**姓名: 张景耀 学号: 3200602123 班级: 计算机2005**

### 实验内容1. 二叉树采用二叉链表存储，编写计算二叉树最大宽度的算法(二叉树的最大宽度是指二叉树所有层中结点个数的最大值)：

#### 源代码

#include<bits/stdc++.h>

using namespace std;

template <class T>

struct Node

{

    T val;

    Node\* left, \*right;

};

template <class T>

class BiTree

{

public:

    Node<T> \*root;

public:

    BiTree(T val)

    {

        this->root = new Node<T>({val, NULL, NULL});

    }

    void bfs()

    {

        queue<Node<T>\*> q;

        q.push(root);

        Node<T>\* curLast = root, \*nextLast = NULL;

        cout << "Breath First Search Result:\n";

        while(q.size())

        {

            Node<T>\* t = q.front();

            q.pop();

            cout << t->val << " ";

            if(t->left)

                q.push(t->left), nextLast = t->left;

            if(t->right)

                q.push(t->right), nextLast = t->right;

            if(t == curLast)

                puts(""), curLast = nextLast;

        }

        puts("");

    }

    int maxWidth()

    {

        auto ans = INT\_MIN, temp = 0;

        queue<Node<T>\*> q;

        q.push(root);

        Node<T>\* curLast = root, \*nextLast = NULL;

        while(q.size())

        {

            temp++;

            auto t = q.front();

            q.pop();

            if(t->left)

                nextLast = t->left, q.push(nextLast);

            if(t->right)

                nextLast = t->right, q.push(nextLast);

            if(t == curLast)

            {

                ans = max(ans, temp);

                temp = 0;

                curLast = nextLast;

            }

        }

        return ans;

    }

};

int main(int argc, char const \*argv[])

{

    BiTree<int> tree(1);

    tree.root->left = new Node<int>({2,

    new Node<int>({4, NULL, NULL}), NULL});

    tree.root->right = new Node<int>({3,

    new Node<int>({6, NULL, NULL}), new Node<int>({7,

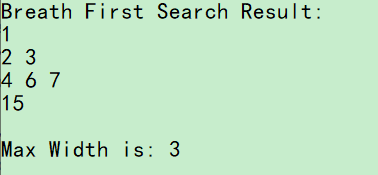
    new Node<int>({15, NULL, NULL}), NULL})});

    tree.bfs();

    cout << "Max Width is: " << tree.maxWidth() << endl;

}

#### 测试结果



### 实验内容2. 以二叉链表作存储结构，设计算法求二叉树中叶子结点的数目：

#### 源代码

#include<bits/stdc++.h>

using namespace std;

template <class T>

struct Node

{

    T val;

    Node\* left, \*right;

};

template <class T>

class BiTree

{

public:

    Node<T> \*root;

public:

    BiTree(T val)

    {

        this->root = new Node<T>({val, NULL, NULL});

    }

    void bfs()

    {

        queue<Node<T>\*> q;

        q.push(root);

        Node<T>\* curLast = root, \*nextLast = NULL;

        cout << "Breath First Search Result:\n";

        while(q.size())

        {

            Node<T>\* t = q.front();

            q.pop();

            cout << t->val << " ";

            if(t->left)

                q.push(t->left), nextLast = t->left;

            if(t->right)

                q.push(t->right), nextLast = t->right;

            if(t == curLast)

                puts(""), curLast = nextLast;

        }

        puts("");

    }

    int leafNum()

    {

        auto ans = 0;

        queue<Node<T>\*> q;

        q.push(root);

        while(q.size())

        {

            auto t = q.front();

            q.pop();

            if(!t->left and !t->right)

            {

                ans++;

                continue;

            }

            if(t->left)

                q.push(t->left);

            if(t->right)

                q.push(t->right);

        }

        return ans;

    }

};

int main(int argc, char const \*argv[])

{

    BiTree<int> tree(1);

    tree.root->left = new Node<int>({2,

    new Node<int>({4, NULL, NULL}), NULL});

    tree.root->right = new Node<int>({3,

    new Node<int>({6, NULL, NULL}), new Node<int>({7,

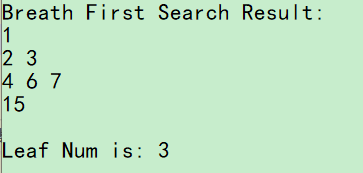
    new Node<int>({15, NULL, NULL}), NULL})});

    tree.bfs();

    cout << "Leaf Num is: " << tree.leafNum() << endl;

}

#### 测试结果



### 实验内容3. 树以孩子兄弟链表为存储结构，请设计算法求树的深度:

#### 源代码

#include<bits/stdc++.h>

using namespace std;

template<class T>

struct Node

{

    Node\* child, \*bro;

    T val;

};

template<class T>

class Tree

{

public:

    Node<T>\* root;

public:

    Tree(T val)

    {

        this->root = new Node<T>({NULL, NULL, val});

    }

    int maxDepth()

    {

        return dfs(this->root);

    }

    void bfs()

    {

        queue<Node<T>\*> q;

        q.push(this->root);

        cout << "Breath First Search Result:\n";

        Node<T>\* curLast = root, \*nextLast = NULL;

        while(q.size())

        {

            auto t = q.front();

            q.pop();

            printf("%d ", t->val);

            auto p = t->child;

            while(p)

            {

                q.push(p);

                nextLast = p;

                p = p->bro;

            }

            if(t == curLast)

            {

                puts("");

                curLast = nextLast;

            }

        }

        puts("");

    }

    int dfs(Node<T>\* node)

    {

        if(!node)

            return 0;

        auto t = node->child;

        auto ans = 0;

        while(t)

        {

            ans = max(ans, dfs(t));

            t = t->bro;

        }

        return ans + 1;

    }

};

int main(int argc, char const \*argv[])

{

    Tree<int> tree(1);

    tree.root->child = new Node<int>({NULL, NULL, 2});

    tree.root->child->bro = new Node<int>({NULL, NULL, 3});

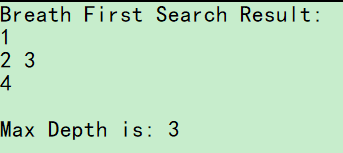
    tree.root->child->child = new Node<int>({NULL, NULL, 4});

    tree.bfs();

    cout << "Max Depth is: " << tree.maxDepth() << endl;

}

#### 测试结果



### 实验内容4. 以孩子—兄弟表示法作为树的存储结构，请设计算法求树的度:

#### 源代码

#include<bits/stdc++.h>

using namespace std;

template<class T>

struct Node

{

    Node\* child, \*bro;

    T val;

};

template<class T>

class Tree

{

public:

    Node<T>\* root;

public:

    Tree(T val)

    {

        this->root = new Node<T>({NULL, NULL, val});

    }

    int degree()

    {

        int ans = INT\_MIN;

        dfs(this->root, ans);

        return ans;

    }

    void bfs()

    {

        queue<Node<T>\*> q;

        q.push(this->root);

        cout << "Breath First Search Result:\n";

        Node<T>\* curLast = root, \*nextLast = NULL;

        while(q.size())

        {

            auto t = q.front();

            q.pop();

            printf("%d ", t->val);

            auto p = t->child;

            while(p)

            {

                q.push(p);

                nextLast = p;

                p = p->bro;

            }

            if(t == curLast)

            {

                puts("");

                curLast = nextLast;

            }

        }

        puts("");

    }

    void dfs(Node<T>\* node, int& ans)

    {

        auto cnt = 0;

        auto t = node->child;

        while(t)

        {

            cnt++;

            dfs(t, ans);

            t = t->bro;

        }

        ans = max(ans, cnt);

    }

};

int main(int argc, char const \*argv[])

{

    Tree<int> tree(1);

    tree.root->child = new Node<int>({NULL, NULL, 2});

    tree.root->child->bro = new Node<int>({NULL, new Node<int>({NULL, NULL, 3}), 4});

    tree.root->child->child = new Node<int>({NULL, NULL, 5});

    tree.bfs();

    printf("Max Degree is: %d", tree.degree());

    return 0;

}

#### 测试结果

