DS

写一递归函数实现在带索引的二叉搜索树(IndexBST)中查找第 k 个小的元素

```
Public BinaryNode findkth(BinaryNode t, int k) {
        if(k<=0 || t==null)
        return null;
    if(k<t.leftsize)
        return findkth(t.left ,k);
    else if(k>t.leftsize)
        return findkth(t.right, k-t.leftsize);
    else
        return t;
}
```

写一个程序判断一个二叉树是不是二叉搜索树

方法 1:检验左子树的最大值和右子树的最小值

```
Public boolean isBST(Node n) {
    If (n==null) return true;
    If (n.left!=null && max(n.left).element>n.element)
        return false;
    If (n.right!=null && min(n.right).element<n.element)
        return false;
    return isBST(n.left) && isBST(n.right);
}
Public Node max (Node n) {
    If (n==null) return null;
    While (n.right!=null) { n=n.right; }
    Return n;
}
Public Node min (Node n) {
    If (n==null) return null;
    While (n.left!=null) { n=n.left; }
```

```
    Return n;
}
```

求树 t 中各结点的度

```
class BinaryNode{
    int data, degree;
    BinaryNode firstchild;
    BinaryNode nextsibling;
}

public void degree(BinaryNode t){
    if(t==null) return;
    t.degree=0;
    if(t.firstchild!=null){
        t.degree++;
        BinaryNode p=t.firstchild;
        while(p.nextsibling!=null){
            t.degree++;
            p=p.nextsibling;
        }
    degree(t.firstchild);
    degree(t.nextsibling);
}
```

调整最小堆算法

```
Private static void percUp(Comparable[ ] a , int start) {
    int j=start , i=j/2;
comparable temp = a[j];
while(j>1) {
    if(a[i]<=temp) break;
    else {
        a[j]=a[i];
        j=i;
        i=i/2;
        }
}
a[j] = temp;
}
```

除了 03——07 年的重点算法外：

13 年的备用卷：利用二叉搜索树实现优先队列。

```
Class BinaryNode {
   BinaryNode left;
   BinaryNode right;
   Comparable data;
}

Public class BST {
   private BinaryNode root;

   public BST( )   {    root=null;}

   public BinaryNode insert(BinaryNode t ,    comparable x) {
        if(t==null)
            t=new BinaryNode(x, null, null);
        else if(x.compareTo(t.data)<0)
            t.left = insert(t.left , x);
        else if(x.compareTo(t.data)>0)
            t.right = insert(t.right , x);
        return t;
   }

     private BinaryNode findMax( BinaryNode t )
 {     if( t != null )
              while( t.right != null )
                    t = t.right;

         return t;
 }

private BinaryNode remove( Comparable x, BinaryNode t )
 {       if( t = = null )
                return t;
           if( x.compareTo( t.element ) < 0 )
                t.left = remove( x, t.left );
           else if( x.compareTo( t.element ) > 0 )
                t.right = remove( x, t.right );
           else if( t.left != null && t.right != null )
                   {      t.element = findMin( t.right ).element;
```

```
                    t.right = remove( t.element , t.right );
                }
        else
                t = ( t.left != null ) ? t.left : t.right;
 }
  public BinaryNode delete() {
     BinaryNode temp = findMax(root);
     remove(temp.data , root);
}
```

已知先序序列和中序序列建立二叉树算法

```
void CreateBT(String pres, ins ; BinaryNode <Type>* & t)
{    int inpos;
       String   prestemp, instemp ;
       if (pres.length( )= =0) t=NULL;

     else {   t=new BinaryNode;
                  t->element=pres.ch[0];       inpos=0;
                  while (ins.ch[inpos]!=t->element) inpos++;

                  prestemp=pres(1,inpos);
                  instemp=ins(0,inpos-1);
                  CreateBT(prestemp, instemp, t->left);
                  prestemp=pres(inpos+1, pres.length( )-1);
                  instemp=ins(inpos+1, pres.length( )-1);
                  CreateBT(prestemp, instemp, t->right);
              }
}
```

此外还应该有二叉树先序，中序非递归算法

手写练习

4) 设 W 为一个二维数组，其每个数据元素占用 6 个字节，行下标 i 从 0 到 8，列下标 j 从 0 到 3，则二维数组 W 的数据元素共占用_____个字节。W 中第 6 行的元素和第 4 列的元素共占用_____个字节。若按行主顺序存放二维数组 W，其起始地址为 100，则二维数组 W 的最后一个数据元素的起始地址为_____。

答案：216　　72　　310

(2) 有 4 个数据依次入栈，有 __14__ 种出栈序列。

$$C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!n!}$$

(7) 当两个栈共享一存储区时，栈利用一维数组 A[n]表示，两栈顶指针为 top[0]与 top[1]，则栈满时的判断条件为___top[0]+1=top[1]_ 或者 top[0] = top[1]+1_____。

(6) 使用加权规则得到改进的 Union 操作 WeightedUnion，其目的是：（ B ）

A. 提高 Union 操作的时间性能

B. 提高 Find 操作的时间性能

C. 减少 Union 操作的空间存储

D. 减少 Find 操作的空间存储

(7) 使用 Kruscal 算法求解最小生成树时，为了设计效率较高的算法，数据结构方面可以选择：（ A ）

A. 利用最小堆存储边

B. 利用栈存储结点

C. 利用二维数组存储结点

D. 利用并查集存储边

(4) 在指针 L 指向一带头结点的双向循环链表，则判断该表中只有一个元素结点的条件是：_____（设结点结构为

lLink	data	rLink

）

L->rLink->rLink= =L && L->rLink!=L;

(10) 在 B 树的删除操作中，最坏情况下可能需要读写磁盘_____次。（假设内存工作区足够大，但操作之前各个结点都存放在磁盘上，B 树的高度为 h）

A 2h-2　　B 2h-1　　C 3h-1　　D 3h-2 次

D

操作系统

1. 静态重定位的时机是____.

- 程序链接时
- 程序编译时
- 程序装入时
- 程序运行时

13. 在请求分页存储管理中，当访问的页面不在内存时，便产生缺页中断，缺页中断是属于____。

- 程序中断
- 访管中断
- I/O 中断
- 外中断

15. LRU 置换算法所基于的思想是____。

- 在最近的过去用得少的在最近的将来也用得少
- 在最近的过去很久未使用的在最近的将来也不会使用
- 在最近的过去很久未使用的在最近的将来会使用

在最近的过去用得多的在最近的将来也用得多

15. 进程资源静态分配方式是指一个进程在建立时就分配了它需要的全部资源，只有该进程所要资源都得到满足的条件下，进程才开始运行。这样可以预防进程死锁。静态分配方式破坏死锁的____必要条件。

非剥夺式等待条件

循环等待条件

互斥条件

占有且等待

16. 银行家算法通过破坏____来避免死锁。

不可抢占条件

部分分配条件

互斥条件

循环等待条件

20. 资源的按序分配策略可以破坏_____条件。

占有且等待资源

循环等待资源

互斥使用资源

非剥夺资源

1. Linux 系统中的 slab 分配器，采用____内存管理方式。
A. 固定分区 B.分页式 C.伙伴系统 D.分段式

2. 实模式下 16 位 CPU 使用段偏移方式的寻址能力为_____。
A. 64kb B. 1M C. 16M D. 4G

3. 下面哪条指令不是从实模式进入保护模式需要的指令_____。

A. lgdt [GdtPtr]　B. out 92h, al　C. jmp $　D. mov cr0, eax

4. FAT12 文件系统里，FAT 表的数量和每张 FAT 表占用的扇区数量为_____。

A. 2, 9　B. 2, 10　C. 3, 9　D. 3, 10

5. 操作系统里没有下面哪种描述符表_____。

A. GDT　B. LDT　C. IDT　D. KDT

（Global Descriptor Table） (Local Descriptor Table)　（Interupt Descriptor Table）

6. C 语言里面调用汇编的函数方法为_____。

A. C 代码中使用 extern 声明，汇编中使用 global 导出

B. C 代码中使用 global 声明，汇编中使用 extern 导出

C. C 代码中使用 extern 声明，汇编中使用 extern 导出

D. C 代码中使用 global 声明，汇编中使用 global 导出

7. 在 Orange'S 系统里，loader 的作用不包括_____。

A. 引导扇区的启动　B. 跳入保护模式

C. 加载 kernel 并将执行转交 kernel　D. 启动分页机制

8. 请画出经典的五状态进程模型及其状态转换图。

8. 请画出经典的七状态进程模型及其状态转换图。

8. 叙述操作系统三个最基础的抽象，为什么要引入它们？