**广度优先搜索**

**算法思想：**

宽度优先搜索算法（又称广度优先搜索）是最简便的图的搜索算法之一，这一算法也是很多重要的图的算法的原型。Dijkstra单源最短路径算法和Prim最小生成树算法都采用了和宽度优先搜索类似的思想。其别名又叫BFS，属于一种盲目搜寻法，目的是系统地展开并检查图中的所有节点，以找寻结果。换句话说，它并不考虑结果的可能位置，彻底地搜索整张图，直到找到结果为止。  
广度优先搜索使用队列（queue）来实现，整个过程也可以看做一个倒立的树形：  
1、把根节点放到队列的末尾。

1. 每次从队列的头部取出一个元素，查看这个元素所有的下一级元素，把它们放到队列的末尾。并把这个元素记为它下一级元素的前驱。

3、找到所要找的元素时结束程序。

4、如果遍历整个树还没有找到，结束程序。  
  
**算法实现：**

#include<iostream>

#include<vector>

#include<queue>

#include<memory.h>

using namespace std;

vector<vector<int> > tree;//声明一个二维向量

int flag[10];//用于搜索到了节点i的第几个节点

queue<int> M;//声明一个队列

int ar\_tree[8] = { 1,1,1,3,5,3,5,7 };

void BFS(int node) {

int temp;

cout << node << " "; //从队列中取出第一个节点

int m\_first = M.front();

1. pop();

while(flag[node] < tree[node].size()) {

temp = tree[node][flag[node]];

flag[node]++;//把temp加入队列中

M.push(temp);

}

if (!M.empty()) {

BFS(M.front());

}

}

int main() {

ios::sync\_with\_stdio(false);

memset(flag, 0, sizeof(flag));

int i,temp;

tree.resize(10);//图中的数共有九个节点

//生成树

for (i = 2; i <=9; i++) {

temp = ar\_tree[i - 2];

tree[temp].push\_back(i);//表示第i个节点为第temp个节点的子节点

} //BFS

cout << "BFS过程：" << endl;

1. push(1);

BFS(1);

cout << endl;

return 0;

}

**运行结果：**

