International Journal of Computer Applications (0975 – 8887)
Volume 54– No.18, September 2012

Algorithmic Approach to Eccentricities, Diameters and Radii of Graphs using DFS

Ishwar Baidari
Asst Professor
Dept of Computer Science
Karnatak University, Dharwad

Ravi Roogi
Research Scholar
Dept of Computer Science
Karnatak University, Dharwad

Shridevi Shinde
Research Scholar
Dept of Computer Science
Karnatak University,Dharwad

ABSTRACT

Let $G = (V, E)$ be a graph. The distance $d(u, v)$ between two nodes u and v is the length of the shortest path between them. The eccentricity $E(v)$ of a graph vertex v in connected graph G is the maximum distance between v and any other vertex u of G. i.e. $max_{u \in V}\{ d(u, v) \}$.The diameter of the graph is a graph the longest shortest path between any two graph vertices (u, v) of a graph i.e. $Diam(G) = max \{ E(v) / v \in V\}$.The minimum eccentricity of a graph is radius i.e. $Rad(G) = min \{ E(v) / v \in V\}$. In this paper we propose algorithms for finding eccentricity diameter and radius of a tree using DFS.

Keywords: Eccentricity, Radius, Distance, Diameter, and Graph.

1. INTRODUCTION

A graph is a collection of points and lines connecting some of them. The points of a graph are most commonly known as graph vertices. Similarly the lines connecting the vertices of a graph are most commonly known as graph edges. Formally we can define a graph as a graph G is a pair of sets V and E together with a function. $f: E \rightarrow V \times V$ The elements of V are vertices and the elements of E are edges. Connections between the points come in two forms those that are non-directional and those that have an implicit direction are undirected and directed respectively.

In a graph theory a tree is connected acyclic graph stated otherwise trees and graph are undirected. A tree is called a rooted tree if one vertex has been designated the root in which case the edges have a natural orientation towards or away from the root.

1. Eccentricity

The concept of eccentricity is fundamental in graph theory. In this paper we are designing an algorithm for finding eccentricity of a tree. Tree is connected graph with no cycles. In an undirected tree a leaf is a vertex of degree*1*. Some basic properties of trees are:

1. Every tree with at least one edge has at least two leaves. If the minimum degree of a graph is at least *2*, then that graph must contain a cycle.

2. Every tree on n vertices has exactly n-*1*edges.

1.1 Eccentricity of a tree: the eccentricity of a vertex v in a graph G. Denoted $ecc(v)$, is the distance from v to a vertex farthest from v that is

$$ecc(v) = max_{x \in VG}\{d(v,x)\}.$$

A central vertex of a graph is a vertex with minimum eccentricity. We begin with some existing [1] preliminary results concerning the eccentricity of vertices in a tree.

Lemma 1: Let T be a tree with at least three vertices

a) If v is a leaf of T and w is its neighbor, then

$ecc(v) = ecc(w)+1$.

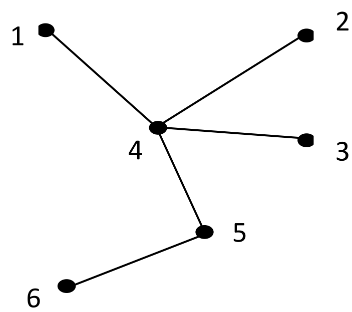
b) If v is a central vertex of T, then

$deg(v) = \geq 2$.

Lemma 2: Let v and w be two vertices in a tree T such that w is of maximum, distance from v (i.e. $ecc(v) = d(v, w)$) then w is a leaf.

Lemma 3: Let T be a tree with at least three vertices, and let T^* be the subtree of T obtained by deleting from T all its leaves. If v is a vertex of T^*, then $ecc_T(v) = ecc\ T^*(V)+1$.

Let T be tree



Consider the above tree, and then the eccentricity of each vertex in the tree is given below.

$E(1) = 3$
$E(2) = 3$
$E(3) = 3$
$E(4) = 2$
$E(5) = 2$
$E(6) = 3$

Algorithm: Eccentricity, Diameter and Radius of a tree
Input: n - number of nodes
Cost[50][50] - adjacency matrix
Output: Eccentricity, Diameter and Radius

1. Input the number of nodes and adjacency matrix
2. for $i \leftarrow 0$ *to n-1*
   for $j \leftarrow 0$ *to n-1*
   D[i][j]=cost[i][j]
   End *j*
   End *i*
3. To find the shortest distance
   for $k \leftarrow 0$ *to n-1*
   for $i \leftarrow 0$ *to n-1*
   for $j \leftarrow 0$ *to n-1*
   D[i][j] = min1(D[i][j],D[j][k],D[k][i])
   End j
   End i
   End k

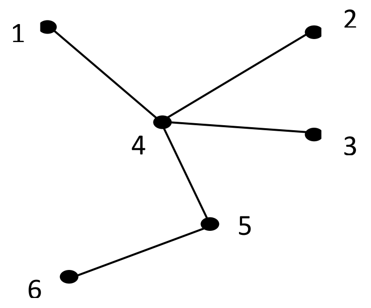
4. **To find eccentricity**
   for $i \leftarrow 0$ *to n-1*
   Initialize *max = D[i][0]*
   for $j \leftarrow 0$ *to n-1*
   if *D[i][j]>max*
   max = D[i][j]
   End *j*
   E[i] = max
   End i

2. Diameter:

Determining the diameter of a graph is a fundamental seemingly quite time consuming operation but we are restricted it to tree. Recall that the eccentricity of vertex x $.ecc(x) = max_{y \in V}$ d(x, y), where d(x, y) denotes the distance between x and y: the diameter of G equals the maximum eccentricity of any vertex in V

Let G be a graph and v be a vertex of G. The diameter of G is the maximum eccentricity among the vertices of e.

Thus, diameter *(G) = max {e (v): v in V (G)}*



E (1) = 3
E (2) = 3
E (3) = 3
E (6) = 3

Diameter is 3.

Let us now consider the existing properties [1] to help us to find the diameter of tree.

2.1 **Fact:** Suppose that $SP_T(v_1, v_2)$ is a diameter of T and r is a vertex on the diameter. For any vertex x,
$d_T(x,r) \le max\{ d_T(r, v_1), d_T(r,v_2)\}$.

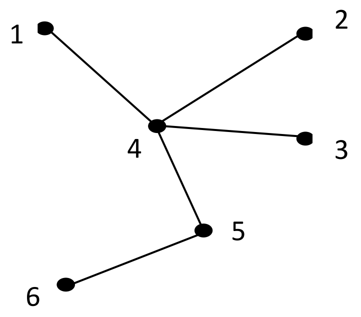
3.2 **Lemma:** Let r be any vertex in a tree T, If v is the farthest vertex to r , the eccentricity of v is the diameter of T.

Algorithm: Diameter

5. **To find Diameter**
   Initialize min and *max = e [0]*
   for $i \leftarrow 0$ *to n-1*
   if *e[i] > max*
   max = e[i]
   else if *e[i]< min*
   min = e[i]
   End *i*
   Diameter = *max*.

3. Radius:

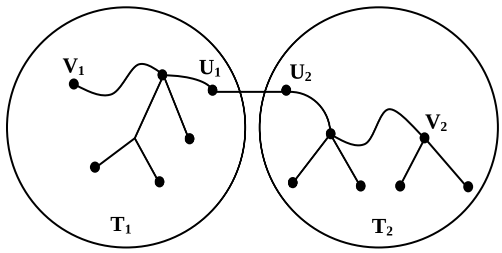
The minimum eccentricity of all points in a graph is called the radius *r (G)* of the graph. The radius can be obtained from a diameter**.**
The radius of G is the minimum eccentricity among the vertices of G. Therefore radius *(G) = min {e (v) / v in V (G)}*



E (4) = 2
E (5) = 2

Radius of the graph is 2.

Suppose that $p = SP_T(v_1, v_2)$ is a diameter. Starting at $V_1$and travelling along the path p. We compute the distance $d_T(u_1,v_1)$ for each vertex u on the path. Let u_1 be the last encountered vertex Such that $d_T(u_1,v_1) \le \frac{1}{2} w(p)$ and u_2 be the next vertex to u_1 as shown in the below fig [1]



3.1 Algorithm: Radius

6. **To find the radius**
   Initialize *min* and *max = e [0]*
   for $i \leftarrow 0$ *to n-1*
   if *e[i] > max*
   max =e[i]
   else if *e[i] <min*
   min = e[i] end *i*
   Radius = *min*

5. We proposed another program which find the eccentricity diameter and radii of graphs using DFS. We implemented it through linked list

5.1

```
#include<stdio.h>
#define FALSE 0
# define TRUE 1
# define SIZE 15
Typedef struc node
{
        int inf;
        int weight;
        struct node *link;
}node;
typedef struct table
{
        int visit;
        char data;
        node *nodeptr;
}table;
table *tab[SIZE]
int max=0,n,e[50],I,j;
Void dfs(int);
Void create(int)
Void main()
{
        int start, radius, center[20],diameter,min;
        node *cur;
        clrscr();
        Printf("Enter the no of nodes:");
        Scanf("%d",&n);
        Create(n);
        for(i=0;i<n;i++)
        {
                e[i]=0;
                max = 0;
                for(j=0;j<n;j++)
                tab[i]-> visit = FALSE;
                tab[i]-> visit = TRUE;
                dfs(i);
                e[i]=max;
                for(j=0;j<n;j++)
                tab[j]->visit=FALSE;
        }
        for(i=0;i<n;i++)
        Printf("n\ Eccentricity of %d is %d\n',I,e[i]);
        Min=e[0];
        Max=e [0];
        for(i=1;i<n;i+)
        {
                if(e[i].max)
                max=e[i];
                if(e[i]<min)
                min =e[i];
        }
                radius =min;
                daimeter=max;
                printf("Radius = %d\n",radius);
                printf("Diameter = %d\n",diameter");

Void create(int n)
{
        Node *new1,*temp;
        printf("Enter the elements of the matrix below:\n");
        for (i=0;i<n;i++)
{
        tab[i] =(table*)malloc(size of (table));
        tab[i] –> visit = FALSE;
        tab[i] -> data = 'A'+I;
        tab[i] -> nodeptr = NULL;
        top = NULL;
        for (j=0;j<n;j++)
        {
        Printf("is there is edge from %d to %d\n",I,j);
                scanf("%d",&item)
                if (item== 1)
                {
                        Printf("Enter the Weight\n");
                        Scanf("%d",&w);
                }
                        else
                        w = 0;
                        if(item)
                {
                new1= (node*)malloc(size of (node));
                        new 1 -> info =j;
                        new1 -> weight = w;
                        new1 -> link = NULL;
                        if (temp)
                        temp -> link =new;
                        else
                        tab[i] -> nodeptr = new1;
                        temp = new;
                }
                }
        }
{
        for (i=0; i<n;i++)
        {
        if (e[i] = = radius)
                {
                Center[i];
                Printf("Center Vetex is %d",center [i]);
                }
        }
        getch();
}
Void dfs(int u)
{
        node *cur;
        int k;
        k = e[i]
        cur = tab[u] -> nodeptr;
        while(cur)
        {
                if (tab[cur -> info] -> visit = = FALSE)
                {
                        E[i] +=cur -> weight;
                        tab[cur ->info] -> visit = TURE;
                        dfs(cur -> info);
                        if (max <- e[i];
                        {
                                (max = e[i]);
                        }
                        e[i] = k;
                        {
                        cur = cur ->link;
                        }
                }
        }
}
```

REFERENCES

[1]. A note on Eccentricities, diameters, and radii Bang Ye Wu Kun–Mao Chao

[2]. Alan Gibbons, Algorithmic Graph Theory. Cambridge University Press. 1999

[3]. Lich – Hsing Hsu and Cheng- kuan Lin Graph Theory and Interconnection Networks.CRC Press 2009.

[4]. Alfred V Aho, John E, Hopcroft and Jeffrey D. Ullman Data structures and Algorithms. Pearson Education 2006.

[5]. Thomas H Cormen Charles E Leiserson and Ronald L, Rivest. Algorithms PHI 2001.

[6]. Geir Agnarsson, Raymond Greenlaw. Graph Theory Modeling Applications and Algorithms.

[7]. Dieter Jungnickel Graphs Networks and Algorithms Springer 2006.

[8]. E COCKAYNE and S.GODDMAN and .HEDETINIEMI. A Liner Algorithm for the Domination Number of a Tree

[9]. B.S.Panda and D.Pradhan. Locally Connected Spanning Trees in Cographs, Complements of Bipartite Graph and Doubly Chordal Graph.

[10]. Gary chartarand, Ortrud R Oellermann, Applied and Algorithmic Graph Theory Mc Graw-Hill Inc 1993