Counting leaves

胡亮泽

Date: 2013-10-1

Chapter 1: Introduction

Trees are common in programming , so it would be very useful to count the leaves of a tree quickly using a algorithm whose time complexity is relatively small. The algorithm we show this time would be able to count and output the leaves of a tree input in each level,which are nodes that have no child in a tree.

Chapter 2: Algorithm Specification

Structure introduction:

In this algorithm, we declare two main structures,named childLink and Node.

Here are the code for them:

```
1. struct childLink
{
    int id;
    childLink *next  ;
    childLink(){next=0;}
};
2. struct Node      {
    int childNum;
    childLink *childHead;
    Node(){childNum=0;childHead=0;}
}*node;
```

Childlink consists of an integer(id) representing the nodes and a pointer(next) which points to the next child of the same parent. It is used to link the children of a node. Finally we initialize the structure variable.

Node consists of an integer(childNum) representing the number of the node and a pointer(childHead) which points to its first child. It stores the child number and the first child of a node. Finally we initialize the

structure variable.

Next is the pseudocode of the functions of the algorithm.

```
void CountLeaves(level of the node, id of the node)
{
    childLink *cl = first child of the node    ;
    if(maxlevel<level)maxlevel=level;
    if（the node has no child) (leaves in this level )++;
    else
    while(cl is not null)
    {
        CountLeaves(level+1,cl->id);
        cl=next child;
    }
}
```

This function is able to count leave numbers in specific level with correct input of the level and an id of a node in this level.

```
int main(void)
{

    int n,m;
    int i,j,id;
    childLink *cl=0;
    scanf(nodes number and internal nodes number);
    while (number of nodes!=0)
    {
        node=(Node *)malloc(sizeof(Node)*(n+1));
        memset(node,0,sizeof(Node)*(n+1));
    memset(count,0,sizeof(int)*(n+1));
        maxlevel=0;
        cl=0;
        for(i=0 to the number of internal nodes)            {
            scanf(id of the node);
            scanf(child number of the node);
            for(j=0 to the number of children of the node)
            {
                cl = new childLink;
                scanf(id of the child);
                if(the node has a first child) cl->next=first child of the node;
                first child of the node= cl;
            }
        }
        if(the root has no child){leaves of the first level=1;maxlevel=1;}
            else
            {
                Leaves of the first level=0;
                CountLeaves(1,1);
            }
        for(i=1 to i<maxlevel;i++)
            printf(the leave number in this level);
        printf(the leave number in max level);
        scanf(nodes number and internal number );
    }
    return 0;    }
```

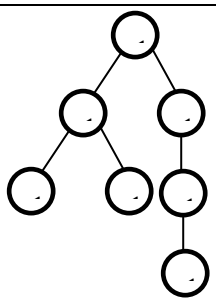
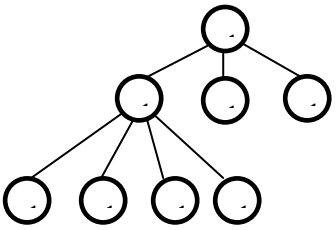
Chapter 3: Testing Results

Purpose: Read a tree from the standard input, then count the number of leaves and output them for every seniority level starting from the root.

Expected result: After the tree being legally input, print the number of leaves of each level from the root.

Possible errors: If the data of the tree is invalid (level>100), then the output may be wrong.

Table of test cases:

Input	Tree	Output
2 1 01 1 02		0 1
7 4 01 2 02 03 06 1 07 02 2 04 05 03 1 06		0 0 2 1
8 3 01 3 02 03 04 02 4 05 06 07 08		0 2 4
99 98 01 1 02 .... 98 1 99		0 0 0 0.....0 1
1 0		1

Chapter 4: Analysis and Comments

The time complexity of the core algorithm (Traverse all nodes and counting the number of leaves) is O(N).

We use link list to store the tree. So suppose the tree have *m* nodes, every node needs to store an int type data and a pointer. It will take more space than using array. But because the number of nodes is variable, using link list will be more convenient, and more space-saving when there are light nodes.

Appendix: Source Code (in C)

```
#include <stdio.h>
#include <memory.h>
#include <stdlib.h>

int count[100];       //the numbers of leaf nodes for each level
int maxlevel=1;       //max level of the tree

struct childLink    // used to link children of a node
{
      int id;
      childLink *next    ;    //point to the next child
      childLink(){next=0;}
};

struct Node           //the nodes of the tree
{
      int childNum;                 //number of the child
      childLink *childHead;      //point to the firsr child
      Node(){childNum=0;childHead=0;}
}*node;

void CountLeaves(int level,int id)              //count leaves in the level input with an id in this
                                                  level
 {
      childLink *cl = node[id].childHead;         //strat form the first child
      if(maxlevel<level)maxlevel=level;
      if(node[id].childNum==0)count[level]++;
      else
```

```
        while(cl)
        {
            CountLeaves(level+1,cl->id);    //recursively traverse the tree
            cl=cl->next;
        }
}

int main(void)
{

        int n,m;
        int i,j,id;
        childLink *cl=0;
        scanf("%d%d\n",&n,&m);
        while (n!=0||m!=0)
        {
            node=(Node *)malloc(sizeof(Node)*(n+1));
            memset(node,0,sizeof(Node)*(n+1));                  //Initialization
            memset(count,0,sizeof(int)*(n+1));
            maxlevel=0;
            cl=0;
            for(i=0;i<m;i++)                                            //read and save the tree
            {
                scanf("%d",&id);
                scanf("%d",&node[id].childNum);
                for(j=0;j<node[id].childNum;j++)
                {
                    cl = new childLink;
                    scanf("%d",&cl->id);
                    if(node[id].childHead) cl->next=node[id].childHead;
```

```
                node[id].childHead = cl;
            }
        }
        if(node[1].childNum==0){count[1]=1;maxlevel=1;}        //Determine the special circumstances
            else
            {
                count[1]=0;
                CountLeaves(1,1);
            }
        for(i=1;i<maxlevel;i++)                                //output
            printf("%d ",count[i]);
        printf("%d\n",count[i]);
        scanf("%d%d",&n,&m);
    }
    return 0;
}
```

Declaration

We hereby declare that all the work done in this project titled "Counting leaves" is of our independent effort as a group.

Duty Assignments:

Programmer: 苏航 18868106259
Tester: 吕锴燮 18658500407
Report Writer: 胡亮泽 18868103152