $\bigcup_{i=1}^{m}V(T_i)=V(G)$，$d(T_i)\leq d$（这里 $d(T_i)=\max_{u,v\in V(T_i)}d(u,v)$ 为 T_i 的直径，$i=1,2,\cdots,m$），则称 $\{T_1,T_2,\cdots,T_m\}$ 为 G 的一个 d-子树划分. G 的含子树数最少的 d-子树划分称为 G 的最小 d-子树划分；最小 d-子树划分所含的子树数称为 G 的 d-子树划分数，记作 $\pi_d(G)$. 所谓 G 的 d-子树划分问题就是求 $\pi_d(G)$ 和 G 的一个最小 d-子树划分.

d-子树划分问题在有线通信网络、道路交通网络、城市供水网络、电力传输网络等网络的运行管理、维护与测试中具有很强的应用背景. 例如，在对某种有线通信网络进行维护时，需要把该通信网络中的结点划分为尽量少的组，使得每组中的结点可用一个直径不超过预定值 d 的树型子网通信. 这就是一个 d-子树划分问题.

理论上看来，d-子树划分问题是路划分问题[1]（path partition problem）、k-路划分问题[2]（k-path partition problem）、ω-路划分问题[3]（ω-path partition problem）等的推广. 如果将子树限定为路径，d 为非负实数 ω，则 d-子树划分问题成为 ω-路划分问题；当 G 是（边）非赋权图，ω 是一个正整数 k 时，ω-路划分问题就是 k-路划分问题；当 $k\geq n-1$ 时（实际上对路径的长度无限制），k-路划分问题简称为路划分问题. d-子树划分问题可进一步推广为图聚类问题（graph clustering problem），包括直径有限的子图划分问题[4].

从 Hamilton 路的 NP-完全性容易看出路划分问题属于 NP-完全问题. 由于 k-路划分问题、ω-路划分问题是路划分问题的直接推广，因此它们也均属于 NP-完全问题. 到目前为止，仅对于树[5]（tree）、圆弧图[6]（circular arc graph）、区间图[7]（interval graph）、仙人掌[8]（cactus）、上比较图[9]（cocomparability graph）、块图[1,10-11]（block graph）、二分置换图[10]（bipartite permutation graph）、上图[12-13]（cograph）、距离遗传图[14]（distance-hereditary graph）、P_4-稀疏图[15]（P_4-sparse graph）以及块为完全图、圈、完全二分图的图[16]等获得了多项式时间算法求解路划分问题. 对于 k-路划分问题，仅对树[17]、二分置换图[2]和仙人掌[18]等获得了多项式时间算法. 对于 ω-路划分问题，仅对树和森林[3]获得了多项式时间算法. 由于 d-子树划分问题的问题结构更加复杂，因此求解起来亦非易事.

本文证明了对任何固定的 $d>0$，边赋非负权二分平面图的 d-子树划分问题是 NP-完全问题. 这表明即使对结构非常简单的边赋非负权图，d-子树划分问题也是非常困难的. 树是一类特殊的二分平面图，也是理论研究和实际应用中最常见的网络拓扑结构之一. 本文提出了求解边赋非负权树 d-子树划分问题的一个线性时间算法；为了进一步提高算法的效率，还详细地讨论了算法的实现策略. 所提出的算法具有简明易实现、耗费时间少等特点.

定义 2.　考虑边赋非负权的无向简单图 $G=(V,E)$.

（1）设 $(u,v)\in E(G)$，以 $w(u,v)$ 表示边 (u,v) 的权.

（2）设 $u,v\in V(G)$，以 $d(u,v)$ 表示 u,v 之间的距离；如果 P 是 u,v 之间的最短路且 $d(u,v)=d(G)$，则称 P 为 G 的直径路；以 $(u\sim v)$ 表示 u 到 v 的一条路径，分别以 $V(u\sim v)$、$E(u\sim v)$ 表示 $(u\sim v)$ 的顶点集和边集.

（3）设 $v\in V(G)$，如果 v 的顶点度 $d(v)=1$，则称 v 为 G 的端顶点.

（4）设 $H,K\subseteq G$，以 $H\ominus K$ 表示在 H 中删除 $V(K)$ 以及与 $V(K)$ 相关联的所有边而形成的子图.

（5）设 $v\in V(G)$，G_1 是 $G-v$ 的一个连通分支，称导出子图 $G[V(G_1)\cup\{v\}]$ 为 v（点）的一个分支；任取 $x\in V(G_1)$，以 $G(v,x)$ 表示 v 的包含 x 的分支. 显然，$x\neq v$，$|V(G(v,x))|\geq 2$.

（6）设 $G_1,G_2\subseteq G$，$V(G_1)\cap V(G_2)=\{v\}$，记 $G_1\textcircled{v}G_2=G_1\cup G_2$，称 $G_1\textcircled{v}G_2$ 为 G_1,G_2 在 v 的粘合子图. 显然，若 G_1,G_2 为树，则 $G_1\textcircled{v}G_2$ 也是树.

除非特别声明，本文所提及的图、树、路径分别指边赋非负权的无向简单图、树、路径，并分别简称为图、树、路径. 约定空树也是树、空树的直径为 0.

文中未定义的术语和记号与文献[19]一致. 为了表述方便，算法采用 C 语言风格设计.

2 d-子树划分问题的计算复杂性

定理 1.　对任意 $d>0$ 的实数，边赋非负权二分平面图的 d-子树划分问题是 NP-完全问题.

证明.　利用平面图的顶点覆盖问题的 NP-完

全性[20-21]. 设 $G=(V,E)$ 是一个平面图，$V=\{v_i \mid i=1,2,\cdots,n\}$. 构造新图 $G_1=(V_1,E_1)$ 如下：

$V_1=\{v_i,x_i,y_i \mid i=1,2,\cdots,n\} \cup$

$\{z_{ij} \mid (v_i,v_j)\in E(G), i<j;\ i,j=1,2,\cdots,n\}$；

$E_1=\{(v_i,x_i),(x_i,y_i) \mid i=1,2,\cdots,n\} \cup$

$\{(v_i,z_{ij}),(z_{ij},v_j) \mid z_{ij}\in V(G_1);\ i,j=1,2,\cdots,n\}$；

对任意 $e\in E_1$，定义 $w(e)=d/2$（见图 1）.

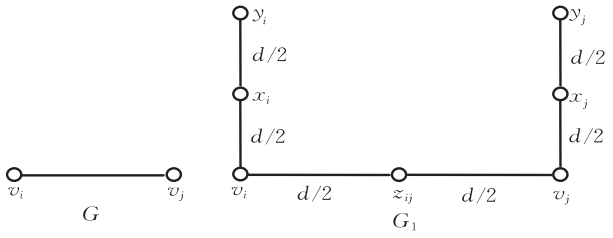

图 1 从图 G 到图 G_1 的转换

容易看出，因为 G 是平面图及 G_1 不含奇圈，故 G_1 是一个边赋非负权的二分平面图.

一方面，设 S 是 G 的任意最小顶点覆盖集，不妨设 $S=\{v_i \mid i=1,2,\cdots,m\}$（$m\leq n$）. 构造 G_1 的 d-子树划分 $\{T_1,T_2,\cdots,T_{m+n}\}$ 如下：

$T_i=\{v_i\} \cup \{z_{ij} \mid z_{ij}\in V(G_1), j=1,2,\cdots,n\}$ 的导出子图（实际上是以 v_i 为中心、直径不大于 d 的一颗星），$i=1,2,\cdots,m$；

$T_{m+i}=\{x_i,y_i\}$ 的导出子图（实际上是 K_2，直径为 $d/2$），$i=1,2,\cdots,m$；

$T_{m+i}=\{v_i,x_i,y_i\}$ 的导出子图（实际上是以 x_i 为中心、直径为 d 的一颗星），$i=m+1,m+2,\cdots,n$.

因此，$\pi_d(G_1)\leq\beta(G)+n$，其中 $\beta(G)$ 为 G 的顶点覆盖数.

另一方面，对 G_1 的任何最小 d-子树划分，因为其中每棵子树的直径不大于 d，故含有 y 型顶点（这里，y 型顶点是形为 y_i 的顶点，$i=1,2,\cdots,n$. 以下提及的 v 型顶点有类似的解释）的子树必不含 z 型顶点（z 型顶点是形为 z_{ij} 的顶点，$z_{ij}\in V(G_1)$；$i,j=1,2,\cdots,n$）；且对一切 $i\neq j$，$i,j=1,2,\cdots,n$，含有 y_i 的子树不含 y_j. 因此，G_1 的任何最小 d-子树划分中不含 z 型顶点的子树数至少为 n.

然而，我们有：存在 G_1 的一个最小 d-子树划分 Π^*，使得其中任何含有 z 型顶点的子树恰好含一个 v 型顶点. 事实上，设 $\Pi=\{T_1,T_2,\cdots,T_m\}$ 是 G_1 的任意最小 d-子树划分，从 Π 出发，分两步构造 Π^*.

第 1 步. 构造 G_1 的一个最小 d-子树划分 Π'，使得其中任何含有 z 型顶点的子树至少含有一个 v 型顶点.

在 Π 中，如果有某棵子树譬如 T_1 含有 z_{ij} 而不含 v 型顶点，此时必有 $V(T_1)=\{z_{ij}\}$. 考虑 G_1 中与 z_{ij} 相邻的顶点 v_i，不妨设 v_i 所在的子树为 T_2. 在 T_2 中，v_i 的顶点度 $d(v_i)$ 必等于 1（否则，如果 $d(v_i)>1$，则 T_2 是一个以 v_i 为中心的直径为 d 的一颗星；因此，$\{T_2\cup(z_{ij}\sim v_i),T_3,\cdots,T_m\}$ 是 G_1 的一个 d-子树划分，与 Π 的最小性相矛盾. 对于 $d(v_i)=0$ 可类似讨论）. 下面分两种情形讨论.

情形 1. 若 T_2 不含 z 型顶点，则 $\{(z_{ij}\sim v_i),T_2-v_i,T_3,\cdots,T_m\}$ 是 G_1 的一个最小 d-子树划分，但其中含有 z 型顶点而不含有 v 型顶点的子树少了一棵.

情形 2. 若 T_2 含有 z 型顶点，并不妨设为 z_{is}（$i<s$），此时 T_2 必含有 v_s（否则应有 $T_2=(z_{is}\sim v_i)$，于是 $\{(z_{ij}\sim v_i)\cup(z_{is}\sim v_i),T_3,\cdots,T_m\}$ 是 G_1 的一个 d-子树划分，与 Π 的最小性相矛盾），于是 $\{(z_{ij}\sim v_i),T_2-v_i,T_3,\cdots,T_m\}$ 是 G_1 的一个最小 d-子树划分，但其中含有 z 型顶点而不含 v 型顶点的子树少了一棵.

重复使用上述方法，即可得到 Π'.

第 2 步. 从 Π' 构造 Π^*. 设 $\Pi'=\{T_1',T_2',\cdots,T_m'\}$.

在 Π' 中，如果有某棵子树譬如 T_1' 含有 z_{ij} 且含两个 v 型顶点（因为 T_1' 的直径不大于 d，T_1' 不可能含有两个以上的 v 型顶点），此时 $T_1'=(v_i\sim z_{ij}\sim v_j)$，不妨设含 x_i 的子树为 T_2'（显然，$T_2'=(x_i\sim y_i)$），于是 $\{(z_{ij}\sim v_j),T_2'\cup(v_i\sim x_i),T_3',\cdots,T_m'\}$ 是 G_1 的一个最小 d-子树划分，但含有 z 型顶点且含两个 v 型顶点的子树少了一棵. 重复使用此方法，就可获得 Π^*.

在 Π^* 中，不妨设含 z 型顶点的子树为 $T_1^*,T_2^*,\cdots,T_q^*$，对应的 v 型顶点分别为 $v_1,v_2,\cdots,v_q$. 显然，$\{v_1,v_2,\cdots,v_q\}$ 是 G 的一个顶点覆盖集，即 $\beta(G)\leq q$. 但是 $\pi_d(G_1)=\Pi^*$ 中含 z 型顶点的子树数 $+\Pi^*$ 中不含 z 型顶点的子树数 $\geq q+n\geq\beta(G)+n$.

因此，$\pi_d(G_1)=\beta(G)+n$.

因为求 $\beta(G)$ 为 NP-完全问题，故求 $\pi_d(G_1)$ 也是 NP-完全问题. 证毕.

3 边赋非负权树的 d-子树划分问题

标号 D. 设 T 是一棵树，$n=|V(T)|\geq 1$，任选 $v_0\in V(T)$. T 的每个顶点 v 标号为 $l(v)$：$l(v)=d(v,v_0)$，称 $l(v)$ 为 v 的标号.

通过标号 D 标号的树 T 可以看作是一棵以 v_0 为根的（倒立的）有根树；这样，我们就可使用与有根

树有关的一些术语，如根（顶点）、双亲（顶点）、子女（顶点）、祖先（顶点）、后代（顶点）.

如无特别说明，本节所论及的树 T 均是指按标号 D 对所有顶点进行标号的以 v_0 为根的有根树.

注 1. 标号 D 可在线性时间内完成. 事实上，在用标号 D 对 T 的顶点进行标号时，还可建立一个包含根 v_0 和 T 的所有使 $d(v)\geq 3$ 的顶点 v 组成的有序集 Q（Q 实际是一个栈）. Q 的构造方法如下：

初始，$Q=\{v_0\}$，$l(v_0)=0$；

第 1 步. 依次对 v_0 的每个子女 v 标号：$l(v)=w(v_0,v)$，如果 $d(v)\geq 3$，则将 v 置于 Q 之首（即 v 进栈 Q）；

第 2 步. 对每个刚刚在第 1 步获得标号的顶点 u，依次对 u 的每个子女 v 标号：$l(v)=l(u)+w(u,v)$，如果 $d(v)\geq 3$，则 v 进栈 Q；

…

第 k 步. 对每个刚刚在第 $k-1$ 步获得标号的顶点 u，依次对 u 的每个子女 v 标号：$l(v)=l(u)+w(u,v)$，如果 $d(v)\geq 3$，则 v 进栈 Q；

…

继续这一过程，直至 T 的所有顶点均已标号. 因为 T 的每个顶点用标号 D 恰好标号一次，故标号 D 可在 $O(n)$ 时间内完成.

显然，如果 T 是路径，则 Q 只包含根 v_0；否则，Q 的首元素（即栈顶元素）是使 $|E(v_0\sim v)|$ 最大的 $d(v)\geq 3$ 的顶点 v，即 Q 的栈顶元素 v 是 T 的深度最大的使 $d(v)\geq 3$ 的顶点.

定义 3. 设 $v\in V(T)$，$T(v,x)$ 是 v 的一个分支，如果 $v_0\notin V(T(v,x)-v)$（即 $T(v,x)$ 的顶点或者为 v、或者是 v 的后代），则称 $T(v,x)$ 为 v 的后代分支.

注 2. 对于 $v\in V(T)$，v 的后代分支与 v 的子女形成一一对应的关系. 可以把 v 的子女看作是从左到右排列的，即把 v 的后代分支也看作是从左到右排列的.

显然，若 $T(v,x)$ 是 v 的一个后代分支，则存在 T 的端顶点 $y\in V(T(v,x))$，$y\neq v$，使 y 是 $T(v,x)$ 中标号最大的顶点.

把 T 的所有顶点分为 3 类：未检查、已检查和待检查，T 的每个顶点属于且仅属于其中某一类.

定义 4. 设 $v\in V(T)$，$T(v,x)$ 是 v 的一个分支，如果 $T(v,x)-v$ 的所有顶点均为已检查顶点，则称 $T(v,x)$ 为已检查（的）分支；否则称 $T(v,x)$ 为未检查（的）分支. 将 $T(v,x)$ 标为已检查是指将 $T(v,x)-v$ 的所有顶点标为已检查.

特别地，如果 $T(v,x)$ 是 v 的一个后代分支，则 $T(v,x)$ 为已检查分支当且仅当 v 的所有后代均为已检查顶点；将 $T(v,x)$ 标为已检查是指将 $T(v,x)$ 中的 v 的全部后代标为已检查.

定义 5. 设 $v\in V(T)$，$T(v,x_i)(i=1,2,\cdots,k,k\geq 0)$ 为 v 的全部（不同的）已检查的后代分支，其中 $x_i(i=1,2,\cdots,k)$ 为 T 的端顶点且是 $T(v,x_i)$ 中具有最大标号的顶点，$l(x_1)\geq l(x_2)\geq\cdots\geq l(x_k)$；记 T_v 为 v 的全部已检查的后代分支在 v 的粘合子树，即 $T_v=T(v,x_1)\textcircled{v}T(v,x_2)\textcircled{v}\cdots\textcircled{v}T(v,x_k)$. 定义 v 处的二元组 $(v^{(1)},v^{(2)})$ 如下：当 $k\geq 1$ 时定义 $v^{(1)}=x_1$，否则 $v^{(1)}=v$；当 $k\geq 2$ 时定义 $v^{(2)}=x_2$，否则 $v^{(2)}=v$.

注意，当 $k=0$ 时，$T_v=\varnothing$，$d(T_v)=0$. 特别地，因为端顶点（除 v_0 外）没有后代分支，故对于任意端顶点 $v(v\neq v_0)$：$v^{(1)}=v$，$v^{(2)}=v$，$T_v=\varnothing$.

3.1 算　法

对于树 T，算法 1 通过自底向上检查 $d(v)\geq 3$ 的顶点 v（即自顶向底检查栈 Q 中的元素）、从左到右考虑 v 的最左边的未检查的后代分支 $T(v,x)$ 等方法求解 $\pi_d(T)$ 及 T 的一个最小 d-子树划分 $\Pi_d^*(T)$. 主要步骤如下：

1. 初始时，将 T 的所有端顶点 $v(v\neq v_0)$ 标为待检查，其余顶点标为未检查. 因此，初始时，对于一切 $v\in V(T)$，有 $v^{(1)}=v$，$v^{(2)}=v$，$d(T_v)=0$. 但算法 1 只用到了 $v\in Q$ 和 v 为端顶点的 $(v^{(1)},v^{(2)})$ 及 $d(T_v)$；因此，算法 1 在执行过程中仅更新这些顶点的 $(v^{(1)},v^{(2)})$ 及 $d(T_v)$.

2. 取 Q 的栈顶元素 v，考虑 v 的最左边的未检查的后代分支 $T(v,x)$，转步 3. 当不存在未检查的后代分支 $T(v,x)$ 时，如果 $v=v_0$ 则 $d(T)\leq d$，T 作为 $\Pi_d^*(T)$ 的一个元素，结束；否则将 v 标为待检查，并从 Q 中删除 v，转步 2.

3. 如果 $d(T(v,x))>d$，则从 $T(v,x)$ 中找出 $\Pi_d^*(T)$ 的一个元素，并进行相关的更新，转步 2；否则转步 4.

4. 如果 $d(T(v,x)\textcircled{v}T_v)>d$，则 $T(v,x)-v$、$T(v,v^{(1)})-v$ 中必有某一个可作为 $\Pi_d^*(T)$ 的一个元素，辨出之，并进行相关的更新，转步 2；否则转步 5.

5. 将 $T(v,x)$ 标为已检查，并进行相关的更新，转步 2.

以上步 3、4 需要计算相关子树的直径，常规方法计算树的直径需要线性时间（例如见文献[22]）. 这里提供了一种方法，用常数时间计算相关子树的直径（见算法 1 及 3.2 节、3.3 节的讨论），从而计算所有的相关子树的直径才耗费线性时间.

具体算法如下.

算法 1. 求解边赋非负权树的 d-子树划分问题.

输入：树 T（$n\geq 1$），常数 $d\geq 0$.

输出：$\pi_d(T)$ 及 T 的一个最小 d-子树划分 $\Pi_d^*(T)$.

初始化：$\pi_d(T)=0$；$\Pi_d^*(T)=\varnothing$；
将 T 按标号 D 标号并构造 Q； //片段 1
将 T 的所有端顶点 $v(v\neq v_0)$ 标为待检查，其余顶点标为未检查； //片段 2
对于一切 $v\in Q$ 或 v 为端顶点，$v^{(1)}=v$，$v^{(2)}=v$，$d(T_v)=0$； //片段 3
while $V(T)\neq\varnothing$
{
取 Q 的栈顶元素 v； //对应引理 1
如果不存在 v 的未检查的后代分支 //片段 4
{
如果 $v=v_0$
{
$\Pi_d^*(T)=\Pi_d^*(T)\bigcup\{T\}$；$\pi_d(T)=\pi_d(T)+1$；
$T=\varnothing$；
//对应引理 5，片段 5
}
否则
{
将 v 标为待检查；$Q=Q-\{v\}$；continue； //片段 6
}
}
取 v 的最左边的未检查的后代分支 $T(v,x)$； //片段 7
取 $T(v,x)$ 中的待检查顶点 $u(u\neq v)$；
//对应引理 2，片段 8
如果 $l(u^{(1)})-l(v)>d$ //对应引理 6
{
取 $z,z_1\in V(u\sim v)$，使 $(z,z_1)\in E(u\sim v)$，$l(u^{(1)})-l(z)\leq d$ 且 $l(u^{(1)})-l(z_1)>d$； //片段 9
$\Pi_d^*(T)=\Pi_d^*(T)\bigcup\{T(z_1,u^{(1)})-z_1\}$；$\pi_d(T)=\pi_d(T)+1$；$T=T\ominus(T(z_1,u^{(1)})-z_1)$； //片段 10
如果 z_1 为端顶点且 $z_1\neq v_0$
{
将 z_1 标为待检查；$z_1^{(1)}=z_1$；$z_1^{(2)}=z_1$；$d(T_{z_1})=0$；
//片段 11
}
continue；
}
//以下处理 $l(u^{(1)})-l(v)\leq d$ 的情形
如果 $l(u^{(1)})+l(v^{(1)})-2l(v)>d$ //对应引理 4、引理 7
{
如果 $l(u^{(1)})\geq l(v^{(1)})$ //对应引理 7(1)
{
$\Pi_d^*(T)=\Pi_d^*(T)\bigcup\{T(v,x)-v\}$；$\pi_d(T)=\pi_d(T)+1$；$T=T\ominus(T(v,x)-v)$； //片段 12
}
否则 //对应引理 4、引理 7(2)
{
将 $T(v,x)$ 标为已检查； //片段 13
$\Pi_d^*(T)=\Pi_d^*(T)\bigcup\{T(v,v^{(1)})-v\}$；$\pi_d(T)=\pi_d(T)+1$；$T=T\ominus(T(v,v^{(1)})-v)$； //片段 14
$d(T_v)=\max\{d(T_u),l(u^{(1)})+l(v^{(2)})-2l(v)\}$；
$v^{(1)}=u^{(1)}$； //片段 15
}
continue；
}
//以下处理 $l(u^{(1)})-l(v)\leq d$ 且 $l(u^{(1)})+l(v^{(1)})-2l(v)\leq d$ 的情形，对应引理 3
将 $T(v,x)$ 标为已检查； //片段 16
如果 $l(u^{(1)})\geq l(v^{(1)})$ //对应引理 3(1)
{
$d(T_v)=\max\{d(T_u),l(u^{(1)})+l(v^{(1)})-2l(v)\}$；
$v^{(2)}=v^{(1)}$；$v^{(1)}=u^{(1)}$； //片段 17
}
否则
{
如果 $l(u^{(1)})\geq l(v^{(2)})$ //对应引理 3(2)
{
$d(T_v)=\max\{d(T_v),l(u^{(1)})+l(v^{(1)})-2l(v)\}$；
$v^{(2)}=u^{(1)}$； //片段 18
}
}
} //循环结束
输出 $\pi_d(T)$、$\Pi_d^*(T)$；
结束.

3.2 算法的正确性

除非特别声明，本小节及 3.3 节所讨论的 T、Q 分别指算法 1 执行过程中任何阶段的 T、Q；T_0、Q_0 分别为算法 1 初始时的 T、Q.

性质 1. 设 T_1 为 T 的子树，$v\in V(T)$，$T(v,x)$ 为 v 的任意分支，则 $T_1\ominus(T(v,x)-v)$ 是树. 特别地，如果 T 是以 v_0 为根的有根树，$T(v,x)$ 为 v 的任意后代分支，则 $T\ominus(T(v,x)-v)$ 是以 v_0 为根的有根树.

证明. 记 $T'=T_1\ominus(T(v,x)-v)$. 因为 T' 不含有圈，故只需证明 T' 连通. 反证，设存在 $y,z\in V(T')$，y,z 在 T' 中不存在路径相连. 因为 y,z 在 T_1 中有路径相连，故存在路径 $P=(y\sim z)$，$P\subseteq T_1$ 但 $P\not\subseteq T'$. 因此，P 含有 $T(v,x)-v$ 中的顶点 u. 又因 $v\notin V(T(v,x)-v)$，故 $u\neq v$. 因为 T_1 是树，且 $y,z\notin V(T(v,x)-v)$，故从 y 到 u 先必经过 v 才能达到 u，从 u 到 z 也必须最后经过 v 才能离开 $T(v,x)$. 这样，$P\subseteq(y\sim v)\bigcup(v\sim u)\bigcup(u\sim v)\bigcup(v\sim z)$. 但 $(v\sim u)\bigcup(u\sim v)$ 含有圈，与 P 是路径相矛盾. 因此，

T'是树.

特别地，取 $T_1=T$，$T'=T_1\ominus(T(v,x)-v)=T\ominus(T(v,x)-v)$是树. 而当 T 是以 v_0 为根的有根树、$T(v,x)$为 v 的后代分支时，因为 $T(v,x)-v$ 的所有顶点都是 v 的后代，故 $v_0\notin V(T(v,x)-v)$，即 $v_0\in V(T\ominus(T(v,x)-v))$，从而 $T\ominus(T(v,x)-v$ 是以 v_0 为根的有根树. 证毕.

由此，容易得到性质 2～性质 4.

性质 2. 设$v\in V(T)$，u 为 v 的后代，则 $l(u)=l(v)+d(u,v)$.

性质 3. 设$v\in V(T)$，$T(v,x_i)(i=1,2)$为 v 的任意两个不同的分支，则对任意 $y_i\in V(T(v,x_i))(i=1,2)$，$d(y_1,y_2)=d(y_1,v)+d(y_2,v)$. 特别地，如果 $T(v,x_i)(i=1,2)$为 v 的任意两个不同的后代分支，则对任意 $y_i\in V(T(v,x_i))(i=1,2)$，$d(y_1,y_2)=l(y_1)+l(y_2)-2l(v)$.

性质 4. 树的直径不小于其任何子树的直径.

性质 5[22]. 设$|V(T)|\geq 2$，$u,v\in V(T)$，v 是 T 的端顶点且使 $d(u,v)=\max\limits_{x\in V(T)}d(u,x)$(即 v 是离 u 最远的顶点)，则存在 T 的直径路 P 使 $v\in V(P)$.

引理 1. 若 $V(T)\neq\varnothing$，则 $v_0\in V(T)$，$v_0\in Q$.

证明. 算法 1 对 T 的更新是在片段 5 通过置 $T=\varnothing$(更新后的 T 不满足引理的条件)、在片段 10、片段 12、片段 14 通过$\ominus$运算进行的. 由性质 1，通过$\ominus$运算更新后的 T 仍有 $v_0\in V(T)$.

Q 是在片段 1(指算法 1 的片段 1，下类同. 不再一一说明)建立的，初始的 Q 满足 $v_0\in Q$. 对 Q 的更新仅在片段 6 进行. 显然，直至算法 1 结束仍然有 $v_0\in Q$. 证毕.

引理 2. 设 v 为 Q 的栈顶元素，$T(v,x)$为 v 的未检查的后代分支，则

(1) 在 $T(v,x)$中，存在唯一的待检查顶点 u，且 $u\neq v$，$u\neq v_0$，u 不是孤立点；

(2) 对一切 $y\in V(u\sim v)(y\neq u,y\neq v)$，都有 y 为未检查顶点且 $d(y)=2$；

(3) u 的全部后代分支均是已检查分支.

证明.

(1) 首先证明 u 的存在性.

设 u 是 $T(v,x)$中使$|E(u\sim v)|$最大的非已检查顶点(即 u 是 $T(v,x)$中深度最大的非已检查顶点. u 可能是待检查顶点，也可能是未检查顶点). 由$|E(u\sim v)|$的最大性知，u 的所有后代分支都为已检查分支. 因此，$u\neq v$. 而 u 为 v 的后代表明 $u\neq v_0$. 由引理 1 知 $v_0\in V(T)$，故 u 不是孤立点.

我们有：u 必定是待检查顶点. 反证，设 u 是未检查顶点.

首先，$u\notin Q_0$. 否则，$u\in Q_0$. 因为 v 是 Q 的栈顶元素、u 为 v 的后代，故 $u\notin Q$. 因此，按算法 1，u 或者标为待检查(片段 6)，或者标为已检查(片段 13、片段 16)，与 u 是未检查顶点的假设相矛盾.

其次，对 $u\notin Q_0$，u 不会是端顶点，因为算法 1 将端顶点(除 v_0 外)或者标为待检查(片段 2、片段 11)，或者标为已检查(片段 13、片段 16).

因此，u 不是端顶点，且 $u\notin Q_0$. u 不是端顶点也不是孤立点表明 u 至少有一个后代 u'(注意 u'是已检查顶点). 按算法 1，u'成为已检查顶点必定是在考虑离 u 最近的祖先 $w\in Q_0$ 时，将 w 的 u 所在的后代分支标为已检查所致. 作为 w 的后代 u 也将同时标为已检查(片段 13、片段 16)，与 u 是未检查顶点的假设相矛盾.

因此，u 必定是待检查顶点.

下面证明 u 的唯一性.

设 $u_1\neq v$，$u_2\neq v$，且 u_1，u_2 为 $T(v,x)$上的两个待检查顶点.

情形 1. u_1 位于 u_2 的某个后代分支上. 按算法 1，当 u_2 标为待检查时，必定有它的所有后代均为已检查顶点，故 u_1 为已检查顶点. 矛盾. 类似讨论 u_2 位于 u_1 的某个后代分支上的情形.

情形 2. u_1 不在 u_2 的任何后代分支上，且 u_2 不在 u_1 的任何后代分支上. 此时路径$(u_1\sim v)$、$(u_2\sim v)$必在 $T(v,x)-v$ 上相交，设 y 是标号最大的交点. 因此，$d(y)\geq 3$，即 $y\in Q_0$. 因为 v 是 Q 的栈顶元素、y 为 v 的后代，故 $y\notin Q$. 因此，按算法 1，y 要么是已检查顶点，要么是待检查顶点. 不论是何种情况出现，作为 y 的后代，u_1，u_2 均是已检查顶点. 矛盾.

综上所述，在 $T(v,x)$中，存在唯一的待检查顶点 u，且 $u\neq v$.

(2) 先证明对一切 $y\in V(u\sim v)(y\neq u,y\neq v)$，都有 y 为未检查顶点. 反证，设存在$y\in V(u\sim v)(y\neq u,y\neq v)$，$y$ 是待检查顶点或是已检查顶点.

情形 1. y 是待检查顶点. 按算法 1，y 要么是端顶点(但 y 有后代 u，矛盾)，要么 $y\in Q_0$(此时，类似上面的分析可得，y 的所有后代均为已检查顶点，但其后代 u 却是待检查顶点，矛盾). 因此，y 不是待检查顶点.

情形 2. y 是已检查顶点. 按算法 1，y 成为已检查顶点必定是在考虑离 y 最近的祖先 $w\in Q_0$ 时，将 w 的 y 所在的后代分支标为已检查所致. 因为 y 的

后代 u 也是 w 的后代，u 将同时标为已检查. 矛盾.

因此，y 为未检查顶点.

再证明 $d(y)=2$. 反证，设存在 $y\in V(u\sim v)$，$y\neq u, y\neq v$，使 $d(y)\neq 2$. 因为 y 有后代 u，故 y 不是端顶点、也不是孤立点. 从而 $d(y)\geq 3$，即 $y\in Q_0$. 类似上面的分析可得，y 要么是已检查顶点，要么是待检查顶点. 不论是何种情况出现，y 的后代 u 必为已检查顶点. 矛盾.

(3)上述证明过程(1)已经证明. 证毕.

注 3. v、$T(v,x)$ 如引理 2 所设. 算法 1 只对除 v_0 外的端顶点、Q_0 中的元素标为待检查，且 v_0 永远是未检查顶点. 因此，引理 2 中的 u 要么是端顶点，要么是 Q_0 中的元素；$u^{(1)}$ 为 T 和 $T(v,x)$ 的端顶点且是 $T(v,x)$ 中具有最大标号的顶点. 此外，根据引理 2(2)，可通过以下方法在 v 的未检查的后代分支 $T(v,x)$ 中寻找唯一的待检查顶点 u：从 v 出发，如果 v 的唯一子女 x_1 为待检查顶点，则 $u=x_1$；否则考虑 x_1 的唯一子女 x_2. 如果 x_2 为待检查顶点，则 $u=x_2$；否则考虑 x_2 的唯一子女 x_3，…，最后必能找到待检查顶点 u.

根据引理 2(2)和性质 5，易得下面性质.

性质 6. 设 v 为 Q 的栈顶元素，$T(v,x)$ 为 v 的未检查的后代分支，u 为 $T(v,x)$ 中的待检查顶点，则 $d(T(v,x))=\max\{d(T_u), l(u^{(1)})-l(v)\}$.

引理 3. 设 v 为 Q 的栈顶元素，$T(v,x)$ 为 v 的未检查的后代分支，u 为 $T(v,x)$ 中的待检查顶点.

(1)如果 $l(u^{(1)})\geq l(v^{(1)})$，则 $d(T(v,x)\textcircled{v}T_v)=\max\{d(T(v,x)), l(u^{(1)})+l(v^{(1)})-2l(v)\}=\max\{d(T_u), l(u^{(1)})+l(v^{(1)})-2l(v)\}$.

(2)如果 $l(u^{(1)})<l(v^{(1)})$ 且 $l(u^{(1)})\geq l(v^{(2)})$，则 $d(T(v,x)\textcircled{v}T_v)=\max\{d(T_v), l(u^{(1)})+l(v^{(1)})-2l(v)\}$.

(3)如果 $l(u^{(1)})<l(v^{(2)})$，则 $d(T(v,x)\textcircled{v}T_v)=d(T_v)$.

证明. 根据性质 4 有

$$d(T(v,x)\textcircled{v}T_v)\geq d(T(v,x)) \quad (1)$$

$$d(T(v,x)\textcircled{v}T_v)\geq d(T_v) \quad (2)$$

情形 1. $V(T_v)\neq\varnothing$. 此时 $u^{(1)}, v^{(1)}$ 在 v 的不同后代分支上. 由性质 3 有

$$d(u^{(1)}, v^{(1)})=l(u^{(1)})+l(v^{(1)})-2l(v) \quad (3)$$

由 $d(T(v,x)\textcircled{v}T_v)\geq d(u^{(1)}, v^{(1)})$ 知

$$d(T(v,x)\textcircled{v}T_v)\geq l(u^{(1)})+l(v^{(1)})-2l(v) \quad (4)$$

情形 2. $V(T_v)=\varnothing$. 此时 $v^{(1)}=v$. 由性质 2，$d(u^{(1)}, v)=l(u^{(1)})-l(v)$. 故式(3)成立，从而式(4)仍然成立.

(1)因为 $l(u^{(1)})\geq l(v^{(1)})$，故 $u^{(1)}$ 为 T 的端顶点且是 $T(v,x)\textcircled{v}T_v$ 中具有最大标号的顶点，即 $u^{(1)}$ 是 $T(v,x)\textcircled{v}T_v$ 中离 v 最远的端顶点. 由性质 5 知，存在 $T(v,x)\textcircled{v}T_v$ 的直路径 $(u^{(1)}\sim y)$，$d(T(v,x)\textcircled{v}T_v)=d(u^{(1)}, y)$，其中 $y\in V(T(v,x)\textcircled{v}T_v)$.

情形 1. $y\in V(T(v,x))$. 因为 $u^{(1)}\in V(T(v,x))$，故 $d(T(v,x))\geq d(u^{(1)}, y)=d(T(v,x)\textcircled{v}T_v)$. 结合式(1)有 $d(T(v,x)\textcircled{v}T_v)=d(T(v,x))$；再结合式(4)有 $d(T(v,x))\geq l(u^{(1)})+l(v^{(1)})-2l(v)$. 从而 $d(T(v,x)\textcircled{v}T_v)=\max\{d(T(v,x)), l(u^{(1)})+l(v^{(1)})-2l(v)\}$.

情形 2. $y\in V(T_v)$. 因为 $y=v$ 可归为情形 1，不妨设 $y\neq v$. 从而 y 在 v 的某个已检查的后代分支中. 由性质 3 有 $d(u^{(1)}, y)=l(u^{(1)})+l(y)-2l(v)$. 因为 $l(v^{(1)})\geq l(y)$，故 $l(u^{(1)})+l(v^{(1)})-2l(v)\geq l(u^{(1)})+l(y)-2l(v)=d(u^{(1)}, y)=d(T(v,x)\textcircled{v}T_v)$. 结合式(4)有 $d(T(v,x)\textcircled{v}T_v)=l(u^{(1)})+l(v^{(1)})-2l(v)$；再结合式(1)有 $l(u^{(1)})+l(v^{(1)})-2l(v)\geq d(T(v,x))$. 从而 $d(T(v,x)\textcircled{v}T_v)=\max\{d(T(v,x)), l(u^{(1)})+l(v^{(1)})-2l(v)\}$.

由性质 6，$d(T(v,x))=\max\{d(T_u), l(u^{(1)})-l(v)\}$. 又因 $v^{(1)}$ 是 v 的后代，故 $l(v^{(1)})\geq l(v)$，即得 $l(u^{(1)})-l(v)\leq l(u^{(1)})+l(v^{(1)})-2l(v)$. 从而 $d(T(v,x)\textcircled{v}T_v)=\max\{d(T(v,x)), l(u^{(1)})+l(v^{(1)})-2l(v)\}=\max\{d(T_u), l(u^{(1)})-l(v), l(u^{(1)})+l(v^{(1)})-2l(v)\}=\max\{d(T_u), l(u^{(1)})+l(v^{(1)})-2l(v)\}$.

结论(1)得证.

(2)因为 $l(u^{(1)})\geq l(v)$，再由条件 $l(u^{(1)})<l(v^{(1)})$ 有 $l(v^{(1)})>l(v)$，故 $v^{(1)}\neq v$，即 T_v 是非空的树，$v^{(1)}$ 是 $T(v,x)\textcircled{v}T_v$ 中离 v 最远的端顶点. 由性质 5 知，存在 $T(v,x)\textcircled{v}T_v$ 的直路径 $(v^{(1)}\sim y)$，$d(T(v,x)\textcircled{v}T_v)=d(v^{(1)}, y)$，其中 $y\in V(T(v,x)\textcircled{v}T_v)$.

情形 1. $y\in V(T(v,x))$. 由性质 3，$d(v^{(1)}, y)=l(v^{(1)})+l(y)-2l(v)$. 因为 $l(y)\leq l(u^{(1)})$，故 $d(T(v,x)\textcircled{v}T_v)=d(v^{(1)}, y)\leq l(v^{(1)})+l(u^{(1)})-2l(v)$. 结合式(4)有 $d(T(v,x)\textcircled{v}T_v)=l(v^{(1)})+l(u^{(1)})-2l(v)$；再结合式(2)有 $d(T(v,x)\textcircled{v}T_v)=\max\{d(T_v), l(u^{(1)})+l(v^{(1)})-2l(v)\}$.

情形 2. $y\in V(T_v)$. 因$(v^{(1)}\sim y)$是 T_v中的路径，$d(T(v,x)\textcircled{v}T_v)=d(v^{(1)},y)\le d(T_v)$. 结合式(2)，$d(T(v,x)\textcircled{v}T_v)=d(T_v)$；再结合式(4)有 $d(T(v,x)\textcircled{v}T_v)=\max\{d(T_v),l(u^{(1)})+l(v^{(1)})-2l(v)\}$.

(3) 由条件 $l(u^{(1)})<l(v^{(2)})$知 $l(u^{(1)})<l(v^{(1)})$，即 $v^{(1)}$为 $T(v,x)\textcircled{v}T_v$中离 v 最远的端顶点. 由性质 5 知，存在 $T(v,x)\textcircled{v}T_v$的直路径$(v^{(1)}\sim y)$. 我们有 $y\notin V(T(v,x))$. 否则，由性质 3，$d(v^{(1)},y)=l(v^{(1)})+l(y)-2l(v)\le l(v^{(1)})+l(u^{(1)})-2l(v)<l(v^{(1)})+l(v^{(2)})-2l(v)=d(v^{(1)},v^{(2)})\le d(T_v)$，即 $d(T(v,x)\textcircled{v}T_v)=d(v^{(1)},y)<d(T_v)$，与式(2)相矛盾. 因此，$y\in V(T_v)$，即$(v^{(1)}\sim y)$是 T_v中的路径，$d(T(v,x)\textcircled{v}T_v)=d(v^{(1)},y)\le d(T_v)$；再结合式(2)有 $d(T(v,x)\textcircled{v}T_v)=d(T_v)$.　　证毕.

注 4.　v、$T(v,x)$、u 如引理 3 所设. 算法 1 在片段 16 将 $T(v,x)$标为已检查后，引理 3 表明，算法 1 正确地计算了 $d(T_v)$的新值. 其中，引理 3(3)表明，若 $l(u^{(1)})<l(v^{(2)})$，$d(T_v)$保持不变. 引理 3(1)、(2)表明，若 $l(u^{(1)})\ge l(v^{(2)})$，按 $l(u^{(1)})\ge l(v^{(1)})$是否成立，片段 17、片段 18 分别正确地计算了 $d(T_v)$的新值；且 $d(T_v)$的新值只与其原值或 $d(T_u)$以及 $l(u^{(1)})+l(v^{(1)})-2l(v)$有关. 而在计算 $d(T_v)$的新值时，$d(T_u)$是已知的. 事实上，因 u 为待检查顶点，若 u 是端顶点，算法 1 在片段 3 和片段 11，正确地置 $d(T_u)=0$；若 $u\in Q_0$，算法 1 也事先计算了 $d(T_u)$. 此外，不难看出，片段 16 执行后，对于 $w\in V(T)\cap Q_0$，$w\ne v$，T_w无变化.

利用引理 3 及性质 4，可进一步得到下面性质.

性质 7.　设 v 为 Q 的栈顶元素，$T(v,x)$为 v 的未检查的后代分支，u 为 $T(v,x)$中的待检查顶点，则 $d(T(v,x)\textcircled{v}T_v)=\max\{d(T_u),d(T_v),l(u^{(1)})+l(v^{(1)})-2l(v)\}$.

引理 4.　设 v 为 Q 的栈顶元素，$T(v,x)$为 v 的未检查的后代分支，u 为$T(v,x)$中的待检查顶点. 记T^{**}为 v 的、除$T(v,v^{(1)})$外的全部已检查的后代分支在 v 的粘合子树. 如果 $d(T_v)\le d$，$l(u^{(1)})+l(v^{(1)})-2l(v)>d$，则 $d(T(v,x)\textcircled{v}T^{**})=\max\{d(T_u),l(u^{(1)})+l(v^{(2)})-2l(v)\}$，$l(u^{(1)})>l(v^{(2)})$.

证明.　首先，$l(v^{(1)})+l(v^{(2)})-2l(v)\le d(T_v)\le d$. 再由引理条件 $l(u^{(1)})+l(v^{(1)})-2l(v)>d$，得 $l(u^{(1)})>l(v^{(2)})$. 考虑 v 的后代分支 $T(v,x)$和 v 的、除 $T(v,v^{(1)})$外的全部已检查的后代分支 $T(v,v^{(2)}),\cdots$，因为 $l(u^{(1)})>l(v^{(2)})\ge\cdots$，应用引理 3(1)得 $d(T(v,x)\textcircled{v}T^{**})=d(T(v,x)\textcircled{v}T(v,v^{(2)})\textcircled{v}\cdots)=\max\{d(T_u),l(u^{(1)})+l(v^{(2)})-2l(v)\}$.

证毕.

注 5.　v、$T(v,x)$、u 如引理 4 所设，并设 $d(T_v)\le d$. 算法 1 在片段 13 将 $T(v,x)$标为已检查、片段 14 将 $T(v,v^{(1)})-v$ 通过$\ominus$运算删除后，引理 4 表明，算法 1 的片段 15 正确地计算了 $d(T_v)$的新值. 且 $d(T_v)$的新值只与 $d(T_u)$以及 $l(u^{(1)})+l(v^{(1)})-2l(v)$有关. 而在计算 $d(T_v)$的新值时，$d(T_u)$是已知的(参考注 4). 此外，不难看出，片段 13、片段 14 执行后，对于 $w\in V(T)\cap Q_0$，$w\ne v$，T_w无变化.

性质 8.　若 $V(T)\ne\varnothing$，$v\in V(T)$，则$d(T_v)\le d$.

证明.

情形 1. v 为端顶点且 $v\ne v_0$. 显然，$d(T_v)=0\le d$.

情形 2. $v\in Q_0$. 应用归纳法原理证明. 初始阶段，$d(T_v)=0\le d$. 考虑到算法 1 的操作，$d(T_v)$仅当 v 的某个后代分支标为已检查、或者 v 的某个已检查的后代分支中的除 v 外的所有顶点通过$\ominus$运算被删除时才会发生变化. 因此，按照注 4、注 5 的说明，仅需证明：对于 Q 的栈顶元素 v，若 $T(v,x)$为 v 的未检查的后代分支、u 为 $T(v,x)$中的待检查顶点，如果在算法 1 的片段 13、片段 14、片段 16 执行前有 $d(T_y)\le d$(一切 $y\in V(T)$)，则执行后仍然有 $d(T_v)\le d$. 为方便，这里将上述相关片段执行前、后的 $d(T_v)$分别记作 $d(T_v)$、$d'(T_v)$.

(i) 片段 13、片段 14 执行后. 片段 13、片段 14 执行前满足 $l(u^{(1)})+l(v^{(1)})-2l(v)>d$，$l(u^{(1)})<l(v^{(1)})$，$d(T_u)\le d$、$d(T_v)\le d$(归纳法假设). 由引理 4，片段 13、片段 14 执行后，$d'(T_v)=\max\{d(T_u),l(u^{(1)})+l(v^{(2)})-2l(v)\}$，而 $l(u^{(1)})+l(v^{(2)})-2l(v)<l(v^{(1)})+l(v^{(2)})-2l(v)\le d(T_v)\le d$，因此，$d'(T_v)<d$.

(ii) 片段 16 执行后. 片段 16 执行前满足 $l(u^{(1)})+l(v^{(1)})-2l(v)\le d$，$d(T_u)\le d$、$d(T_v)\le d$(归纳法假设). 那么，当片段 16 执行后，

若 $l(u^{(1)})\ge l(v^{(1)})$，由引理 3(1)，$d'(T_v)=\max\{d(T_u),l(u^{(1)})+l(v^{(1)})-2l(v)\}\le d$.

若 $l(u^{(1)})<l(v^{(1)})$且 $l(u^{(1)})\ge l(v^{(2)})$，由引理 3(2)，$d'(T_v)=\max\{d(T_v),l(u^{(1)})+l(v^{(1)})-2l(v)\}\le d$.

若 $l(u^{(1)})<l(v^{(2)})$，由引理 3(3)，$d'(T_v)=d(T_v)\leq d$.

情形 3. v 不是端顶点，且 $v\notin Q_0$. v 不是端顶点表明 v 有唯一子女 y（因为 $V(T)\neq\varnothing$，由引理 1，$v_0\in V(T)$；由 $v\notin Q_0$ 知 $v\neq v_0$. 即 v 不是孤立点）.

(i) y 是已检查顶点. 按算法 1，y 成为已检查顶点必定是在考虑离 v 最近的祖先 $w\in Q_0$ 时，将 w 的 v 所在的后代分支标为已检查所致. 因此，$T_v\subseteq T_w$. 由性质 4 及情形 2，$d(T_v)\leq d(T_w)\leq d$.

(ii) y 是未检查顶点或待检查顶点. 此时 $T_v=\varnothing$，$d(T_v)=0\leq d$. 证毕.

注 6. 按照性质 8，引理 4 中的条件 $d(T_v)\leq d$ 是冗余的，可以去掉.

引理 5. 若 $V(T)\neq\varnothing$，如果 v_0 的所有后代分支均为已检查分支，则 $d(T)\leq d$.

证明. 因为 v_0 的所有后代分支均为已检查分支，故 $T_{v_0}=T$，$d(T)=d(T_{v_0})$. 再由性质 8 知，$d(T)\leq d$. 证毕.

引理 6. 设 v 为 Q 的栈顶元素，$T(v,x)$ 为 v 的未检查的后代分支，u 为 $T(v,x)$ 中的待检查顶点. 如果 $l(u^{(1)})-l(v)>d$，则在路径 $(u\sim v)$ 中存在两个相邻的顶点 z,z_1，使

(1) $l(u^{(1)})-l(z)\leq d$，$l(u^{(1)})-l(z_1)>d$；

(2) 存在 T 的最小 d-子树划分 $\Pi_d^*(T)$，使 $T(z_1,u^{(1)})-z_1\in\Pi_d^*(T)$；

(3) $\pi_d(T)=\pi_d(T\ominus(T(z_1,u^{(1)})-z_1))+1$；

其中 z_1 为 z 的双亲.

证明.

(1) 把路径 $(u\sim v)$ 表示为顶点序列，即 $(u\sim v)=y_0y_1y_2\cdots y_ry_{r+1}$，其中 $y_0=u$，$v=y_{r+1}$. 故 $d(u^{(1)},y_i)=d(u^{(1)},y_{i-1})+w(y_{i-1},y_i)(i=1,2,\cdots,r+1)$. 而 $w(y_{i-1},y_i)\geq 0$，故 $d(u^{(1)},y_0),d(u^{(1)},y_1),\cdots,d(u^{(1)},y_{r+1})$ 是非减序列. 但由性质 8，$d(u^{(1)},y_0)=d(u^{(1)},u)\leq d(T_u)\leq d$. 由性质 2 和引理条件有 $d(u^{(1)},y_{r+1})=d(u^{(1)},v)=l(u^{(1)})-l(v)>d$. 因此，必存在两个相邻的顶点 $z=y_i$，$z_1=y_{i+1}(i=0,1,2,\cdots,r)$，使 $d(u^{(1)},z)\leq d$，$d(u^{(1)},z_1)>d$（显然，$z,z_1\in V(u\sim v)$，$(z,z_1)\in E(u\sim v)$，$z_1\neq u$，$z_1\neq u^{(1)}$，z_1 为 z 的双亲）.

由性质 2，$d(u^{(1)},z)=l(u^{(1)})-l(z)$，$d(u^{(1)},z_1)=l(u^{(1)})-l(z_1)$. 于是 $l(u^{(1)})-l(z)\leq d$，$l(u^{(1)})-l(z_1)>d$.

(2) 记 $T_1'=T(z_1,u^{(1)})-z_1$. 注意，$T(z_1,u^{(1)})$ 是 z_1 的后代分支，且由引理 2(2) 知 $T_1'=T_u\cup(u\sim z)$；由引理 2(1)，u 不是孤立点.

情形 1. $d(u)\geq 2$. 此时 $|V(T_u)|\geq 2$. T_1' 中离 z 最远的端顶点是 $u^{(1)}$. 由性质 5，即知 T_1' 的直路径或者是 T_u 中的一端为 $u^{(1)}$ 的直路径、或者是 $(u^{(1)}\sim z)$，即 $d(T_1')=\max\{d(T_u),d(u^{(1)},z)\}=\max\{d(T_u),l(u^{(1)})-l(z)\}$，由性质 8 和结论(1)有 $d(T_1')\leq d$.

情形 2. $d(u)=1$. 此时 $T_u=\varnothing$，即 $u=u^{(1)}$. 因此，$T_1'=(u\sim z)=(u^{(1)}\sim z)$，$d(T_1')=d((u^{(1)}\sim z))=d(u^{(1)},z)=l(u^{(1)})-l(z)$；由结论(1)有 $d(T_1')\leq d$.

综合情形 1、情形 2，即得 $d(T_1')\leq d$.

设 $\{T_1,T_2,\cdots,T_m\}$ 是 T 的一个最小 d-子树划分，不妨设 $u^{(1)}\in V(T_1)$.

对于任意 $y\in V(T)$，$y\notin V(T_1')$，因为 $T_1'=T(z_1,u^{(1)})-z_1$，故路径 $(u^{(1)}\sim y)$ 必经过 z_1. 于是 $d(u^{(1)},y)=d(u^{(1)},z_1)+d(z_1,y)$，故 $d(u^{(1)},y)\geq d(u^{(1)},z_1)>d$，即 $y\notin V(T_1)$. 可见 $V(T_1)\subseteq V(T_1')$. 记 $T_2'=T_2\ominus T_1'$，$T_3'=T_3\ominus T_1'$，$\cdots$，$T_m'=T_m\ominus T_1'$. 由性质 1 知 $T_i'\subseteq T_i(i=2,\cdots,m)$ 是树. 由性质 4，$d(T_i')\leq d(T_i)\leq d(i=2,\cdots,m)$. 显然，$T_1',T_2',\cdots,T_m'$ 顶点两两不相交. 又

$$\begin{aligned}\bigcup_{i=1}^{m}V(T_i')&=V(T_1')\cup\left(\bigcup_{i=2}^{m}(V(T_i)-V(T_1'))\right)\\&=V(T_1')\cup\left(\bigcup_{i=2}^{m}V(T_i)\right)\supseteq V(T_1)\cup\left(\bigcup_{i=2}^{m}V(T_i)\right)\\&=V(T).\end{aligned}$$

因此，$\{T_1',T_2',\cdots,T_m'\}$ 是 T 的一个 d-子树划分（从而是 T 的一个最小 d-子树划分），结论(2)成立.

(3) 由结论(2)的证明过程得知，$\{T_2',T_3',\cdots,T_m'\}$ 是树 $T\ominus(T(z_1,u^{(1)})-z_1)=T\ominus T_1'$ 的一个 d-子树划分. 因此，$\pi_d(T\ominus(T(z_1,u^{(1)})-z_1))\leq\pi_d(T)-1$，即 $\pi_d(T)\geq\pi_d(T\ominus(T(z_1,u^{(1)})-z_1))+1$. 另一方面，如果 $\{T_2,T_3,\cdots,T_m\}$ 是 $T\ominus(T(z_1,u^{(1)})-z_1)$ 的一个 d-子树划分，则 $\{T(z_1,u^{(1)})-z_1,T_2,T_3,\cdots,T_m\}$ 是 $(T\ominus(T(z_1,u^{(1)})-z_1))\cup(T(z_1,u^{(1)})-z_1)=T-(z_1,z)$ 的一个 d-子树划分，于是 $\{T(z_1,u^{(1)})-z_1,T_2,T_3,\cdots,T_m\}$ 是 T 的一个 d-子树划分，即 $\pi_d(T)\leq\pi_d(T\ominus(T(z_1,u^{(1)})-z_1))+1$. 从而 $\pi_d(T)=\pi_d(T\ominus(T(z_1,u^{(1)})-z_1))+1$. 这样就获得了结论(3). 证毕.

注 7. v、$T(v,x)$、u 如引理 6 所设. 由性质 6，$d(T(v,x))=\max\{d(T_u),l(u^{(1)})-l(v)\}$. 又 $d(T_u)\leq d$. 因此，引理 6 的条件 $l(u^{(1)})-l(v)>d$ 等价于 $d(T(v,x))>d$.

引理 7. 设 v 为 Q 的栈顶元素，$T(v,x)$ 为 v

的未检查的后代分支，u 为 $T(v,x)$ 中的待检查顶点，$l(u^{(1)})-l(v)\leq d$，$l(u^{(1)})+l(v^{(1)})-2l(v)>d$.

(1) 如果 $l(u^{(1)})\geq l(v^{(1)})$，则

① 存在 T 的最小 d-子树划分 $\Pi_d^*(T)$，使 $T(v,x)-v\in\Pi_d^*(T)$；

② $\pi_d(T)=\pi_d(T\ominus(T(v,x)-v))+1$.

(2) 如果 $l(u^{(1)})<l(v^{(1)})$，则

① 存在 T 的最小 d-子树划分 $\Pi_d^*(T)$，使 $T(v,v_1^{(1)})-v\in\Pi_d^*(T)$；

② $\pi_d(T)=\pi_d(T\ominus(T(v,v^{(1)})-v))+1$.

证明.　首先，据注 7 及条件 $l(u^{(1)})-l(v)\leq d$，有 $d(T(v,x))\leq d$. 其次，$v^{(1)}\neq v$，否则 $l(u^{(1)})+l(v^{(1)})-2l(v)=l(u^{(1)})+l(v)-2l(v)=l(u^{(1)})-l(v)\leq d$，与条件 $l(u^{(1)})+l(v^{(1)})-2l(v)>d$ 相矛盾.

设 $\{T_1,T_2,\cdots,T_m\}$ 是 T 的一个最小 d-子树划分，由性质 3 有 $d(u^{(1)},v^{(1)})=l(u^{(1)})+l(v^{(1)})-2l(v)$，据引理条件即得 $d(u^{(1)},v^{(1)})>d$，这意味着不存在 $i(i=1,2,\cdots,m)$ 使 $v^{(1)}\in V(T_i)$，且 $u^{(1)}\in V(T_i)$. 因此，不妨设 $v^{(1)}\in V(T_1)$，$u^{(1)}\in V(T_2)$.

记 $T_1'=(T_1\ominus(T(v,x)-v))\cup(T(v,v^{(1)})-v)$，$T_2'=(T_2\ominus(T(v,v^{(1)})-v))\cup(T(v,x)-v)$，$T_3'=T_3\ominus(T(v,v^{(1)})-v)\ominus(T(v,x)-v)$，$\cdots$，$T_m'=T_m\ominus(T(v,v^{(1)})-v)\ominus(T(v,x)-v)$. 注意 $T(v,v^{(1)})-v\subseteq T_1'$、$T(v,x)-v\subseteq T_2'$.

由性质 1，$T_1\ominus(T(v,x)-v)$ 是树. 但 $v^{(1)}\in V(T_1\ominus(T(v,x)-v))\cap V(T(v,v^{(1)})-v)$，故 T_1' 是树.

我们有 $d(T_1')\leq d$. 事实上，任取 y,z，使 $y,z\in V(T_1')$，只要证明 $d(y,z)\leq d$. 分 4 种情形讨论.

情形 1. $y,z\in V(T_1\ominus(T(v,x)-v))$. $d(y,z)\leq d(T_1\ominus(T(v,x)-v))\leq d(T_1)\leq d$.

情形 2. $y,z\in V(T(v,v^{(1)})-v)$. $d(y,z)\leq d(T(v,v^{(1)})-v)\leq d(T(v,v^{(1)}))\leq d(T_v)\leq d$.

情形 3. $y\in V(T_1\ominus(T(v,x)-v))$ 且 $z\in V(T(v,v^{(1)})-v)$. 不妨设 $y\notin V(T(v,v^{(1)})-v)$（否则归为情形 2），即 y,z 在 v 的不同分支上，于是路径 $(y\sim z)$ 必经过 v. 又由性质 2，$d(v,z)=l(z)-l(v)$. 因 $y,v^{(1)}\in V(T_1)$，故 $d(y,v^{(1)})\leq d(T_1)$. 由性质 3，$d(y,z)=d(y,v)+d(v,z)=d(y,v)+l(z)-l(v)\leq d(y,v)+l(v^{(1)})-l(v)=d(y,v)+d(v^{(1)},v)=d(y,v^{(1)})\leq d(T_1)\leq d$.

情形 4. $y\in V(T(v,v^{(1)})-v)$ 且 $z\in V(T_1\ominus(T(v,x)-v))$. 仿情形 3 的证明，可得 $d(y,z)\leq d$.

类似，可以证明：T_2' 是树，且 $d(T_2')\leq d$.

对于 $i=3,\cdots,m$，由性质 1、T_i' 是树且由性质 4，$d(T_i')\leq d(T_i)\leq d$.

易见 T_1'，T_2'，$\cdots$，T_m' 顶点两两不相交，且 $\bigcup_{i=1}^{m}V(T_i')=V(T)$. 因此，$\{T_1',T_2',\cdots,T_m'\}$ 是 T 的一个 d-子树划分（从而是 T 的一个最小 d-子树划分）.

(1) 当 $l(u^{(1)})\geq l(v^{(1)})$，记 T_1'' 为 $V(T_1')\cup V(T_2'\ominus(T(v,x)-v))$ 的导出子图.

首先，T_1'' 是树. 分两种情形证明.

情形 1. $V(T_2'\ominus(T(v,x)-v))=\varnothing$. 显然，$T_1''$ 是树.

情形 2. $V(T_2'\ominus(T(v,x)-v))\neq\varnothing$. 任取 $y\in V(T_2'\ominus(T(v,x)-v))$. 注意 $T(v,x)-v\subseteq T_2'$. 在 T_2' 中，y 要到达 $T(v,x)-v$ 中的顶点必经过 v，即 $v\in V(T_2')$；但 $v\notin V(T(v,x)-v)$，故 $v\in V(T_2'\ominus(T(v,x)-v))$，即 $v\in V(T_1'')$. 又 $T(v,v^{(1)})-v\subseteq T_1'$，故 $V(T_1')\cup\{v\}$ 的导出子图是连通的. 另外，由性质 1，$T_2'\ominus(T(v,x)-v)$ 是树. 从而 T_1' 中的任意顶点经 v 可到达 y，于是 T_1'' 是连通的，即 T_1'' 是树.

其次，$d(T_1'')\leq d$. 任取 y,z，使 $y,z\in V(T_1'')$，只要证明 $d(y,z)\leq d$. 分 4 种情形讨论.

情形 1. $y,z\in V(T_1')$. $d(y,z)\leq d(T_1')\leq d$.

情形 2. $y,z\in V(T_2'\ominus(T(v,x)-v))$. $d(y,z)\leq d(T_2'\ominus(T(v,x)-v)\leq d(T_2')\leq d$.

情形 3. $y\in V(T_1')$，$z\in V(T_2'\ominus(T(v,x)-v))$.

显然，$z\notin V(T_1')$，$z\notin V(T(v,x)-v)$. 因此，路径 $(u^{(1)}\sim z)$ 必经过 v. 又由上面的讨论易知，路径 $(y\sim z)$ 必经过 v. 注意 $T_1'=(T_1\ominus(T(v,x)-v))\cup(T(v,v^{(1)})-v)$.

(i) $y\in V(T(v,v^{(1)})-v)$. 此时 $d(y,v)\leq d(v^{(1)},v)$. 故 $d(y,z)=d(y,v)+d(v,z)\leq d(v^{(1)},v)+d(v,z)=l(v^{(1)})-l(v)+d(v,z)\leq l(u^{(1)})-l(v)+d(v,z)=d(u^{(1)},v)+d(v,z)=d(u^{(1)},z)\leq d$（注意 $z,u^{(1)}\in V(T_2')$）.

(ii) $y\notin V(T(v,v^{(1)})-v)$. 此时必有 $d(y,v)\leq d(u^{(1)},v)$. 否则 $d(y,v)>d(u^{(1)},v)$，因 y、$v^{(1)}$、$u^{(1)}$ 在 v 的不同分支上，由性质 3，$d(y,v^{(1)})=d(y,v)+d(v,v^{(1)})>d(u^{(1)},v)+d(v,v^{(1)})=d(u^{(1)},v^{(1)})=l(u^{(1)})+l(v^{(1)})-2l(v)>d\geq d(T_1')$，即 $d(y,v^{(1)})>d(T_1')$；与 $y,v^{(1)}\in V(T_1')$ 相矛盾. 从而 $d(y,z)=d(y,v)+d(v,z)\leq d(u^{(1)},v)+d(v,z)=d(u^{(1)},z)\leq d$.

情形 4. $y\in V(T_2'\ominus(T(v,x)-v))$，$z\in$

$V(T_1')$. 仿情形 3 的证明,可得 $d(y,z)\leq d$.

最后,容易验证 $\{T_1'', T(v,x)-v, T_3', \cdots, T_m'\}$ 是 T 的一个 d-子树划分(从而是 T 的一个最小 d-子树划分),于是结论 1)成立.

因为 $\{T_1'', T_3', \cdots, T_m'\}$ 是树 $T\ominus(T(v,x)-v)$ 的一个 d-子树划分,故 $\pi_d(T\ominus(T(v,x)-v))\leq \pi_d(T)-1$,即 $\pi_d(T)\geq\pi_d(T\ominus(T(v,x)-v))+1$. 另一方面,如果 $\{T_1', T_2', \cdots, T_{m-1}'\}$ 是 $T\ominus(T(v,x)-v)$ 的一个 d-子树划分,则 $\{T(v,x)-v, T_1', T_2', \cdots, T_{m-1}'\}$ 是 $(T\ominus(T(v,x)-v))\bigcup T(v,x)$ 的一个 d-子树划分($T\ominus(T(v,x)-v)$ 中含有 v). 而 $V((T\ominus(T(v,x)-v))\bigcup T(v,x))=V(T)$. 因此,$\{T(v,x)-v, T_1', T_2', \cdots, T_{m-1}'\}$ 也是 T 的一个 d-子树划分,即 $\pi_d(T)\leq\pi_d(T\ominus(T(v,x)-v))+1$. 从而 $\pi_d(T)=\pi_d(T\ominus(T(v,x)-v))+1$. 这样就获得了结论 2).

(2)当 $l(u^{(1)})<l(v^{(1)})$,因由引理 4 知 $l(u^{(1)})>l(v^{(2)})$,故 $l(v^{(1)})>l(u^{(1)})>l(v^{(2)})$. 若将 $T(v,x)$、$T(v,v_1^{(1)})$ 换位,以下就可类似(1)证明,不再赘述.

证毕.

注 8. v、$T(v,x)$、u 如引理 7 所设. 由性质 7,有 $d(T(v,x)\ominus T_v)=\max\{d(T_u), d(T_v), l(u^{(1)})+l(v^{(1)})-2l(v)\}$. 又 $d(T_u)\leq d, d(T_v)\leq d$. 因此,引理 7 的条件 $l(u^{(1)})+l(v^{(1)})-2l(v)>d$ 等价于 $d(T(v,x)\ominus T_v)>d$.

根据性质 1,引理 1~引理 7,我们有

定理 2. 算法 1 正确地求解边赋非负权树的 d-子树划分问题.

3.3 算法的复杂性

定理 3. 算法 1 在 $O(n)$ 时间内完成,其中 $n=|V(T_0)|$.

证明. 算法 1 的执行时间主要取决于片段1~片段 4、片段 7~片段 10、片段 12~片段 18 以及循环.

(1)片段 1、片段 2、片段 3. 根据注 1,片段 1 可在 $O(n)$ 时间完成;而片段 2、片段 3 也可在标号 D 进行时同时进行,所耗时间也为 $O(n)$.

(2)片段 4、片段 7. 参考注 2 的说明,考虑到循环(见下面关于循环次数的讨论,下同),总的时间耗费为 $O(n)$.

(3)片段 8. 根据注 3 的讨论,在一次循环中,为了找到 u,片段 8 把 $(u\sim v)$(u,v 如算法 1 或引理 2 约定)上的所有顶点(除 v 外)均访问了一次;但下一次循环片段 8 可能还会访问 $(u\sim v)$ 上的某些顶点,这样片段 8 可能重复访问 $(u\sim v)$ 上的某些顶点. 不过,应用下面的策略,即使考虑到循环,$(u\sim v)$ 上的顶点(除 v 外)被片段 8 恰好访问一次.

片段 9. 我们可以从 u 出发沿着 $(u\sim v)$、通过逐个顶点验证方式来寻找 z, z_1(z, z_1 如算法 1 或引理 6 约定). 由于 u 的所有后代分支均为已检查分支(引理 2),从而 $l(u^{(1)})-l(u)\leq d(T_u)\leq d$(性质 8). 在 $(u\sim v)$ 上取 u 的(唯一)双亲 x_1,如果 $l(u^{(1)})-l(x_1)>d$,则 $z=u, z_1=x_1$;否则,在 $(u\sim v)$ 上取 x_1 的双亲 x_2,如果 $l(u^{(1)})-l(x_2)>d$,则 $z=x_1, z_1=x_2$;… 依此方法,最后必能找到 z, z_1. 因此,片段 9 访问 $(u\sim z_1)$ 上的每个未检查顶点一次;但随后的片段 10 进行的$\ominus$运算将删除这些访问过的顶点(除 z_1 外). 在片段 11,当 z_1 是端顶点且 $z_1\neq v_0$,z_1 将标为待检查,此时,我们可以直接转到片段 8 后面一行的条件判断(取 $u=z_1$. 注意,片段 8 不需访问此种待检查顶点 u)而无需从循环的入口处开始下次循环,避免了片段 8 重复访问 $(u\sim v)$ 上的顶点;若 z_1 不是端顶点,必有 $z_1=v$(意味着前面片段 10 进行的$\ominus$运算删除了 v 的一个分支).

根据以上讨论,我们可得到:

算法 1 直至结束,片段 8 不访问已检查顶点,待检查顶点可能被片段 8 访问一次,不在 Q_0 中的未检查顶点被片段 8 访问一次,不访问 Q_0 中的未检查顶点. 片段 9 不访问已检查顶点和待检查顶点,不在 Q_0 中的未检查顶点被片段 9 访问一次,Q_0 中的未检查顶点 v 被片段 9 访问至多 $d(v)$ 次(若 $v\neq v_0$,v 的每个后代分支各对应一次,共 $d(v)-1$ 次;若 $v=v_0$,v_0 的每个后代分支也各对应一次,共 $d(v)$ 次). 因为已检查顶点、待检查顶点、未检查顶点分别至多有 n 个,故考虑到循环,片段 8、片段 9 访问的顶点总数不超过

$$(n+n)+\Big(n+\sum_{v\in Q_0, v\neq v_0}(d(v)-1)+d(v_0)\Big)=O(n).$$

(4)片段 10、片段 12、片段 14. 主要时间耗费在$\ominus$运算. 考虑到循环,全部$\ominus$运算所耗费的时间为 $O(n)$.

(5)片段 15、片段 17、片段 18. 分别参考注 5、注 4 的说明. 考虑到循环,总的时间耗费为 $O(n)$.

(6)片段 13、片段 16. 实际上,片段 13、片段 16 只需把不是已检查顶点的顶点标为已检查(即路径 $(u\sim v)$ 中除 v 外的顶点,引理 2). 注意,任何顶点只要标为已检查后,不可能标为未检查、待检查. 考虑到循环,总的时间耗费为 $O(n)$.

(7)循环. 每次循环将处理 T 中对应 Q 的栈顶元素的顶点的一个后代分支,该顶点的某个后代分

支每多处理一次，表明前次循环刚刚得到了 $\Pi_d^*(T_0)$ 的一个元素. 因此，循环次数不超过

$$\pi_d(T_0)+\sum_{v\in Q_0,v\neq v_0}(d(v)-1)+d(v_0)=O(n).$$

综上所述，算法 1 的时间耗费为 $O(n)$.　证毕.

4　算　例

例. 求图 2 所示树 T 的 3-子树划分问题，其中顶点旁的数字为顶点编号（例如顶点编号 1 表示顶点 v_1、顶点编号 10 表示顶点 v_{10}），标在边的上方或右边的数字为对应边的权值.

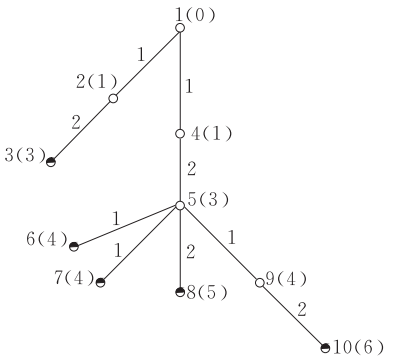

图 2　按标号 D 标号的树

解. 现用算法 1 求解. 置 $d=3$.

(1) 初始化.

$\pi_3(T)=0$；$\Pi_3^*(T)=\varnothing$.

取 $v_0=v_1$，将 T 按标号 D 标号（见图 2，顶点的标号写在顶点编号后面的括号中），$Q=\{v_5,v_1\}$.

把所有的端顶点标为待检查，其余顶点标为未检查（见图 2，未检查顶点、已检查顶点、待检查顶点分别用白色、黑色、半白半黑小圆圈表示. 下同）.

对 $i=1,3,5,6,7,8,10$，置 $v_i^{(1)}=v_i$，$v_i^{(2)}=v_i$，$d(T_{v_i})=0$.

(2) 取 Q 的栈顶元素 v_5，$v=v_5$，取 v_5 的最左边的未检查的后代分支 $T(v_5,v_6)$，取 $T(v_5,v_6)$ 中的待检查顶点 $u=v_6$. 因为 $l(u^{(1)})-l(v)=l(v_6^{(1)})-l(v_5)=l(v_6)-l(v_5)=4-3=1<d$，且 $l(u^{(1)})+l(v^{(1)})-2l(v)=l(v_6^{(1)})+l(v_5^{(1)})-2l(v_5)=l(v_6)+l(v_5)-2l(v_5)=4+3-2\times3=1<d$，故将 $T(v_5,v_6)$ 标为已检查. 又因 $l(u^{(1)})=l(v_6^{(1)})=l(v_6)=4$，$l(v^{(1)})=l(v_5^{(1)})=l(v_5)=3$，即 $l(u^{(1)})>l(v^{(1)})$，故 $d(T_v)=d(T_{v_5})=\max\{d(T_u),l(u^{(1)})+l(v^{(1)})-2l(v)\}=\max\{d(T_{v_6}),l(v_6^{(1)})+l(v_5^{(1)})-2l(v_5)\}=\max\{0,4+3-2\times3\}=1$；$v^{(2)}=v_5^{(2)}=v_5^{(1)}=v_5$；$v^{(1)}=v_5^{(1)}=u^{(1)}=v_6^{(1)}=v_6$. 即得 $T(v_5,v_6)$ 标为已检查，$d(T_{v_5})=1$，$v_5^{(1)}=v_6$，$v_5^{(2)}=v_5$.

(3) 继续考虑 Q 的栈顶元素 v_5，$v=v_5$，取 v_5 的最左边的未检查的后代分支 $T(v_5,v_7)$，取 $T(v_5,v_7)$ 中的待检查顶点 $u=v_7$. 类似(2)可得：$T(v_5,v_7)$ 标为已检查，$d(T_{v_5})=2$，$v_5^{(1)}=v_7$，$v_5^{(2)}=v_6$.

(4) 继续考虑 Q 的栈顶元素 v_5，$v=v_5$，取 v_5 的最左边的未检查的后代分支 $T(v_5,v_8)$，取 $T(v_5,v_8)$ 中的待检查顶点 $u=v_8$. 类似(2)可得：$T(v_5,v_8)$ 标为已检查，$d(T_{v_5})=3$，$v_5^{(1)}=v_8$，$v_5^{(2)}=v_7$.

(5) 继续考虑 Q 的栈顶元素 v_5，$v=v_5$，取 v_5 的最左边的未检查的后代分支 $T(v_5,v_9)$，取 $T(v_5,v_9)$ 中的待检查顶点 $u=v_{10}$. 因为 $l(u^{(1)})-l(v)=l(v_{10}{}^{(1)})-l(v_5)=l(v_{10})-l(v_5)=6-3=3=d$，但 $l(u^{(1)})+l(v^{(1)})-2l(v)=l(v_{10}{}^{(1)})+l(v_5^{(1)})-2l(v_5)=l(v_{10})+l(v_8)-2l(v_5)=6+5-2\times3=5>d$，$l(u^{(1)})=l(v_{10}^{(1)})=l(v_{10})=6$，$l(v^{(1)})=l(v_5^{(1)})=l(v_8)=5$，即 $l(u^{(1)})>l(v^{(1)})$，故 $\Pi_3^*(T)=\Pi_3^*(T)\bigcup\{T(v_5,v_9)-v_5\}=\{\{v_9,v_{10}\}\}$（为了简明起见，$\Pi_3^*(T)$ 中的子树用该子树的顶点集表示. 下同），$\pi_3(T)=\pi_3(T)+1=1$，$T=T\ominus(T(v_5,v_9)-v_5)$（见图 3）.

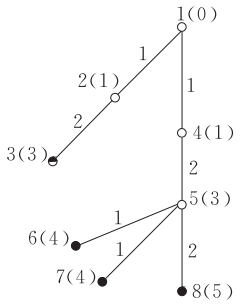

图 3　更新后的树

(6) 继续考虑 Q 的栈顶元素 v_5，$v=v_5$，$v\neq v_0$ 且不存在 v_5 的未检查的后代分支，于是将 v_5 标为待检查；$Q=Q-\{v\}=\{v_1\}$.

(7) 取 Q 的栈顶元素 v_1，$v=v_1$，取 v_1 的最左边的未检查的后代分支 $T(v_1,v_2)$，取 $T(v_1,v_2)$ 中的待检查顶点 $u=v_3$. 类似(2)可得：$T(v_1,v_2)$ 标为已检查，$d(T_{v_1})=3$，$v_1^{(1)}=v_3$，$v_1^{(2)}=v_1$.

(8) 继续考虑 Q 的栈顶元素 v_1，$v=v_1$，取 v_1 的最左边的未检查的后代分支 $T(v_1,v_4)$，取 $T(v_1,v_4)$ 中的待检查顶点 $u=v_5$. 易得 $\Pi_3^*(T)=\{\{v_9,v_{10}\},\{v_5,v_6,v_7,v_8\}\}$，$\pi_3(T)=2$；$v_4$ 标为待检查，$v_4^{(1)}=v_4$，$v_4^{(2)}=v_4$，$d(T_{v_4})=0$.

(9) 继续考虑 Q 的栈顶元素 v_1，$v=v_1$，取 v_1 的最左边的未检查的后代分支 $T(v_1,v_4)$，取 $T(v_1,v_4)$ 中的待检查顶点 $u=v_4$. 易得 $T(v_1,v_4)$ 标为已检查，$\Pi_3^*(T)=\{\{v_9,v_{10}\},\{v_5,v_6,v_7,v_8\},\{v_2,v_3\}\}$，$\pi_3(T)=3$；$d(T_{v_1})=1$，$v_1^{(1)}=v_4$.

(10) 继续考虑 Q 的栈顶元素 v_1，$v=v_1$，$v=v_0$ 且不存在 v_1 的未检查的后代分支，于是 $\Pi_3^*(T)=\Pi_3^*(T)\bigcup\{T\}=\Pi_3^*(T)\bigcup\{(v_1,v_4)\}=\{\{v_9,v_{10}\},\{v_5,v_6,v_7,v_8\},\{v_2,v_3\},\{v_1,v_4\}\}$，$\pi_3(T)=\pi_3(T)+1=4$，$T=\varnothing$.

从而 $\Pi_3^*(T)=\{\{v_9,v_{10}\},\{v_5,v_6,v_7,v_8\},\{v_2,v_3\},\{v_1,v_4\}\}$，$\pi_3(T)=4$.

算法 1 结束.

5 结束语

由于对结构非常简单的图，d-子树划分问题是一个 NP-完全问题，因此，研究一般图的 d-子树划分问题的多项式时间近似算法非常有必要. 当然，寻找在实际应用中常见网络类型或理论上有较大价值的特殊类图(如区间图、仙人掌图)的 d-子树划分问题的多项式时间精确算法也值得重视.

参 考 文 献

[1] Yan J-H, Chang G J. The path-partition problem in block graphs. Information Processing Letters, 1994, 52(6): 317-322

[2] Steiner G. On the k-path partition of graphs. Theoretical Computer Science, 2003, 290(3): 2147-2155

[3] Cai Yan-Guang, Zhang Xin-Zheng, Qian Jin-Xin, Sun You-Xian. An $O(n)$ algorithm for ω-path partition of edge-weighted forests. Journal of Software, 2003, 14(5): 897-903(in Chinese)
(蔡延光，张新政，钱积新，孙优贤. 边赋权森林 ω-路划分的 $O(n)$算法. 软件学报，2003，14(5)：897-903)

[4] Arikati S R. Graph clustering: Complexity, sequential and parallel algorithms [Ph. D. dissertation]. Department of Computer Science, University of Alberta, Edmonton, 1994

[5] Misra J, Tarjan R E. Optimal chain partitions of trees. Information Processing Letters, 1975, 4(1): 24-26

[6] Bonuccelli M A, Bovet D P. Minimum node disjoint path covering for circular-arc graph. Information Processing Letters, 1979, 8(4): 159-161

[7] Arikati S R, Rangan C P. Linear algorithm for optimal path cover problem on interval graphs. Information Processing Letters, 1990, 35(3): 149-153

[8] Moran S, Wolfstahl Y. Optimal covering of cacti by vertex-disjoint paths. Theoretical Computer Science, 1991, 84(2): 179-197

[9] Damaschke P, Deogun J S, Kratsch D, Steiner G. Finding Hamiltonian paths in cocomparability graphs using the bump number algorithm. Order, 1992, 8(4): 383-391

[10] Srikant R, Sundaram R, Singh K S, Rangan C P. Optimal path cover problem on block graphs and bipartite permutation graphs. Theoretical Computer Science, 1993, 115(2): 351-357

[11] Wong P-K. Optimal path cover problem on block graphs. Theoretical Computer Science, 1999, 225(1-2): 163-169

[12] Lin R, Olariu S, Pruesse G. An optimal path cover algorithm for cographs. Computers & Mathematics with Applications, 1995, 30(8): 75-83

[13] Nakano K, Olariu S, Zomaya A Y. A time-optimal solution for the path cover problem on cographs. Theoretical Computer Science, 2003, 290(3): 1541-1556

[14] Hung R-W, Chang M-S. Finding a minimum path cover of a distance-hereditary graph in polynomial time. Discrete Applied Mathematics, 2007, 155(17): 2242-2256

[15] Asdre K, Nikolopoulos S D, Papadopoulos C. An optimal parallel solution for the path cover problem on P_4-sparse graphs. Journal of Parallel and Distributed Computing, 2007, 67(1): 63-76

[16] Pan J-J, Chang G J. Path partition for graphs with special blocks. Discrete Applied Mathematics, 2005, 145(3): 429-436

[17] Yan J-H, Chung G J, Hedetniemi S M, Hedetniemi S T. k-Path partition in trees. Discrete Applied Mathematics, 1997, 78(1-3): 227-233

[18] Jin Z, Li X. On the k-path cover problem for cacti. Theoretical Computer Science, 2006, 355(3): 354-363

[19] Bondy J A, Murty U S R. Graph Theory with Applications. New York: The MacMillan Press Ltd., 1976

[20] Garey M R, Johnson D S. Computers and Intractability: A Guide to the Theory of NP-Completeness. San Francisco: W. H. Freeman, 1979

[21] Yannkakis M, Gavril F. Edge dominating sets in graphs. SIAM Journal on Applied Mathematics, 1980, 38(3): 364-372

[22] Handler G R, Mirchandani P B. Location on Networks. Cambridge: The MIT Press, 1979

CAI Yan-Guang, born in 1963, Ph. D., professor. His current research interests include control and optimization of network, combinatorial optimization, intelligent optimization and intelligent decision supporting system.

ZHANG Yun, born in 1963, Ph. D., professor, Ph. D. supervisor. His current research interests include control and optimization of network, integration of control network, intelligent control and information processing technology.

QIAN Ji-Xin, born in 1939, professor, Ph. D. supervisor. His current research interests include modeling, control and optimization of complex system.

Background

The research is supported by the National Natural Science Foundation of China under grant No. 60374062, the Team Project of the Natural Science Foundation of Guangdong Province under grant No. 8351009001000002, the Science and Technology Program of Guangdong Province under grant No. 2007B010200070 and No. 2008B010200005. These granted projects focus on operating management, control and scheduling of complex network system, in which the authors have worked for more than twenty years.

The d-subtree partition problem arises from operation management, maintenance and test of network. It is theoretically a generalization of the ω-path partition problem, the k-path partition problem and the path partition problem which has been studied extensively. When the subtree is replaced by the path and d is a nonnegative real ω, the d-subtree partition problem becomes the ω-path partition problem. When the graphs is not edge-weighted and ω is an integer k, the ω-path partition problem becomes the k-path partition problem. When $k \geq |V(G)| - 1$, which means that there is no limit to the length of the path, the k-path partition problem becomes the path partition problem. A generalization of the d-subtree partition problem is the graph clustering problem, including the problem of partitioning graphs into a minimum number of subgraphs of bounded diameter.

It is easy to derive that the path partition problem is NP-complete from the NP-Completeness of Hamiltonian path, so is the k-path partition problem and the ω-path partition problem. The path partition problem is polynomial solvable only for several special classes of graphs such as trees, circular arc graphs, interval graphs, cacti, cocomparability graphs, block graphs, bipartite permutation graphs, cographs, distance-hereditary graphs, P_4-sparse graphs and graphs whose blocks are complete graphs, cycles or complete bipartite graphs. The k-path partition problem is polynomial solvable only for trees, bipartite permutation graphs and cacti. The ω-path partition problem is polynomial solvable only for trees and forests. Since the d-subtree partition problem is more complicated, it is not easy to be solved either.

In this paper, it is proved that the d-subtree partition problem is NP-complete for bipartite planar graphs with nonnegative edge-weights for any positive real d. This shows that the d-subtree partition problem is quite difficult even if for the graph with very simple structure. Tree is a subclass of bipartite planar graph. It is also one of the most common network topology in theoretical research and practical applications. We present a linear time algorithm to solve the d-subtree partition problem for trees with nonnegative edge-weights. To improve its performance, realization strategies for the algorithm are discussed in detail. The algorithm presented can be realized easily and requires less time.