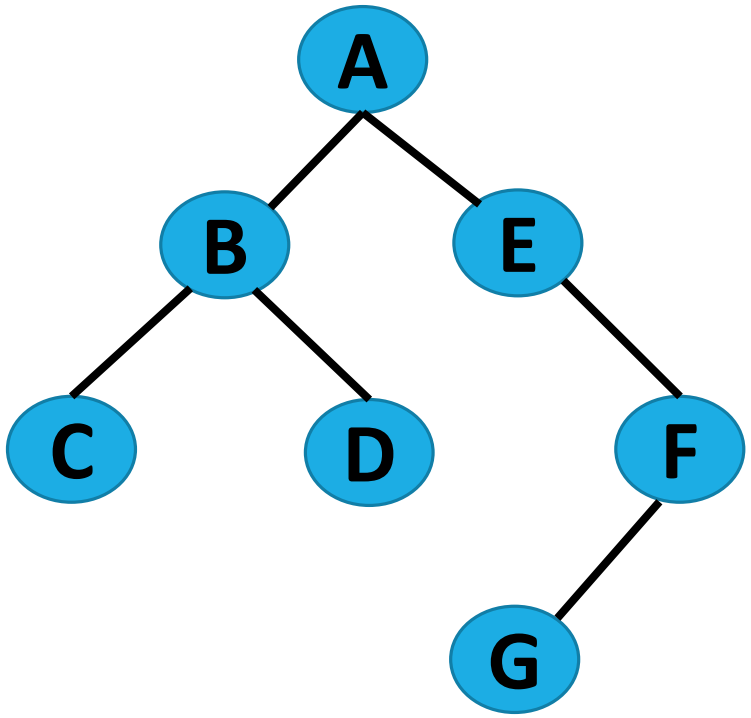
统计二叉树叶子结点个数

算法4-16

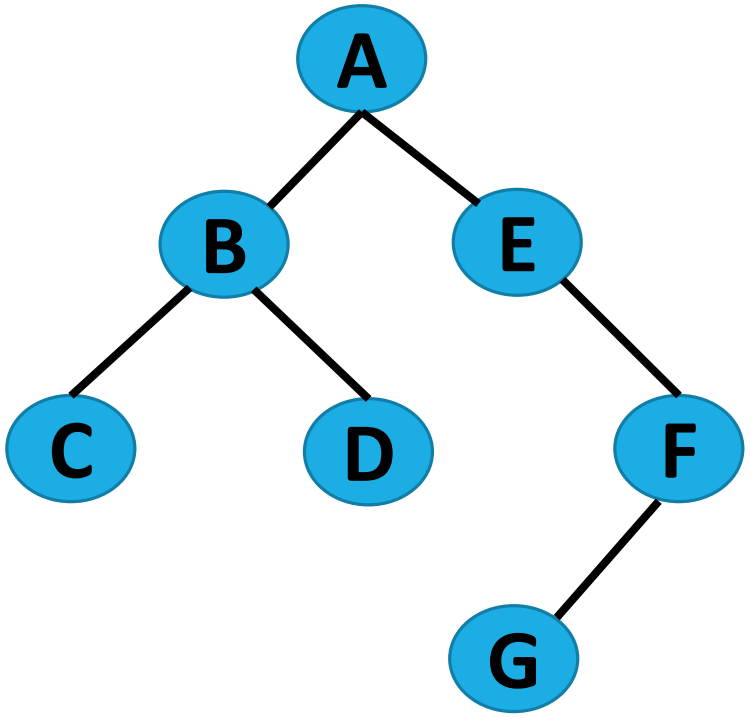
```
1   int CountLeafNode(BinTree bt)  //统计叶子结点数
2   {
3           if (bt==NULL)
4                   return 0;  //递归调用的结束条件
5           else  //左右子树都为空，是叶子
6            if((bt->leftchild==NULL)&&(bt->rightchild==NULL))
7                   return 1;
8            else  //递归遍历左子树和右子树
9               return(CountLeafNode(bt->leftchild)
10                                  +CountLeafNode(bt->rightchild));
11  }
```



统计二叉树的深度

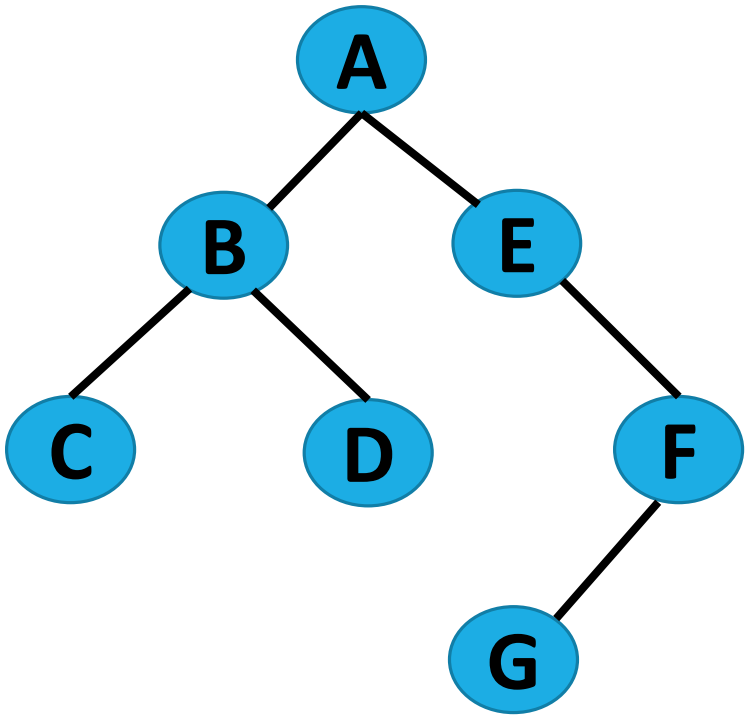
算法4-17

```
1   int CountLevel(BinTree bt)  //计算二叉树的深度
2   {
3     if (bt==NULL) return 0;        //如果空则返回0
4     else
5     {
6      int i=CountLevel(bt->leftchild);  //递归计算左子树的深度
7      int j=CountLevel(bt->rightchild);  //递归计算右子树的深度
8      return (i>j?i:j)+1;          //返回两个子树中高的深度+1
9     }
10  }
```



复制一棵二叉树　　算法4-18

```
1   BinTree Copy(BinTree original)   //复制一棵二叉树
2   {
3      BinTreeNode * temp;
4      if (original ==NULL) return NULL;    //如果空则返回NULL
5      else {
6        temp = (BinTreeNode *)malloc(sizeof(BinTreeNode));
7       if(!temp)  {
8         printf("out of space!");
9         exit(1);
10       }
11     temp->leftchild = Copy(original->leftchild);
12     temp->rightchild = Copy(original->rightchild);
13     temp->data = original->data;
14     return temp;
15   }
16  }
```



思考：查找某数据元素是否存在，如果存在返回位置