Information Processing Letters 23 (1986) 317–319
North-Holland

3 December 1986

A LINEAR ALGORITHM FOR THE CUTTING CENTER OF A TREE

Frank HARARY

Department of Mathematics, The University of Michigan, Ann Arbor, MI 48109, U.S.A.

Peter J. SLATER

Department of Mathematics and Statistics, School of Science, The University of Alabama in Huntsville, Huntsville, AL 35899, U.S.A.

Communicated by David Gries
Received 19 June 1985
Revised 2 December 1985

As a measure of the extent to which the removal of a node disconnects a graph, the cutting number c(v) of a node v in a connected graph G has been defined to be the number of pairs of nodes in different components of $G-\{v\}$. We present a linear algorithm for determining c(v) for all nodes of a tree, and hence for identifying the cutting center, which consists of the nodes v at which c(v) is maximized.

Keywords: Tree, cutting center

Dedicated to the memory of Phillip A. Ostrand

1. Introduction

The cutting number c(v) of a node v in a connected graph G is the number of unordered pairs {u, w} of nodes such that every path from u to w contains v. In other words, c(v) is the number of pairs of nodes that lie in different components of the subgraph $G-\{v\}$. For the tree T1 of Fig. 1, deleting node 6 leaves five components of order one and another of order eight, and $c(6)=50$. Also, $c(8)=48$. The *cutting center* of a graph is the set of nodes with maximum cutting number. For example, the cutting center of T1 is $\{6,9\}$. This concept arose from a structural model in psychology [2,3], but it is applicable to any network in which there are communicating processes involving the pairs of nodes of the network.

We present a rather surprising linear algorithm —i.e., linear in the number of nodes of the tree—for computing the cutting number of all nodes of a tree. From the list of cutting numbers, it is easy to find the cutting center by determining the maximum of the cutting numbers and forming the set of all nodes with that cutting number.

2. The cutting center algorithm for trees

We assume that a tree T has a root (which can be chosen arbitrarily) and that T is represented by three items: p, the number of nodes in T, an

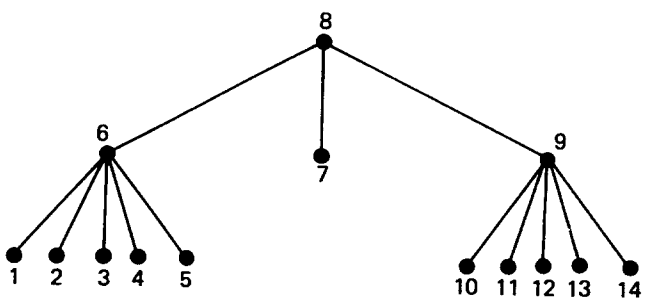

Fig. 1. Tree T1 with cutting center $\{6,9\}$.

endnode list $EL = (v_1, \ldots, v_p)$, and an associated parent list $PA = (u_1, \ldots, u_{p-1})$.

The endnode list is any enumeration of the nodes of T such that each node precedes its parent. In the associated parent list, each u_i is the parent of v_i in T. Note that PA has length $p - 1$ and not p, since the root v_p has no parent. As an example, tree T1 of Fig. 1 has an endnode list

EL = (1, 2, 3, 10, 11, 4, 5, 7, 6, 12, 13, 14, 9, 8)

with the associated parent list

PA = (6, 6, 6, 9, 9, 6, 6, 8, 8, 9, 9, 9, 8).

These lists can be constructed for a tree of p nodes in O(p) time (see, e.g., [4,5]), so requiring them does not increase the order of execution time of the algorithm.

The correctness of the algorithm rests on the following analysis. Consider any node v, and let the components of $T - v$ be $C_1, \ldots, C_k, C_{k+1}$ with C_{k+1} containing the parent of v (if v is not the root) (see Fig. 2). Denote by v_i the node in C_i that is adjacent to v. Let d_i denote the number of descendants of v_i so that $|C_i| = d_i + 1$ for $1 \leqslant i \leqslant k$. Then, the number of descendants of v is $(d_1 + 1) + (d_2 + 1) + \cdots + (d_k + 1)$. Furthermore,

$$c(v) = \sum_{1 \leqslant i < j \leqslant k+1} |C_i| \cdot |C_j| \qquad (1)$$

$$= \frac{1}{2}\Big(\sum_{1 \leqslant i \leqslant k+1} |C_i| \cdot (p - 1 - |C_i|) \Big). \qquad (2)$$

Line (1) is simply the definition of c(v). For (2), simply note that each $w \in C_i$ is separated by v from every vertex except v and those in C_i, that is, from $p - 1 - |C_i|$ of them. Each pair $w \in C_i$ and $x \in C_h$ with $j \neq h$ is counted twice (when $i = j$ and when $i = h$), so one divides by 2.

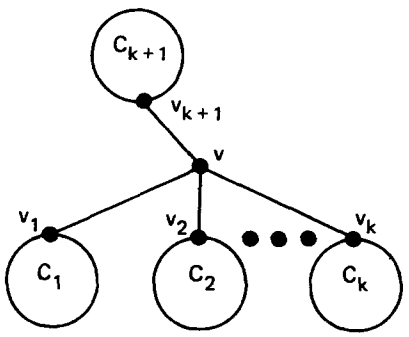

Fig. 2. Computing the cutting number of node v.

The algorithm computes two arrays d(1..p) and c(1..p) which are defined as

d(i) = the number of descendants of node i,
c(i) = the cutting number of node i.

These arrays are computed in a single left-to-right scan of lists EL and PA. An important thing to note in the algorithm is that each node in EL occurs and is processed before its parent (that is, if EL(j) = PA(i), then $j > i$).

The loop invariant, which indicates the values of c and d just before each loop iteration, is as follows:

PD: For each j, d(j) is the number of descendants of j in all subtrees with a root r in EL(1..i − 1) for which j is the parent of r.

PC: For each j, c(j) is the sum of the terms $|C_h| * \frac{1}{2}(p - 1 - |C_h|)$, where C_h is a subtree with a root r in EL(1..i − 1) for which j is the parent of r, plus, if j is in EL(1..i − 1), this term for the component of $T - \{j\}$ containing the parent of node j (see formula (2)).

Algorithm. Store in each d(j) the number of descendants of node j and in each c(j) the cutting number of node j, for $1 \leqslant j \leqslant p$:

```
c := 0; d := 0;
{invariant: PD ∧ PC}
for i := 1 to p − 1 do
  begin
    d(PA(i)) := d(PA(i)) + d(EL(i)) + 1;
    c(PA(i)) := c(PA(i)) + (d(EL(i)) + 1)
                         * ½(p − 2 − d(EL(i)));
    c(EL(i)) := c(EL(i)) + (p − 1 − d(EL(i)))
                         * ½(d(EL(i)))
  end.
```

The first statement of the loop body augments the number of descendants of the parent PA(i) of vertex EL(i) by the number of vertices in component $T - PA(i)$ that contains EL(i). The second statement adds a term in (2) for one of the first k components C_k for node PA(i), while the third adds the term for component $k + 1$ of node EL(i).

Acknowledgment

The authors gratefully acknowledge the considerable efforts of Professor David Gries to improve the form of this paper.

References

[1] F. Harary, Graph Theory (Addison-Wesley, Reading, MA, 1969).

[2] F. Harary and P.A. Ostrand, How cutting is a cutpoint?, in: Combinatorial Structures and Their Applications (Gordon & Breach, New York, 1970) 147–149.

[3] F. Harary and P.A. Ostrand, The cutting center theorem for trees, Discrete Math. 1 (1971) 7–18.

[4] D.E. Knuth, The Art of Computer Programming, Vol. 1: Fundamental Algorithms (Addison-Wesley, Reading, MA, 1968) 334–338.

[5] R.E. Tarjan, Depth-first search and linear graph algorithms, SIAM J. Comput. 1 (1972) 146–160.