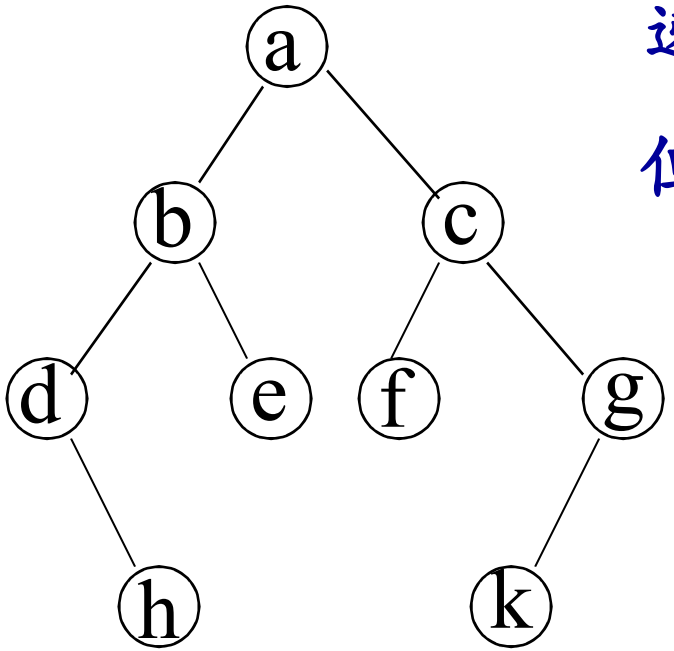
6.3二叉树遍历的非递归算法



递归算法书写简单，

但是效率低

先（根）序遍历算法：

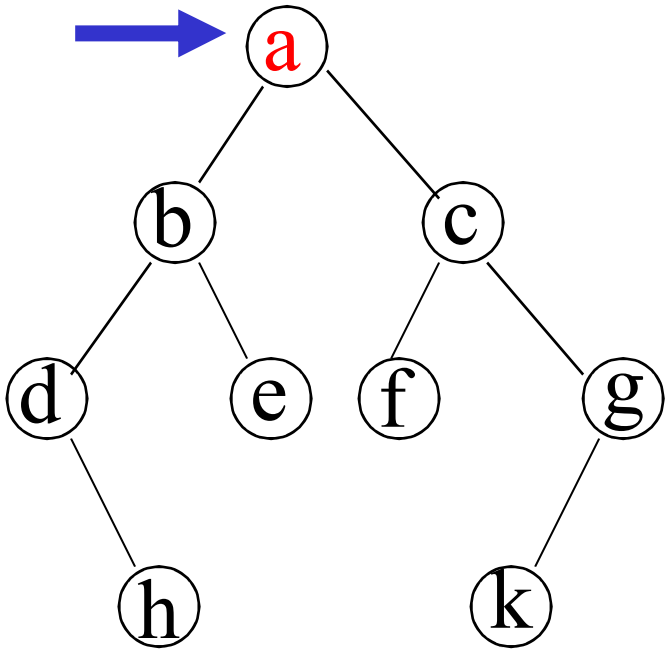
若二叉树为空树，则空操作；否则，

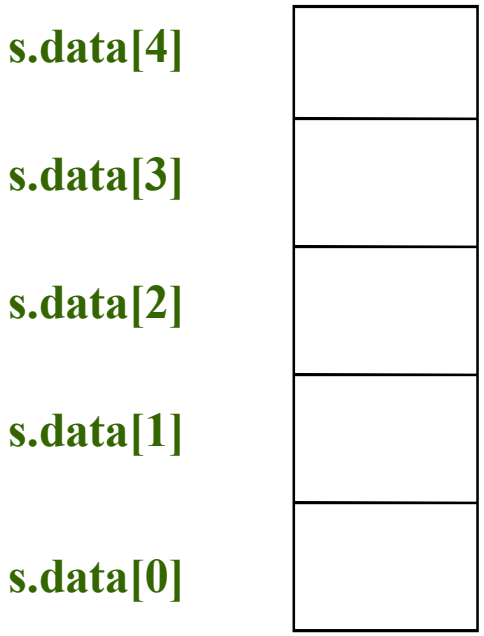
（1）访问根结点；

（2）先序遍历左子树；

（3）先序遍历右子树。

a



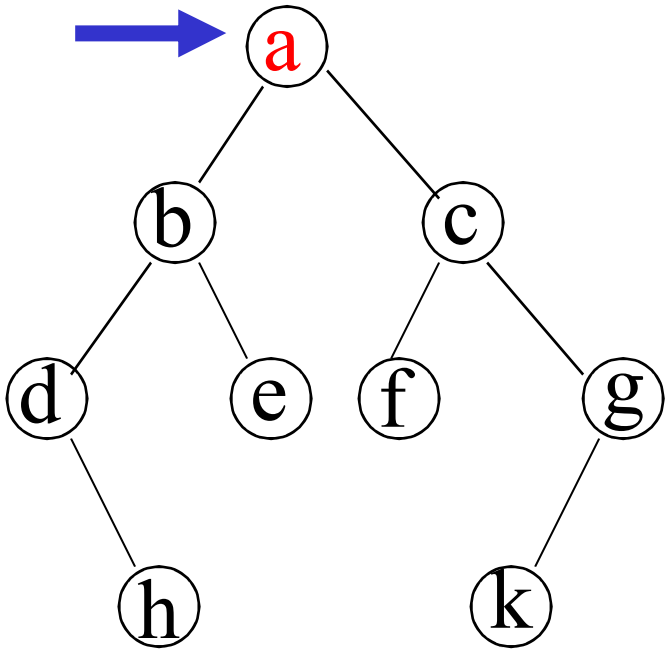


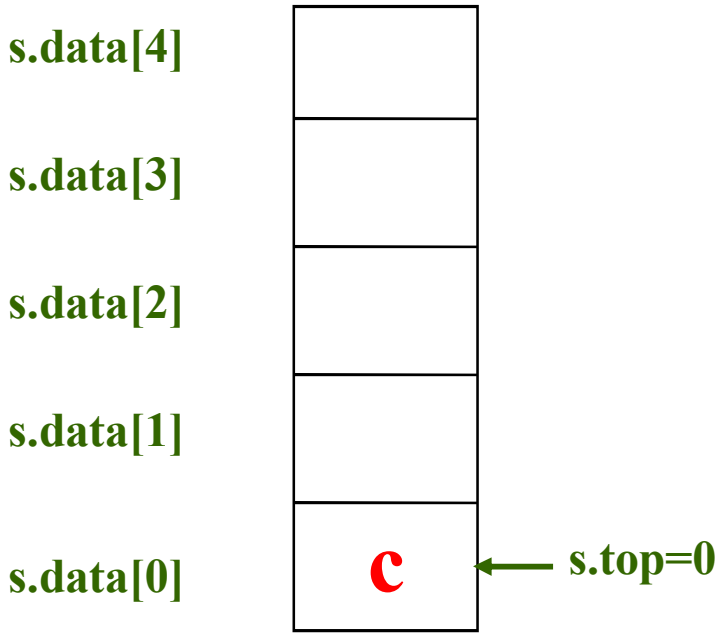
s.top=-1

先序遍历非递归算法的实现：访问根结点后，在访问左子树前，应保存其非空右子树

图中a的右子树先于b的右子树被保存，但是其访问要在b的右子树被访问后进行----先保存后访问----先进后出----借助栈实现

a

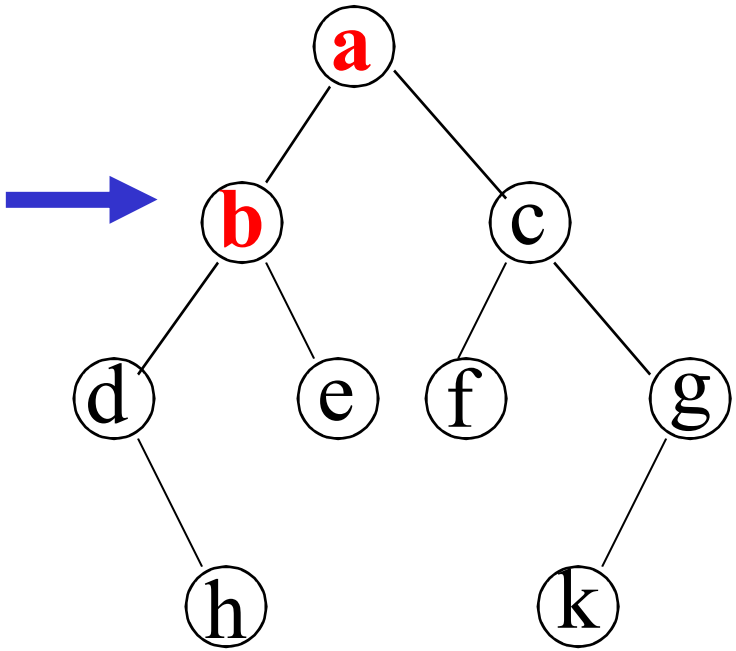


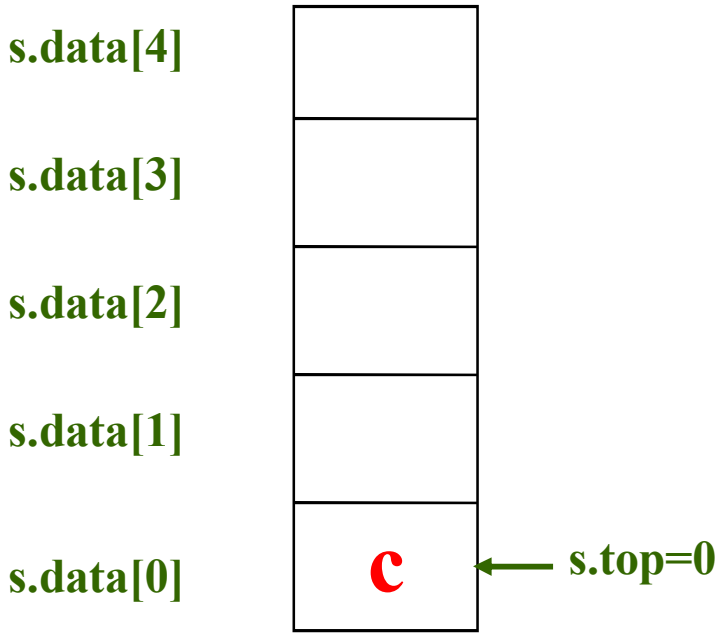


先序遍历非递归算法的实现：访问根结点后，在访问左子树前，应保存其非空右子树

图中a的右子树先于b的右子树被保存，但是其访问要在b的右子树被访问后进行----先保存后访问----先进后出----借助栈实现

ab

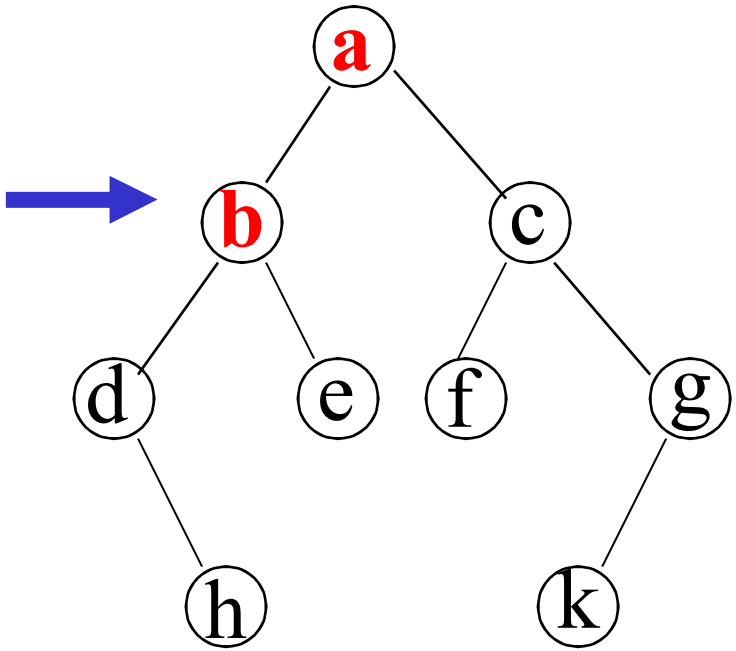


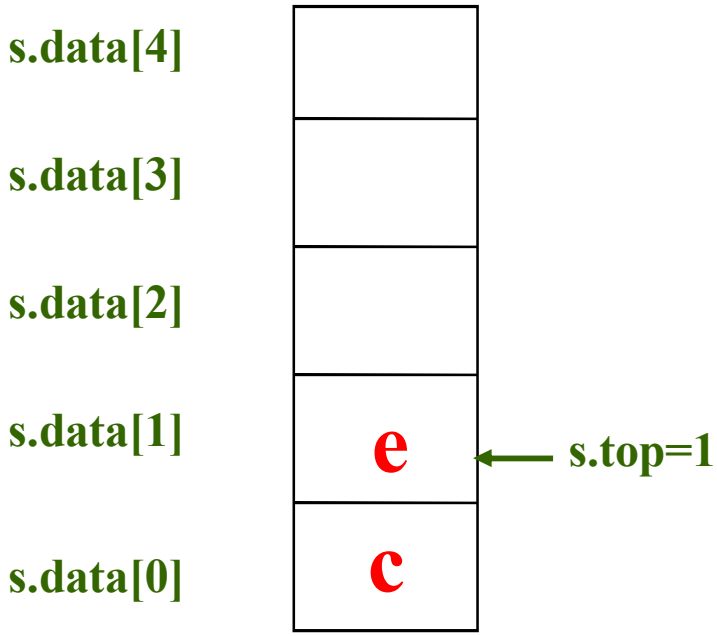


先序遍历非递归算法的实现：访问根结点后，在访问左子树前，应保存其非空右子树

图中a的右子树先于b的右子树被保存，但是其访问要在b的右子树被访问后进行----先保存后访问----先进后出----借助栈实现

ab

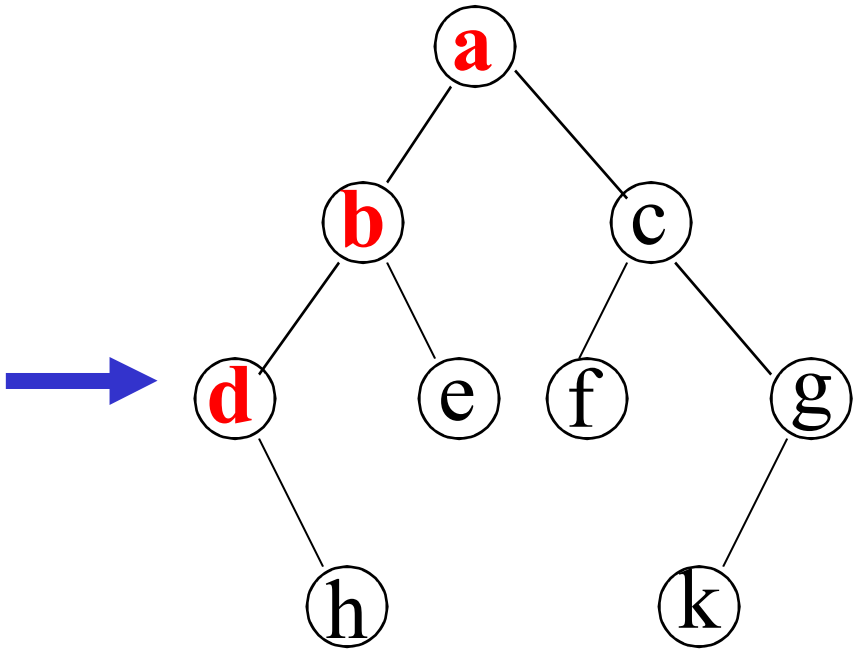


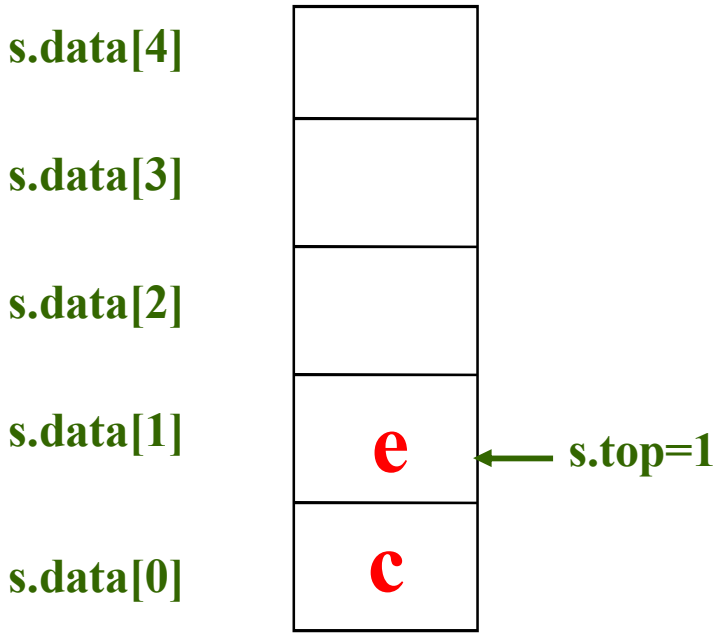


先序遍历非递归算法的实现：访问根结点后，在访问左子树前，应保存其非空右子树

图中a的右子树先于b的右子树被保存，但是其访问要在b的右子树被访问后进行----先保存后访问----先进后出----借助栈实现

abd

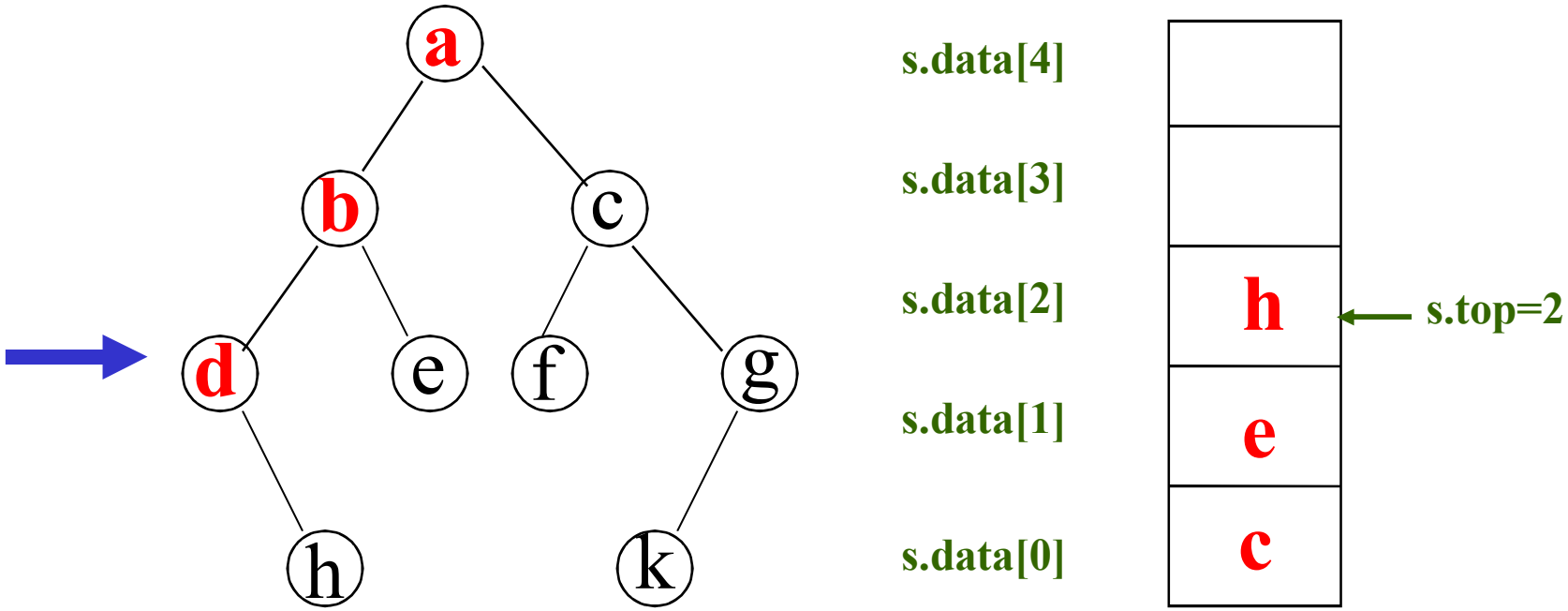




先序遍历非递归算法的实现：访问根结点后，在访问左子树前，应保存其非空右子树

图中a的右子树先于b的右子树被保存，但是其访问要在b的右子树被访问后进行----先保存后访问----先进后出----借助栈实现

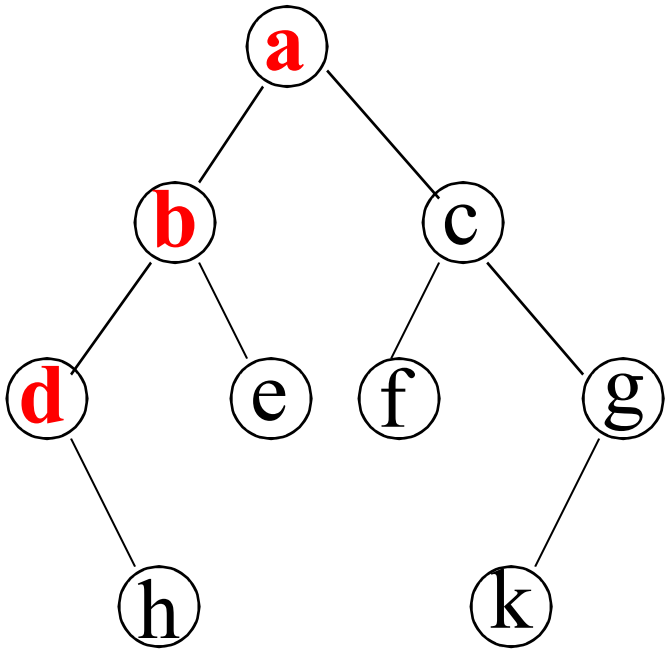
abd

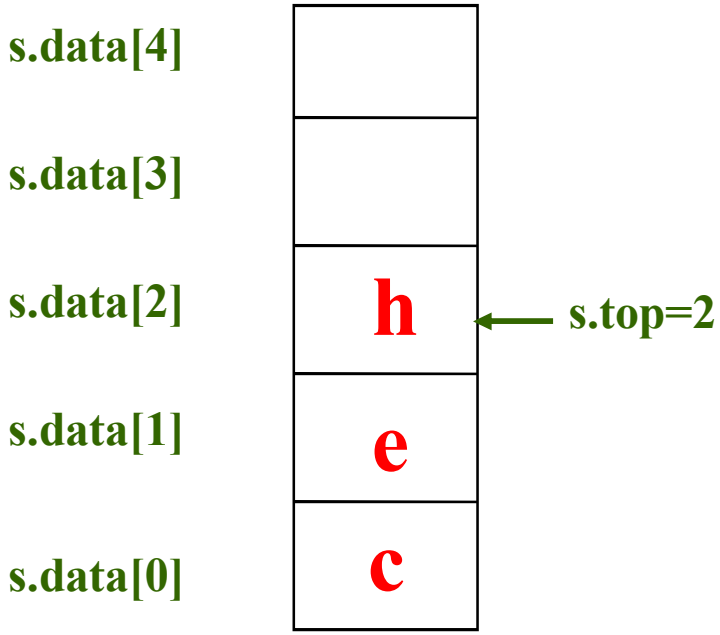


先序遍历非递归算法的实现：访问根结点后，在访问左子树前，应保存其非空右子树

图中a的右子树先于b的右子树被保存，但是其访问要在b的右子树被访问后进行----先保存后访问----先进后出----借助栈实现

abd

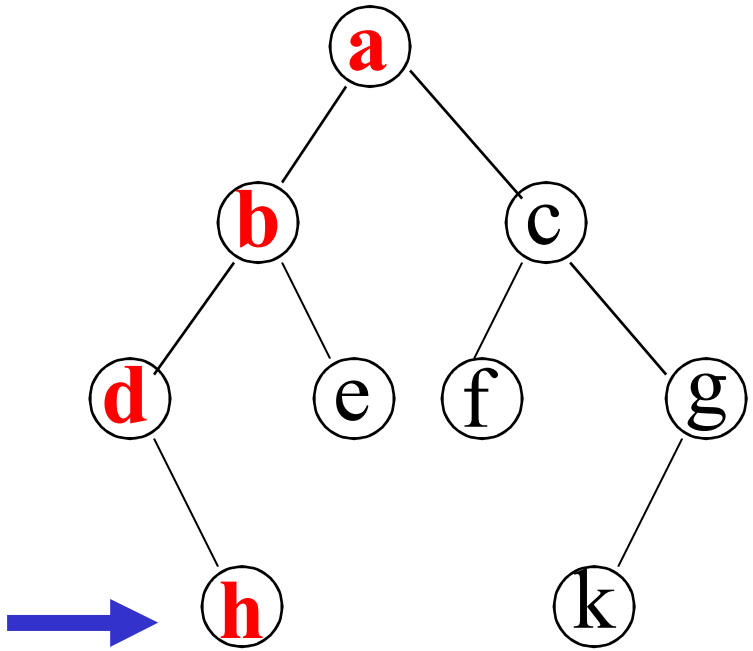


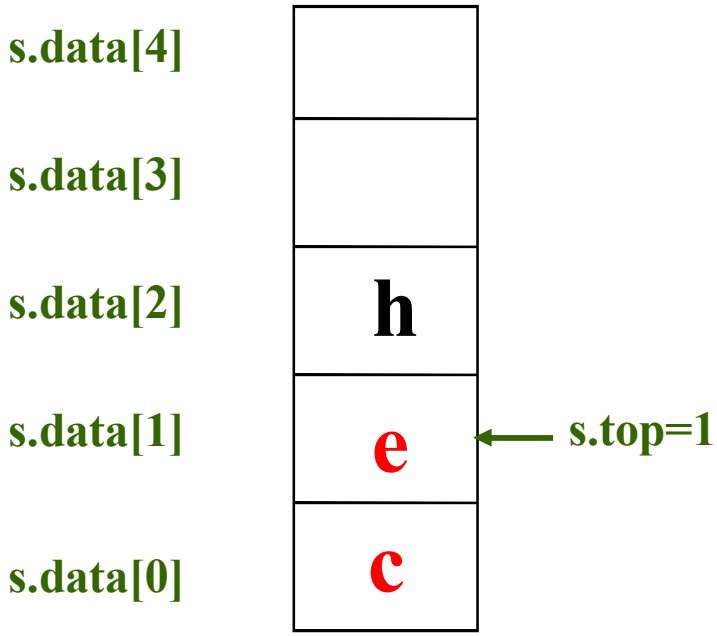


先序遍历非递归算法的实现：访问根结点后，在访问左子树前，应保存其非空右子树

图中a的右子树先于b的右子树被保存，但是其访问要在b的右子树被访问后进行----先保存后访问----先进后出----借助栈实现

abdh

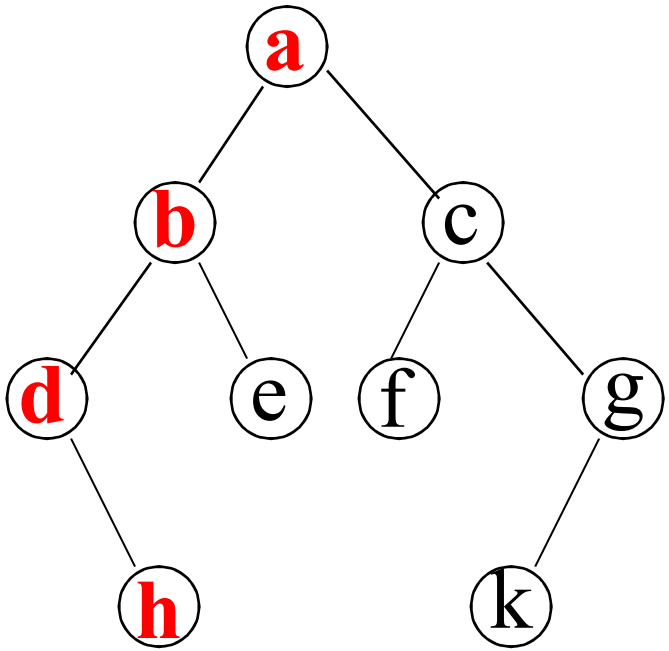


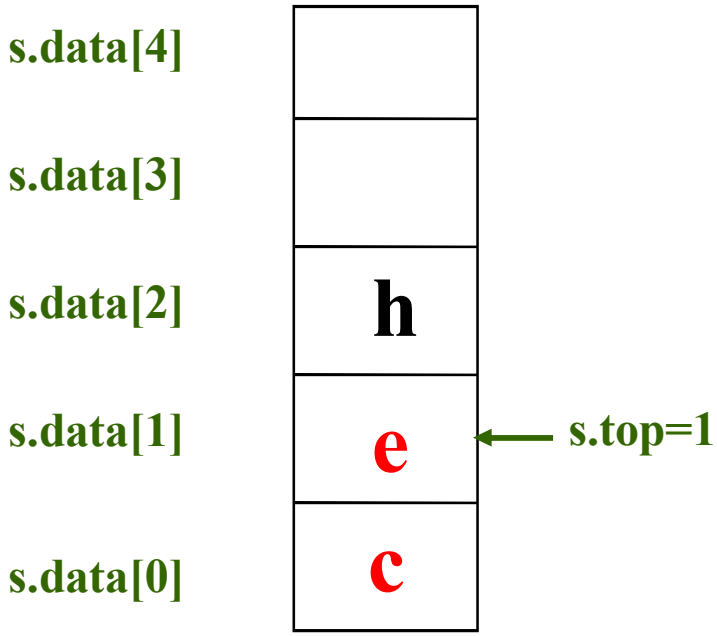


先序遍历非递归算法的实现：访问根结点后，在访问左子树前，应保存其非空右子树

图中a的右子树先于b的右子树被保存，但是其访问要在b的右子树被访问后进行----先保存后访问----先进后出----借助栈实现

abdh

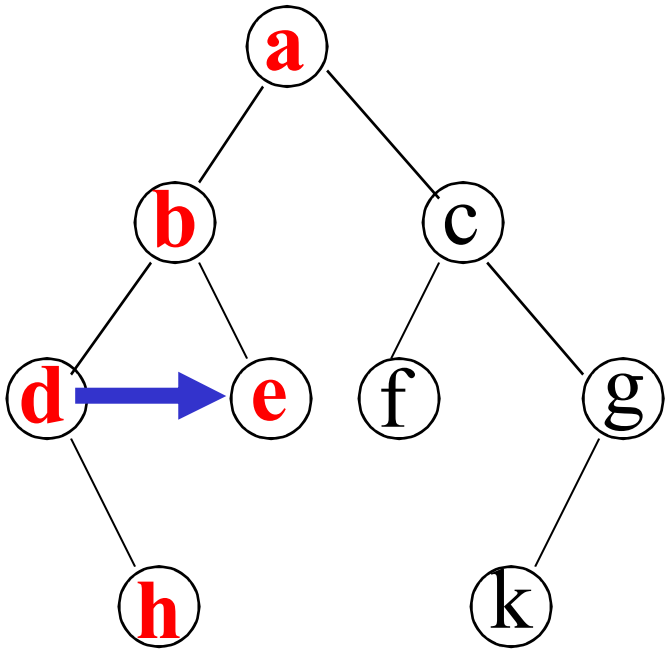


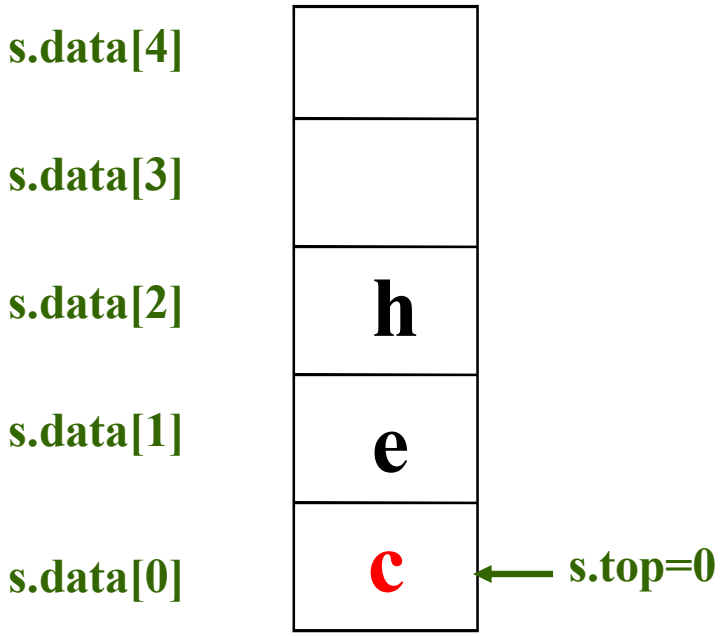


先序遍历非递归算法的实现：访问根结点后，在访问左子树前，应保存其非空右子树

图中a的右子树先于b的右子树被保存，但是其访问要在b的右子树被访问后进行----先保存后访问----先进后出----借助栈实现

abdhe

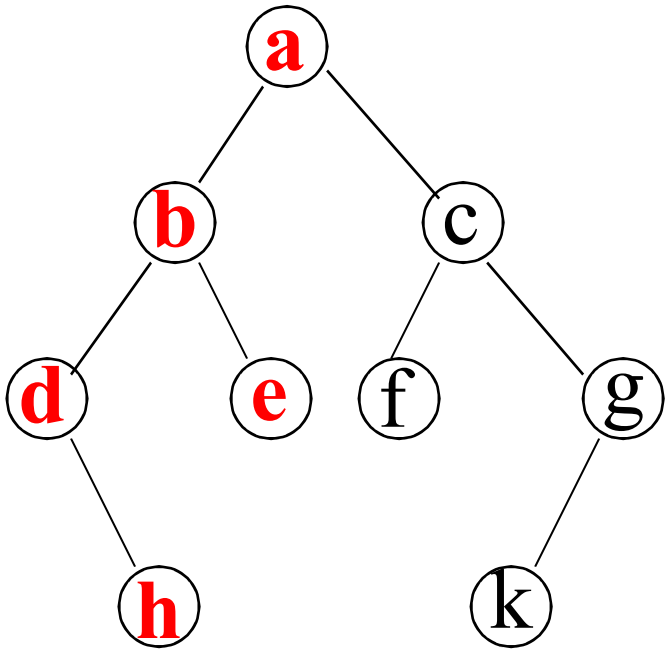


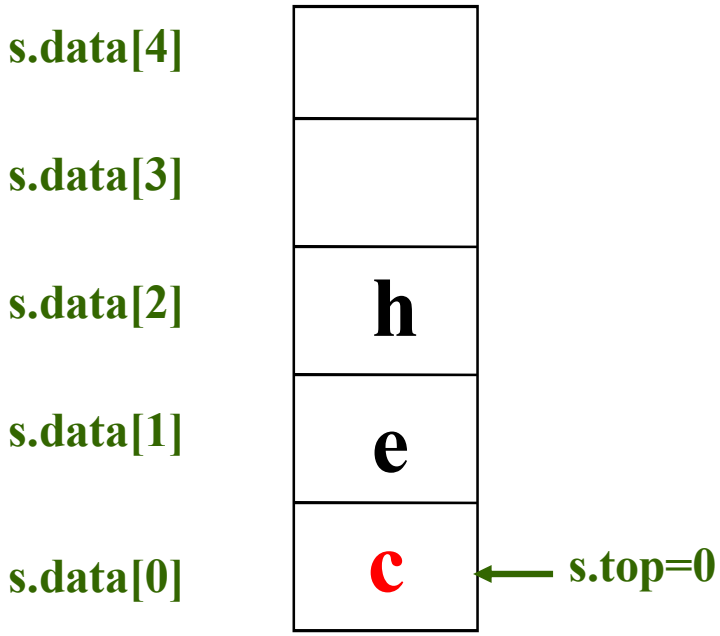


先序遍历非递归算法的实现：访问根结点后，在访问左子树前，应保存其非空右子树

图中a的右子树先于b的右子树被保存，但是其访问要在b的右子树被访问后进行----先保存后访问----先进后出----借助栈实现

abdhe

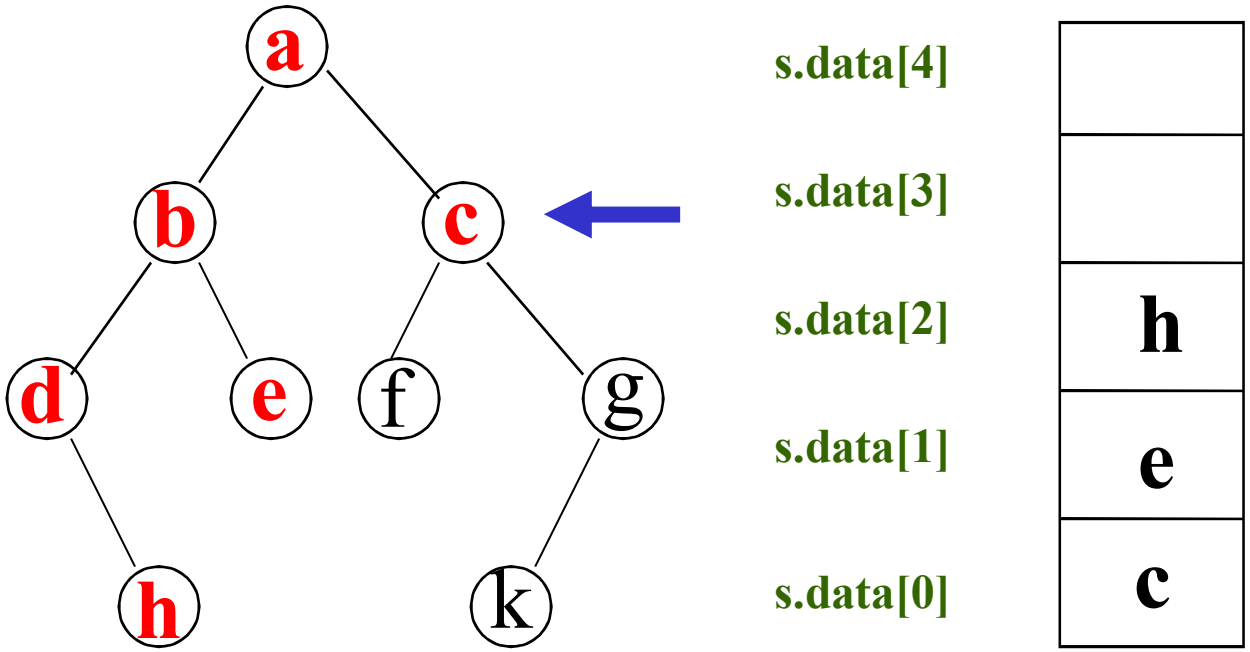




先序遍历非递归算法的实现：访问根结点后，在访问左子树前，应保存其非空右子树

图中a的右子树先于b的右子树被保存，但是其访问要在b的右子树被访问后进行----先保存后访问----先进后出----借助栈实现

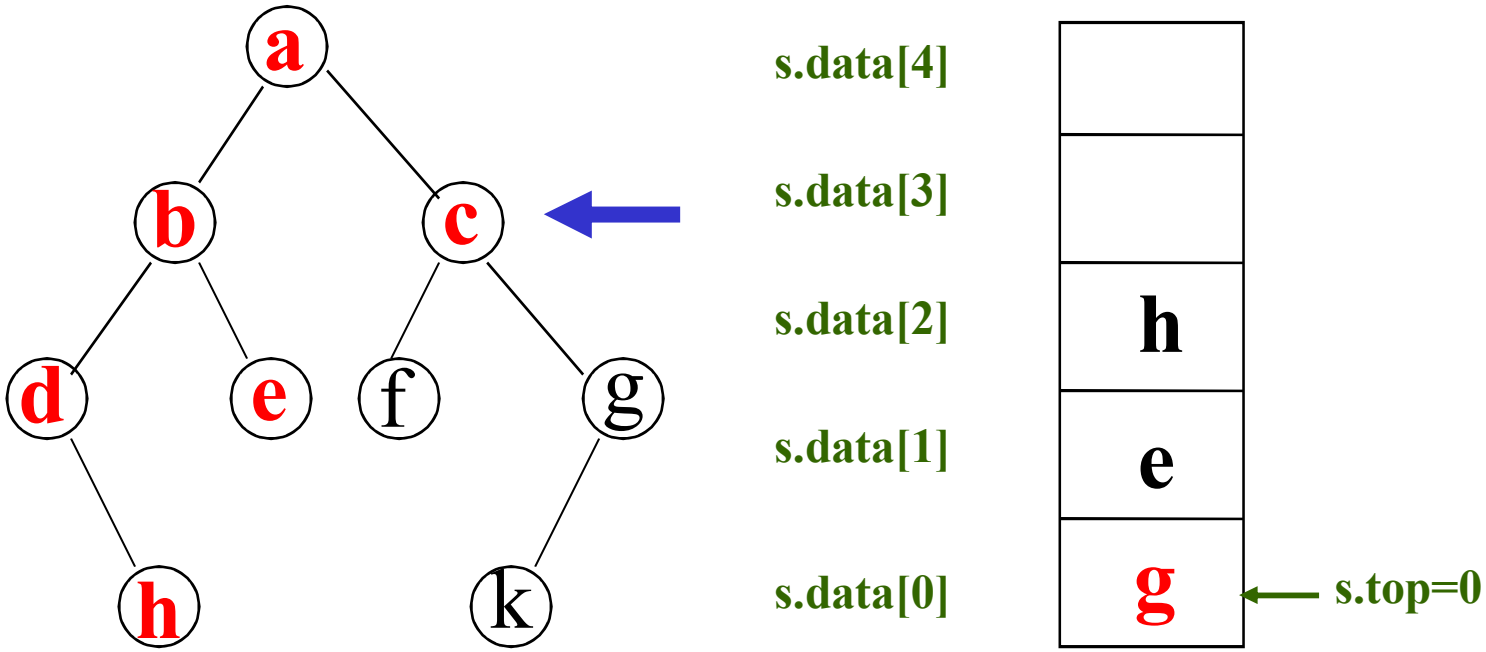
abdhec



先序遍历非递归算法的实现：访问根结点后，在访问左子树前，应保存其非空右子树 stop++-1

图中a的右子树先于b的右子树被保存，但是其访问要在b的右子树被访问后进行----先保存后访问----先进后出----借助栈实现

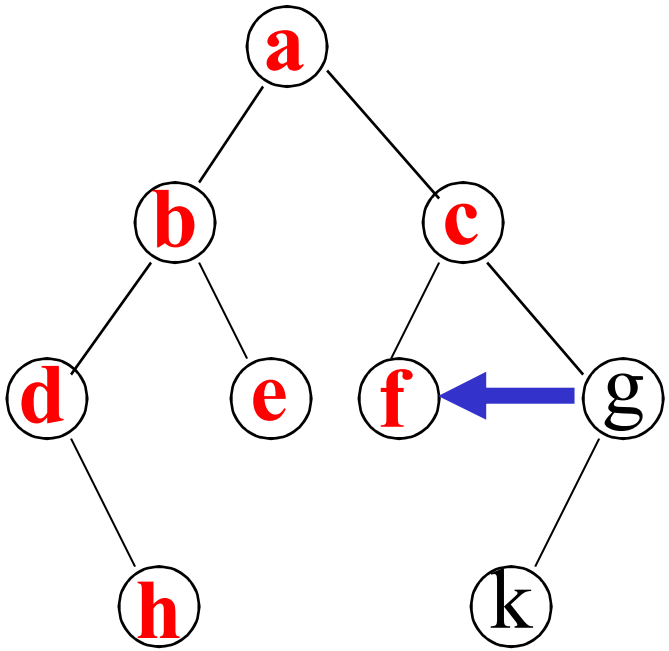
abdhec

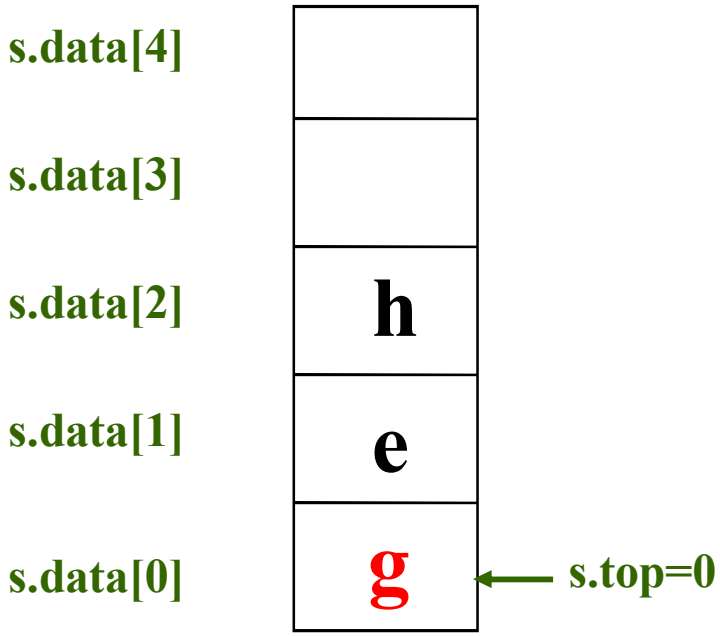


先序遍历非递归算法的实现：访问根结点后，在访问左子树前，应保存其非空右子树

图中a的右子树先于b的右子树被保存，但是其访问要在b的右子树被访问后进行----先保存后访问----先进后出----借助栈实现

abdhecf

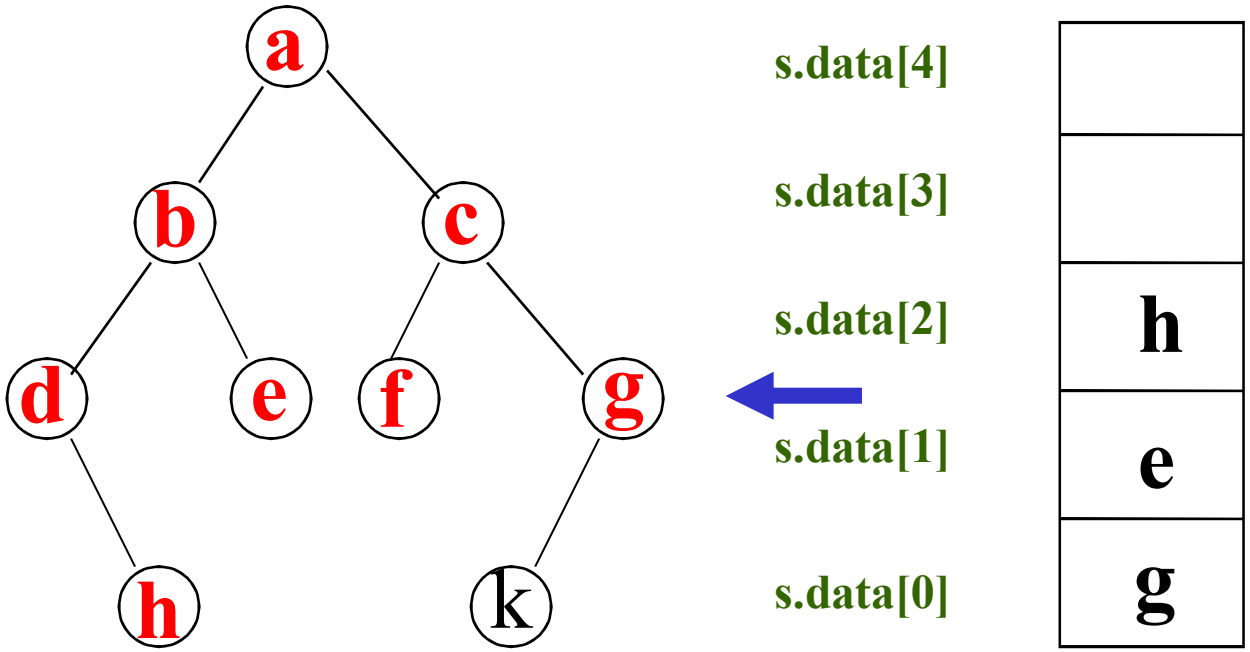




先序遍历非递归算法的实现：访问根结点后，在访问左子树前，应保存其非空右子树

图中a的右子树先于b的右子树被保存，但是其访问要在b的右子树被访问后进行----先保存后访问----先进后出----借助栈实现

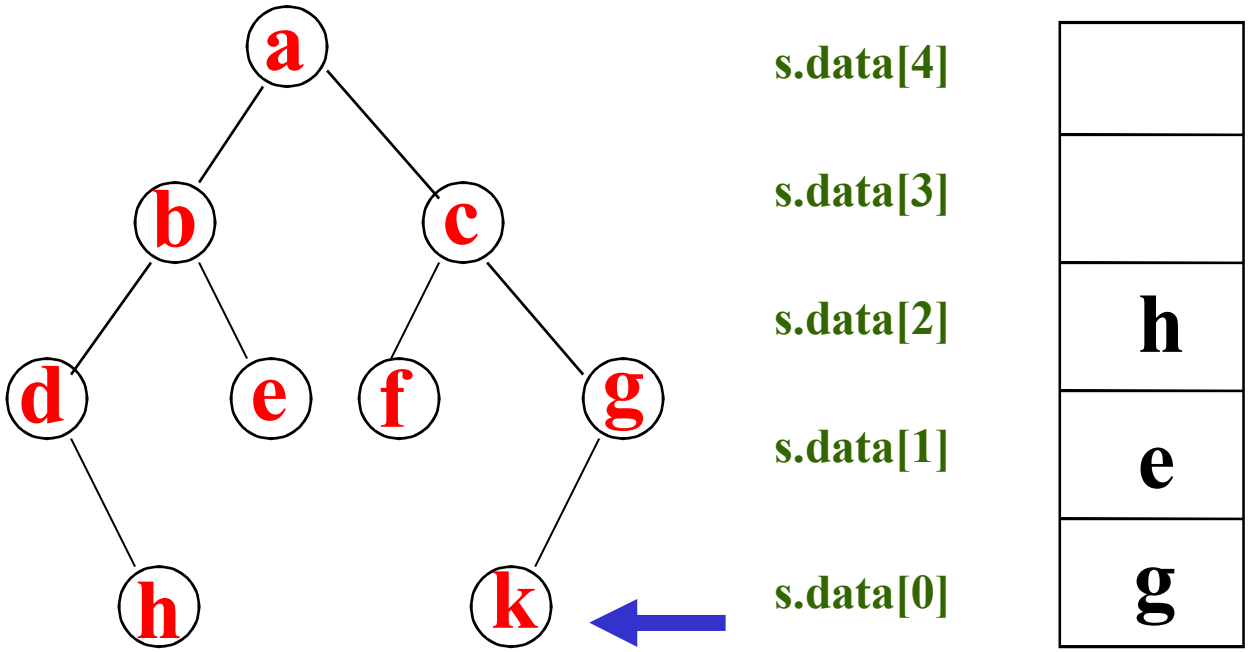
abdhecfg



先序遍历非递归算法的实现：访问根结点后，在访问左子树前，应保存其非空右子树 ← s.top=-1

图中a的右子树先于b的右子树被保存，但是其访问要在b的右子树被访问后进行----先保存后访问----先进后出----借助栈实现

abdhecfgk



先序遍历非递归算法的实现：访问根结点后，在访问左子树前，应保存其非空右子树 ← s.top=-1

图中a的右子树先于b的右子树被保存，但是其访问要在b的右子树被访问后进行----先保存后访问----先进后出----借助栈实现

顺序栈

```
#define MAX  10000
typedef  struct
  {BiTree  data[MAX];
   int  top;
  }SeqStack;
```

先序遍历的非递归描述

1 p=T, 初始化空栈;

2 若p 存在, 则访问该结点 p , 将p的非空右孩子入栈 , p=p→lc,后转至 2; 否则转3;

3 若栈不空, 取栈顶元素→p, 转2; 否则结束。

先序遍历算法的非递归描述

```
void PreorderTraverse(BiTree T){
  SeqStack  s ;
  s.top=-1; p = T;
  while(p){
    while(p){printf("%c",p->data);
                if(p->rc)
                  if(s.top==MAX-1) exit (0);
                  else s.data[++s.top]=p->rc;
                    p =p->lc;}
    if (s.top!=-1)  p=s.data[s.top--];
   }
}
```

中（根）序的遍历算法：

若二叉树为空树，则空操作；否则，

（1）中序遍历左子树；

（2）访问根结点；

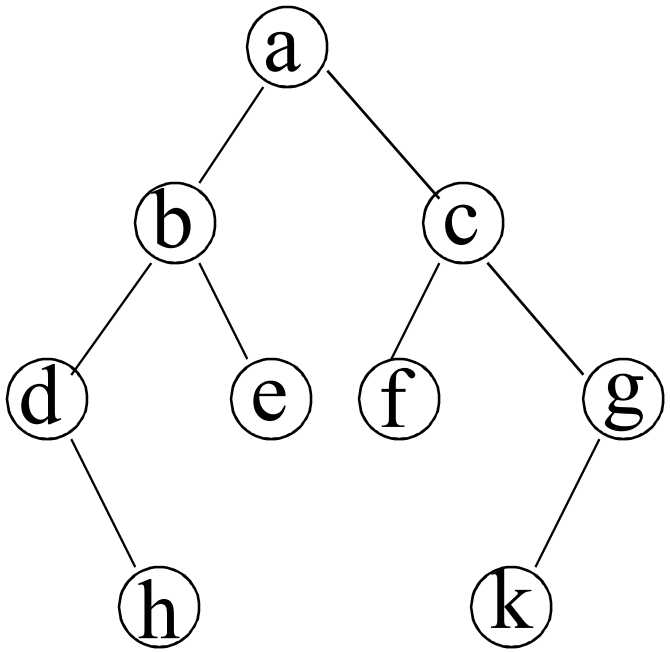
（3）中序遍历右子树。

中序遍历的递归算法

```
void zxbl (BiTree T )
{
  if (T) {
    zxbl(T->lchild);
    printf("%c",T->data);
    zxbl(T->rchild);
  }
}
```

中序遍历非递归算法的实现：访问根结点的左子树前，应保存其根结点，以便左子树访问结束后，访问根和根的右子树

图中a结点先于b结点被保存，但是其访问要在b及其右子树被访问后进行----先保存后访问----先进后出----借助栈实现



中序遍历算法的非递归描述

```
void InorderTraverse(BiTree T){
  SeqStack  s ;
  s.top=-1; p = T;
  while(p||(s.top!=-1)){
    while(p){if(s.top==MAX-1) exit (0);
                s.data[++s.top]=p; p =p->lc;}
    if (s.top!=-1)
        {p=s.data[s.top--];
         printf("%c",p->data);
         p = p->rc;   }
   }
}
```

后（根）序的遍历算法：

若二叉树为空树，则空操作；否则，

（1）**后序**遍历**左**子树；

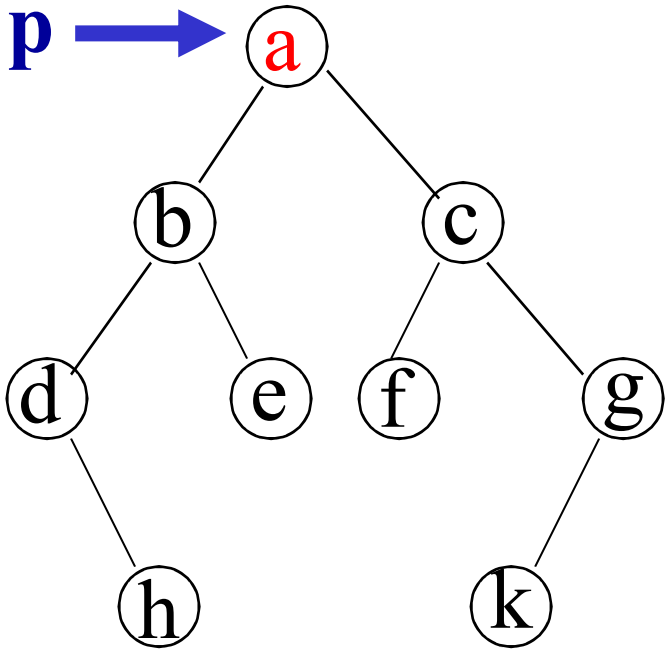
（2）**后序**遍历**右**子树；

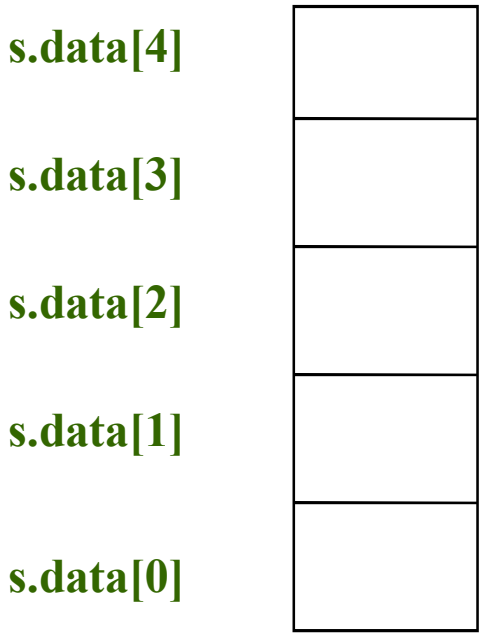
（3）访问**根**结点。

后序遍历非递归算法的实现：访问根结点的左子树前，应保存其根结点，以便左子树访问结束后，访问根的右子树和根

图中a结点先于b结点被保存，但是其访问要在b及其右子树被访问后进行----先保存后访问----先进后出----借助栈实现

图中结点只有在其左、右子树被访问后才能被访问



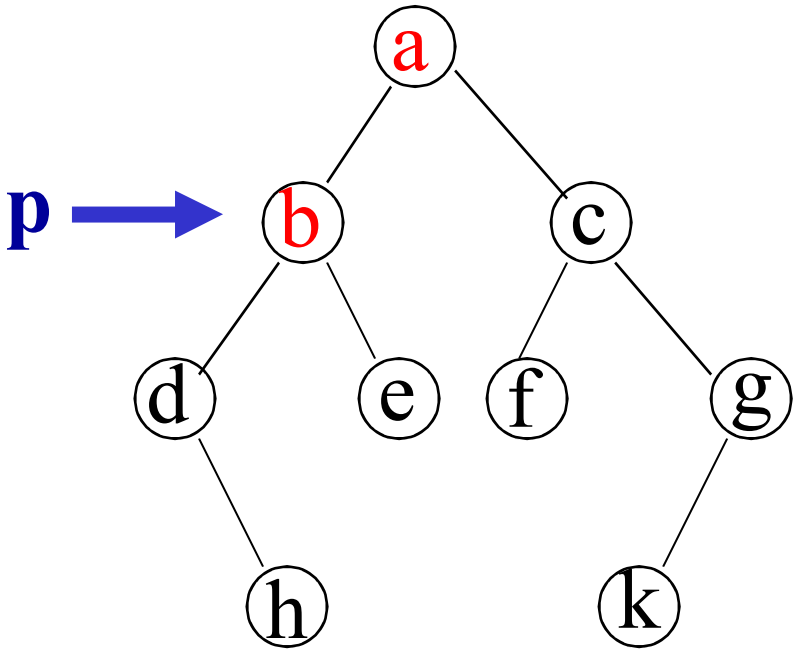


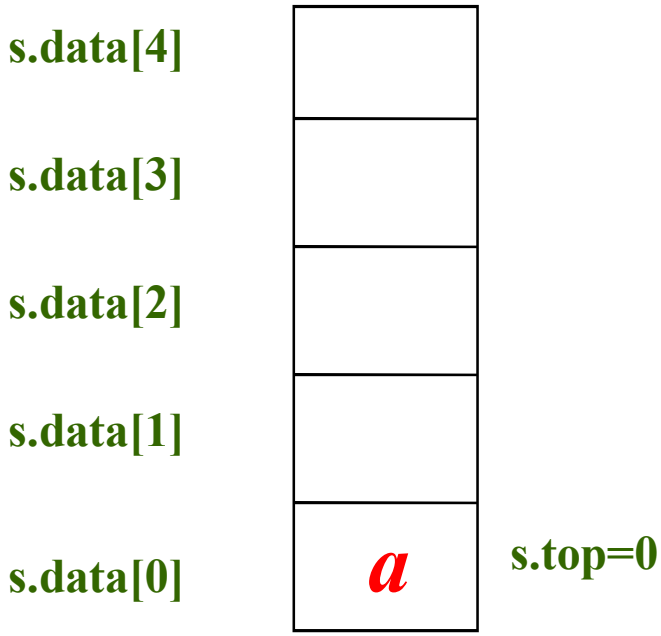
s.top=-1

后序遍历非递归算法的实现：访问根结点的左子树前，应保存其根结点，以便左子树访问结束后，访问根的右子树和根

图中a结点先于b结点被保存，但是其访问要在b及其右子树被访问后进行----先保存后访问----先进后出----借助栈实现

图中结点只有在其左、右子树被访问后才能被访问

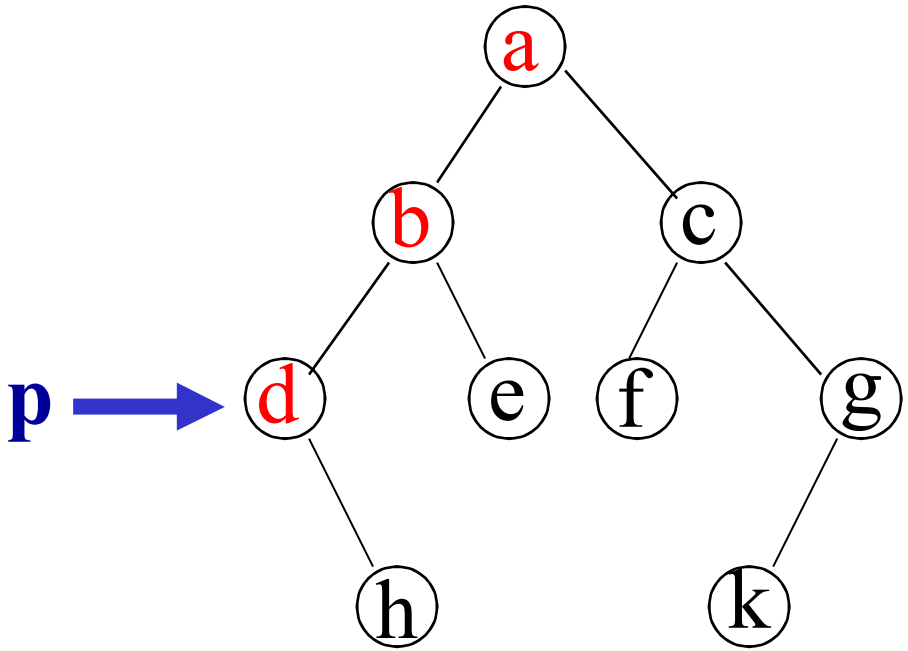


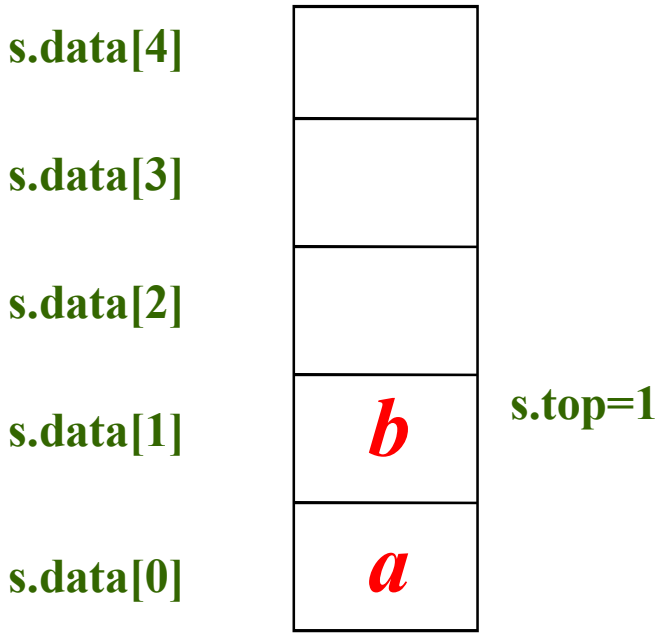


后序遍历非递归算法的实现：访问根结点的左子树前，应保存其根结点，以便左子树访问结束后，访问根的右子树和根

图中a结点先于b结点被保存，但是其访问要在b及其右子树被访问后进行----先保存后访问----先进后出----借助栈实现

图中结点只有在其左、右子树被访问后才能被访问

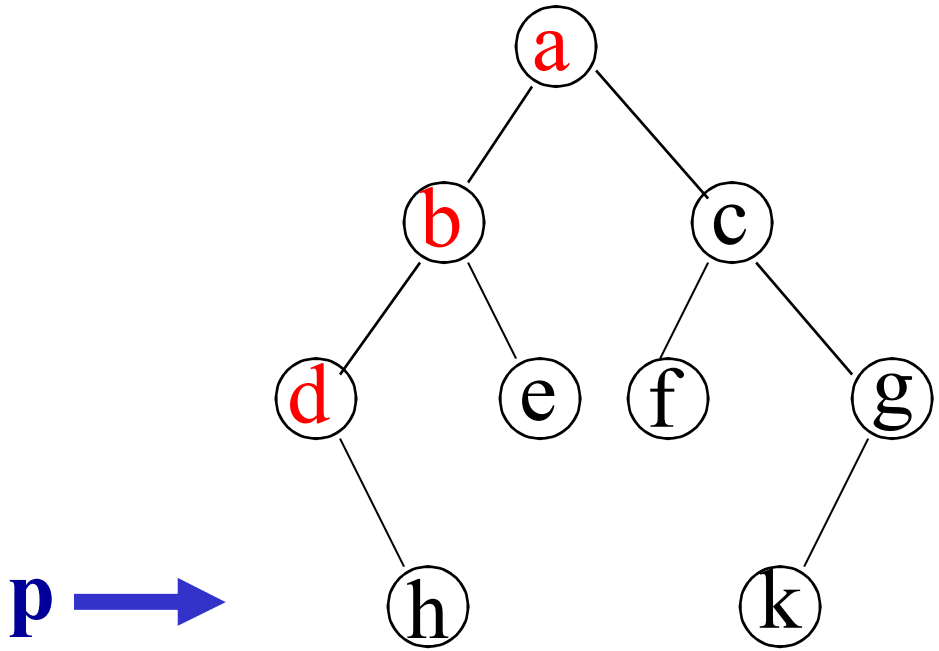


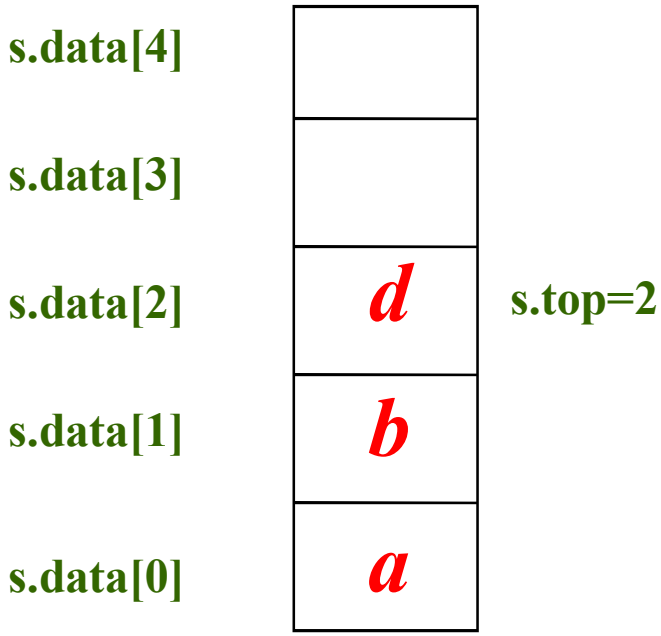


后序遍历非递归算法的实现：访问根结点的左子树前，应保存其根结点，以便左子树访问结束后，访问根的右子树和根

图中a结点先于b结点被保存，但是其访问要在b及其右子树被访问后进行----先保存后访问----先进后出----借助栈实现

图中结点只有在其左、右子树被访问后才能被访问

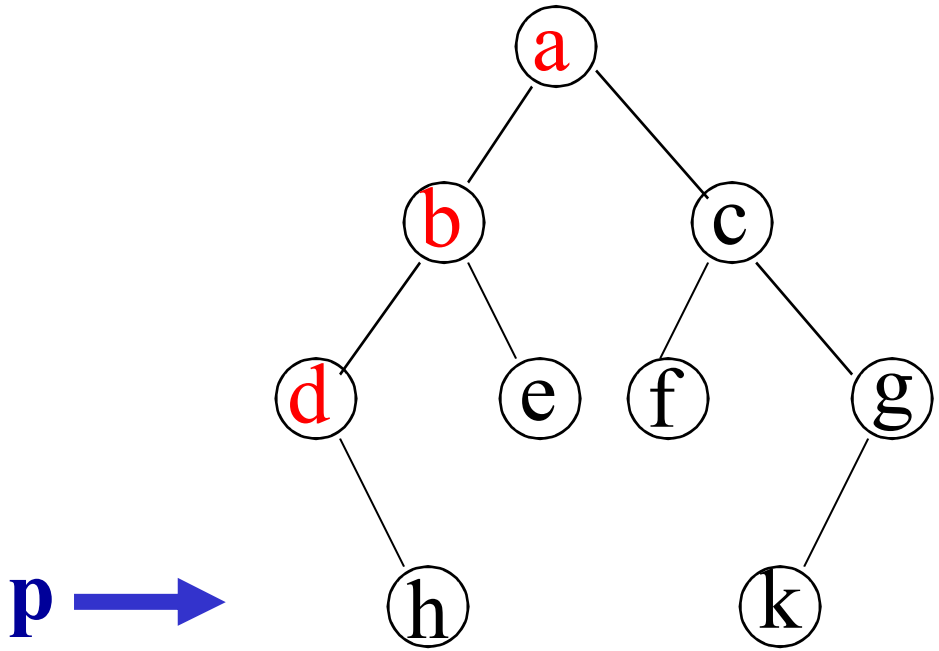


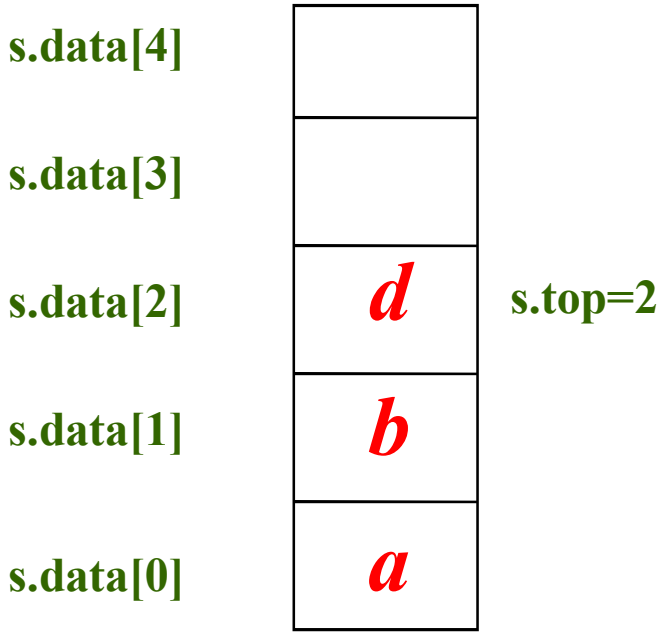


后序遍历非递归算法的实现：访问根结点的左子树前，应保存其根结点，以便左子树访问结束后，访问根的右子树和根

图中a结点先于b结点被保存，但是其访问要在b及其右子树被访问后进行----先保存后访问----先进后出----借助栈实现

图中结点只有在其左、右子树被访问后才能被访问



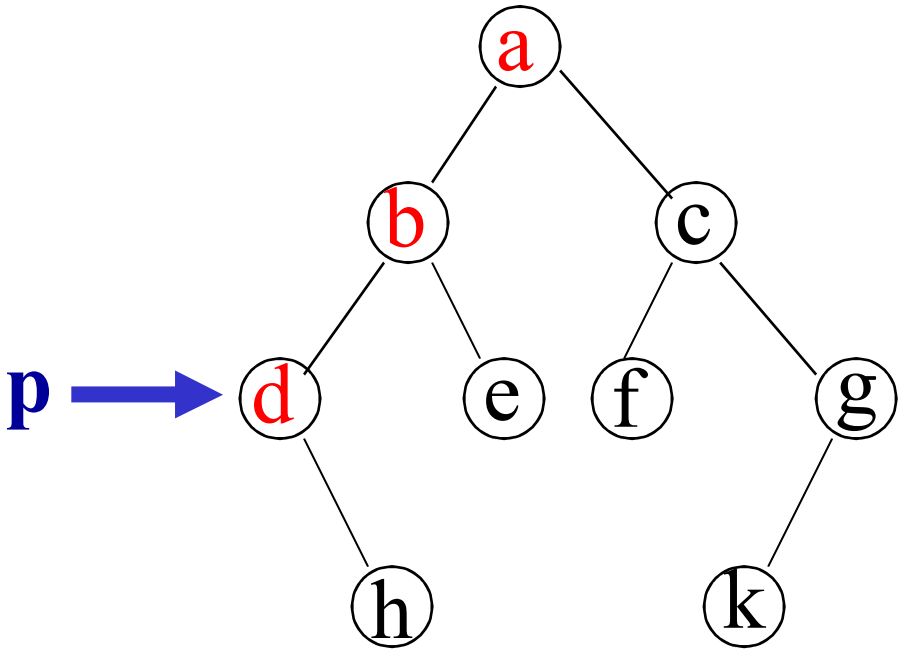


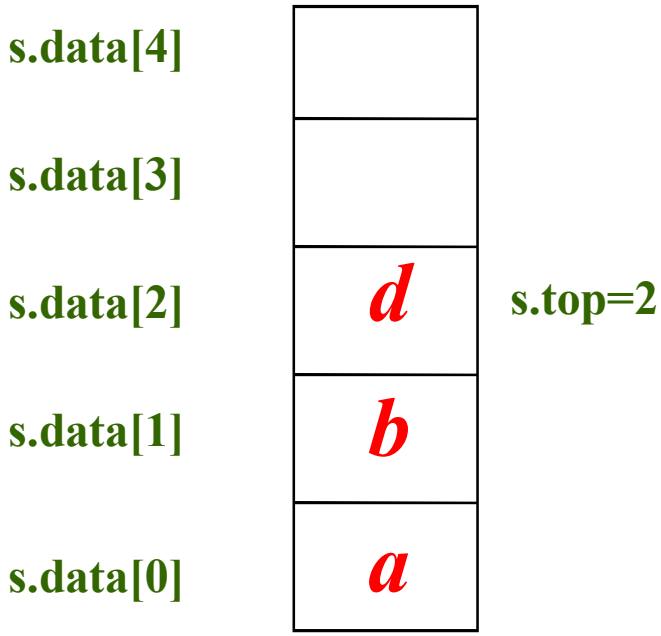
后序遍历非递归算法的实现：访问根结点的左子树前，应保存其根结点，以便左子树访问结束后，访问根的右子树和根

图中a结点先于b结点被保存，但是其访问要在b及其右子树被访问后进行----先保存后访问----先进后出----借助栈实现

图中结点只有在其左、右子树被访问后才能被访问

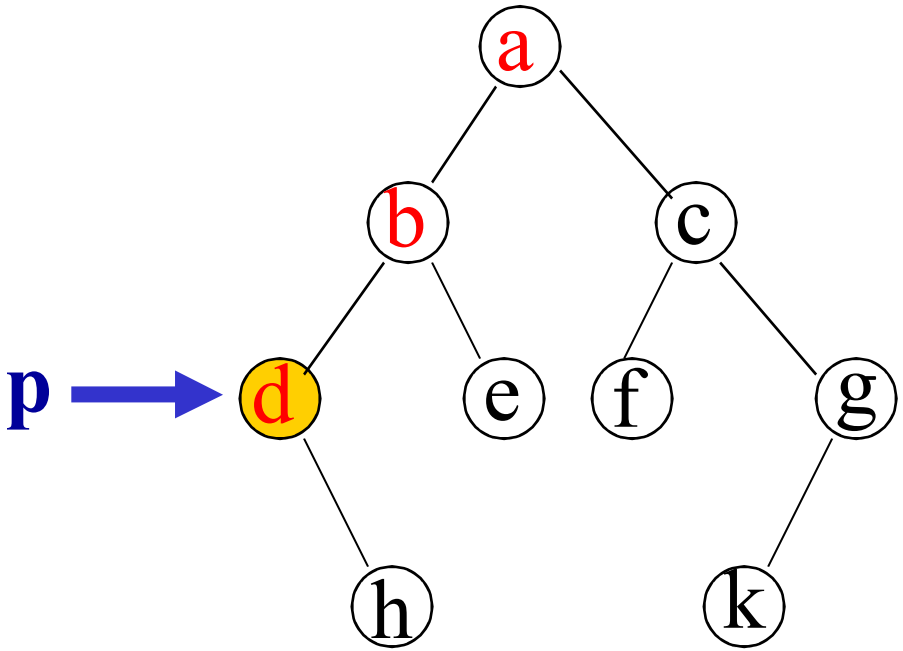
图中一个结点的左子树访问结束，回到该结点----》右子树，右子树访问结束后回到该结点，才能访问该结点

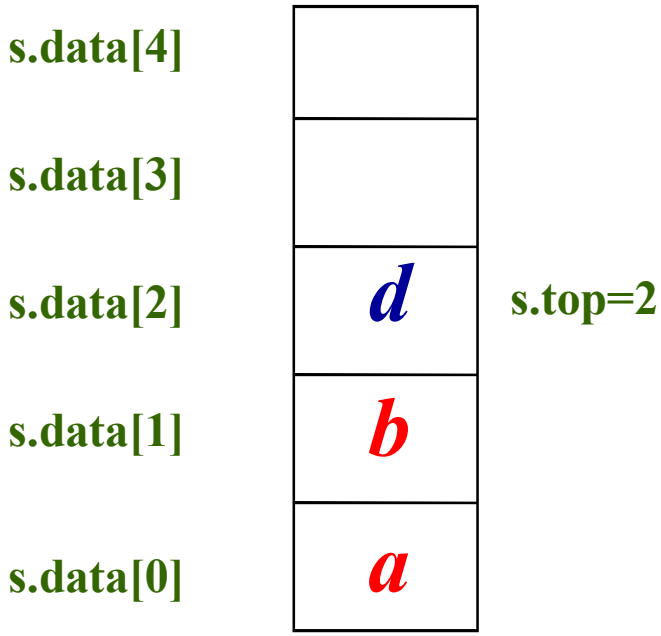




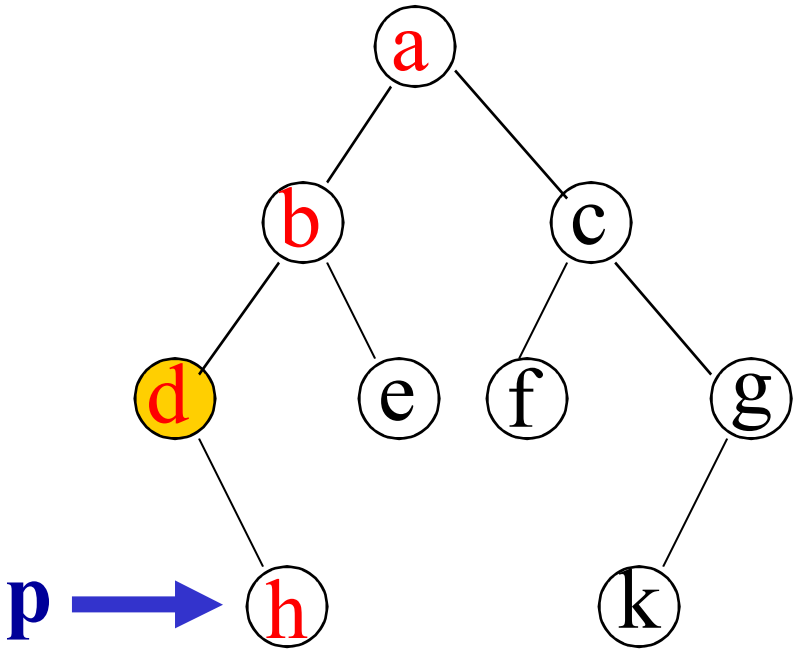
栈中结点要设标志域，指示该结点目前是被访问其左子树还是右子树

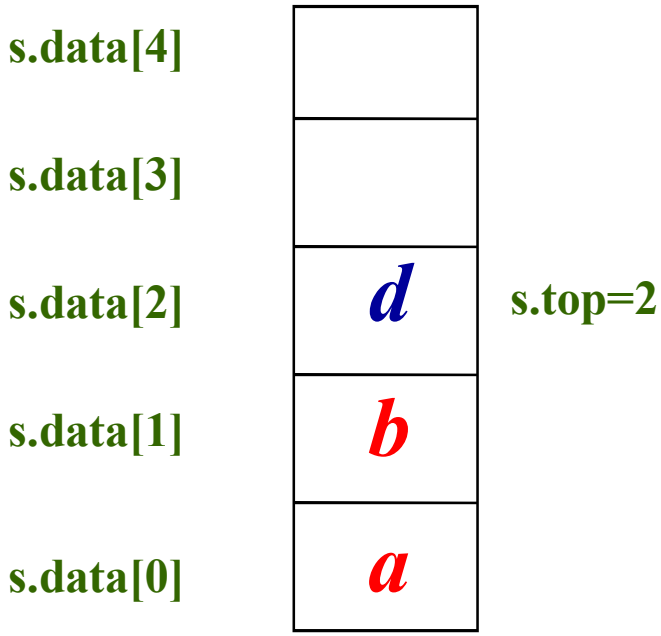
栈中结点红色标志指示该结点目前是被访问其左子树
蓝色标志指示该结点目前是被访问右子树



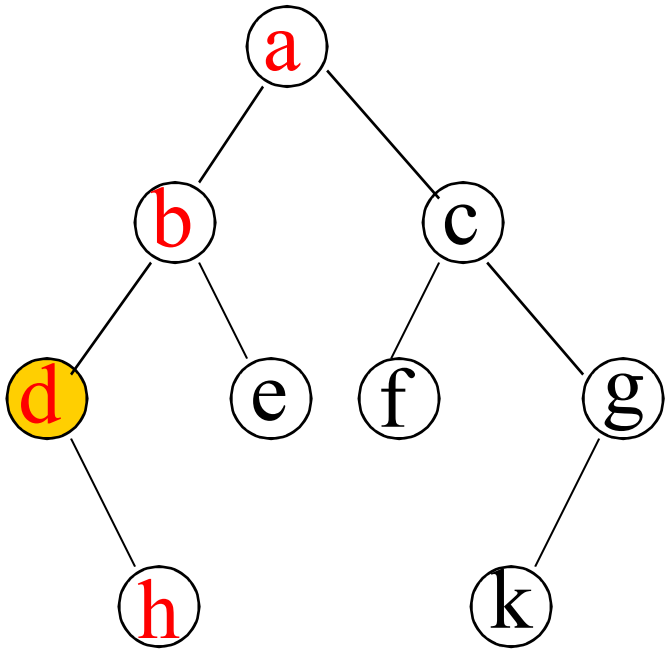


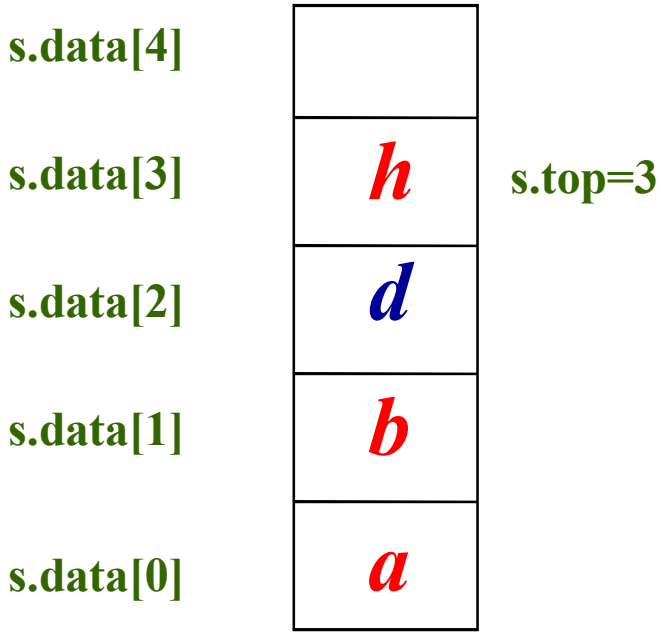
栈中结点红色标志指示该结点目前是被访问其左子树
蓝色标志指示该结点目前是被访问右子树





栈中结点红色标志指示该结点目前是被访问其左子树
蓝色标志指示该结点目前是被访问右子树





p

后序遍历的非递归描述

```
void postorder(BiTree T)
{SeqStack2 s;
  s.top=-1;
  p=T;
 do{while(p!=NULL)   {s.data[++s.top].d=p; s.data[s.top].flag=0; p=p->lc;}
      while((s.top>-1)&&(s.data[s.top].flag==1))
      { p=s.data[s.top--].d;   printf("%c",p->data); }
      if(s.top>-1)
       {s.data[s.top].flag=1;p=s.data[s.top].d;  p=p->rc;}
  }while(s.top>-1);
}
```