完全二叉树判定方法

把完全二叉树以数组的形式存储起来，只要结点是连续存放，就可以说明这棵树是完全二叉树。

也就是说，只要最后一个结点的坐标与结点的总数相同，说明这棵树是完全二叉树。

下面给出一道例题：

甲级PAT：1110 Complete Binary Tree (25分)

1110 Complete Binary Tree (25分)

Given a tree, you are supposed to tell if it is a complete binary tree.

Input Specification:

Each input file contains one test case. For each case, the first line gives a positive integer *N* (≤20) which is the total number of nodes in the tree -- and hence the nodes are numbered from 0 to *N*−1. Then *N* lines follow, each corresponds to a node, and gives the indices of the left and right children of the node. If the child does not exist, a `-` will be put at the position. Any pair of children are separated by a space.

Output Specification:

For each case, print in one line `YES` and the index of the last node if the tree is a complete binary tree, or `NO` and the index of the root if not. There must be exactly one space separating the word and the number.

Sample Input 1:

```
9
7 8
- -
- -
- -
0 1
2 3
4 5
- -
- -
```

Sample Output 1:

```
YES 8
```

Sample Input 2:

```
8
- -
4 5
0 6
- -
2 3
- 7
- -
- -
```

Sample Output 2:

```
NO 1
```

思路：

1. 将结点存起来
2. 数据中不存在点的序号就是根节点
3. DFS，找到最后一个结点的下标。
4. 如果下标与结点个数相同，说明是完全二叉树
5. 在录入数据时，请用string，别用char。数据有可能是两位数。
6. stoi指的是string to int。

代码参考：

```
#include <iostream>
#include <string>
using namespace std;
const int noNode = -1;
struct node
{
    int left, right;
}stor[100];
bool hashTable[100];
int maxn = -1, head = 0, ans = noNode, number;
void init()
{
    scanf("%d\n", &number);
    for(int i=0;i<number;i++)
    {
        string temp1, temp2;
        cin >> temp1 >> temp2;
        if (temp1 != "-") stor[i].left = stoi(temp1), hashTable[stor[i].left] =
true;
        else stor[i].left = noNode;
        if (temp2 != "-") stor[i].right = stoi(temp2), hashTable[stor[i].right]
= true;
        else stor[i].right = noNode;
    }
```

```cpp
}
void DFS(int root,int index)
{
    if (index > maxn) {
        maxn = index;
        ans = root;
    }
    if (stor[root].left != noNode)  DFS(stor[root].left, index * 2);
    if (stor[root].right != noNode) DFS(stor[root].right, index * 2 + 1);
}
int main()
{
    init();
    while (hashTable[head] == true) head++;
    DFS(head, 1);
    if (maxn == number) cout << "YES " << ans ;
    else cout << "NO " << head ;
    return 0;
}
```