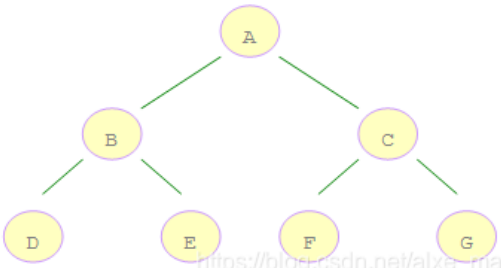
C++实现二叉树深度优先搜索（DFS）、广度优先搜索（BFS）

原创 alxe_made 于 2019-07-06 20:18:21 发布 阅读量7.3k 收藏 30 点赞数 5 版权

分类专栏： C++ 文章标签： 二叉树的深度优先搜索DFS 二叉树的广度优先搜索BFS

> 在前面的文章中我们实现了 二叉树 的深度优先搜索， 先序遍历 ，中序遍历，后序遍历，分别实现了递归和非递归版本。这里我们着重介绍一下广度优先搜索（BFS）

1. 基本概念



如图所示，BFS为是从根节点开始，沿着树的宽度遍历树的节点。如果所有节点均被访问，则算法中止。如上图所示的二叉树，A 是第一个访问的，然后顺序是 B、C，然后再是 D、E、F、G。

2. 基本思路

仔细看看层序遍历过程，其实就是从上到下，从左到右依次将每个数放入到 队列 中，然后按顺序依次打印就是想要的结果。所以这里我们借助一个队列进行操作。

实现过程

1. 首先将二叉树的根节点push到队列中，判断队列不为nullptr，就输出队头的元素，
2. 判断节点如果有孩子，就将孩子push到队列中，
3. 遍历过的节点出队列，
4. 循环以上操作，直到Node == nullptr。

3. 代码

```
#include <iostream>
#include <queue>

struct Node {
    int value;
    Node* left;
    Node* right;
    Node(int value):
        value(value), left(nullptr), right(nullptr) {}
};

void dfs(Node* head) {
    if (head == nullptr) {
        return;
    }
    std::cout << head->value << ",";
    dfs(head->left);
    dfs(head->right);
}

void bfs(Node* head) {
```

```
    if (head == nullptr) {  // if head is nullptr, return directly
        return;
    }
    std::queue<Node*> qbfs;
    qbfs.push(head);
    while (!qbfs.empty()) {
        std::cout << qbfs.front()->value << ",";
        if (qbfs.front()->left != nullptr) {    // if current queue front has left child
            qbfs.push(qbfs.front()->left);
        }
        if (qbfs.front()->right != nullptr) {   // if current queue front has right child
            qbfs.push(qbfs.front()->right);
        }
        qbfs.pop();  // pop if we have alreay visit the node
    }
}
int main() {
    Node* head = new Node(1);
    head->left = new Node(2);
    head->right = new Node(3);
    head->left->left = new Node(4);
    head->left->right = new Node(5);
    head->right->left = new Node(6);
    head->right->right = new Node(7);

    std::cout << "==============DFS==============\n";
    dfs(head);
    std::cout << "\n==============BFS==============\n";
    bfs(head);
    return 0;
}
```