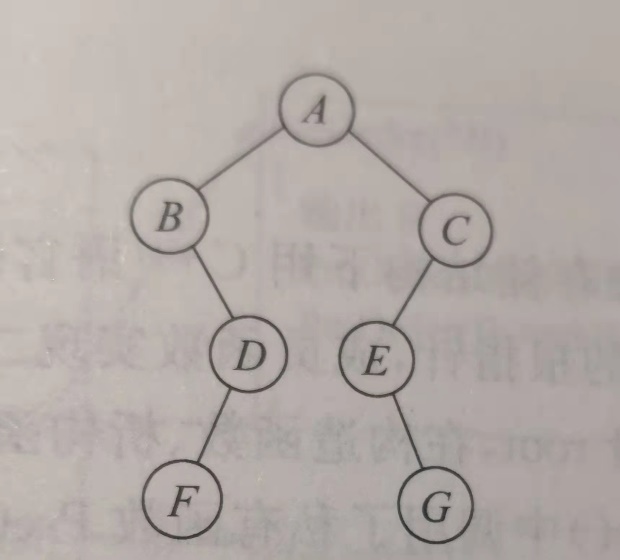
**二叉树顺序结构下的前序遍历**

【问题描述】一棵具有n个结点的二叉树采用顺序存储结构，设计算法对该二叉进行前序遍历。并进行测试

【输入形式】二叉树的顺序存储序列  
【输出形式】二叉树的前序遍历

【样例输入】ABC DE   F  G

【样例输出】ABDFCEG



该二叉树的顺序存储序列为ABC\*DE\*\*\*F\*\*G，此处\*表示空格。

void preorder(const char a[], int n, int i) {

if (i >= n)

return;

if (a[i] == ' ')

return;

cout << a[i];

preorder(a, n, 2 \* i + 1);

preorder(a, n, 2 \* i + 2);

}

int main(){

string s;

char a[30];

getline(cin, s);

preorder(s.c\_str(), s.length(), 0);

system("pause");

return 0;

}

**最大二叉树**

【问题描述】设计算法，通过一个数组a创建一棵最大二叉树的二叉链表并进行测试。

最大二叉树是每棵子二叉树的根结点值都大于其左右子树上所有结点的值。

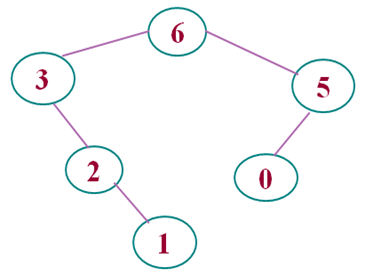
给定一个不含重复元素的整数数组。一个以此数组构建的最大二叉树定义如下：

二叉树的根是数组中的最大元素。

左子树是通过数组中最大值左边部分构造出的最大二叉树。

右子树是通过数组中最大值右边部分构造出的最大二叉树。

如数组a的内容依次为{3,2,1,6,0,5} ，则创建出如图所示的最大二叉树。



输入形式】数组a的长度和内容  
【输出形式】该二叉树的前序序列  
【样例输入】

6

3 2 1 6 0 5

【样例输出】

6 3 2 1 5 0

template <class DataType>

BiNode<DataType>\* BiTree<DataType>::creat3(DataType a[], int start, int end) {

if (start <= end) {

int max = start;

for (int j = start + 1; j <= end; j++)

if (a[j] > a[max])

max = j;

BiNode<DataType>\* root = new BiNode<DataType>(a[max]);

root->lchild = creat3(a, start, max - 1);

root->rchild = creat3(a, max + 1, end);

return root;

}

else

return NULL;

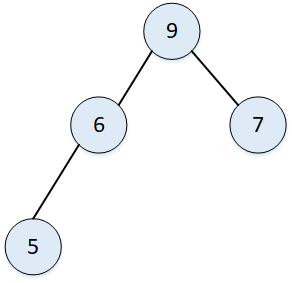
}

root = creat3(a, 0, n - 1);

**3.是否存在长度等于给定值sum的路径**

【问题描述】设计算法，判断结点值为正整数的二叉树中是否存在一条根出发的路径，该路径上的结点值之和等于给定的值sum。请对算法进行测试。

如通过输入带特殊值0的前序序列创建二叉树。当输入9 6 5 0 0 0 7 0 0，创建二叉树结构如下：



在该二叉树下，如sum为9,15,20,16等，则结论成立，如sum为14,17等，则结论不成立。

【输入形式】

第一行：带特殊值的前序序列

第二行：sum的值  
【输出形式】1或0

【样例输入】

9 6 5 0 0 0 7 0 0

15

类的接口方法参考如下：

template <class DataType>

bool BiTree<DataType>::path\_sum(int sum) {

        return recursive\_path\_sum(root, sum);

 }

递归函数参考如下：

template <class DataType>

bool BiTree<DataType>::recursive\_path\_sum(BiNode<DataType>\* sub\_root, int sum);

template <class DataType>

BiNode<DataType>\* BiTree<DataType> :: CreatFromPre()

{

int val;

BiNode<DataType>\* bt = nullptr;

cin >> val;

if (val == 0) bt = nullptr;

else {

bt = new BiNode<DataType>(val);

bt->lchild = Creat();

bt->rchild = Creat();

}

return bt;

}

template <class DataType>

bool BiTree<DataType>::recursive\_path\_sum(BiNode<DataType>\* sub\_root, int sum) {

if (sub\_root == nullptr) {

return (sum == 0);

}

if (sub\_root->data == sum) {

return true;

}

int remain\_sum = sum - sub\_root->data;

return recursive\_path\_sum(sub\_root->lchild, remain\_sum) || recursive\_path\_sum(sub\_root->rchild, remain\_sum);

}

int main(){

BiTree<int> T;

int sum;

T.CreatFromPre();

cin >> sum;

cout<<T.path\_sum(sum);

system("pause");

return 0;

}

|  |  |
| --- | --- |
| **1** | 一棵完全二叉树上有1001 个结点，其中叶子结点的个数是 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 。  A． 250  B． 501  C． 254  D． 505  **正确答案:**b |

n=n2+n1+n0

n0=n2+1

n=2n2+1+n1

n1要么0 要么1

1001= 2n2+1 (n1=0) n2=500 n0=501

|  |  |
| --- | --- |
| **2** | 以二叉链表作为二叉树的存储结构，在有n个结点的二叉链表中(n>0)，链表中空链域的个数为 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  A. 2n - 1  B. n - 1  C. n + 1  D. 2n + 1  **正确答案:**C |

2n-(n-1) =n+1 (有n-1条边)

|  |  |
| --- | --- |
| **3** | 已知n个结点的二叉树具有最小路径长度时，其深度为k，那么第k层上的结点数为 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  A. n - 2k-2 + 1  B. n - 2k-1 + 1  C. n - 2k + 1  D. n - 2k-1  **正确答案:**B |

第1-第k-1层 共含2K-1 -1

所以第k层 n-(2K-1 -1)

|  |  |
| --- | --- |
| **4** | 已知一棵二叉树的前序序列为**B**ACDEGHF，中序序列为CADBHGEF，则其后序序列为 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  A. CAHGDFEB  B. CDABGHFE  C. CDAHGFEB  D. HGFEBCDA    **正确答案:**c  ACD| EGHF | B  CD A| GHF E |B |
| **5** | 一棵二叉树高度为h，所有结点的度或为0，或为2，则这棵二叉树最少有 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 个结点。  A．2h    ﻿B．2h-1    C．2h+1    ﻿D．h+1  正确答案: B |

举例

|

| |

2h-1