数据结构与算法设计
习题课

一至五章

- **什么是ADT？**
- **它包含哪几个部分？**

♦ 1、写出以下各语句执行次数

(1) for (i=0;i<n;i++) $n+1$

(2) for (j=0;j<n;j++) $n(n+1)$

(3) { c[i,j] = 0; n^2

(4) for (k=0;k<n;k++) $n^2(n+1)$

(5) c[i,j]=c[i,j]+a[i,k]*b[k,j] n^3

}

♦ 2、写出加@语句执行次数

```
t = 0;
for( i=1; i<=n; i++)
    for (j=1; j<=i; j++)
        for (k=1; k<=j; k++)
@                       t ++;
```

语句频度= $\sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=1}^{j}1 = \sum_{i=1}^{n}\sum_{j=1}^{i}j = \sum_{i=1}^{n}\frac{i(i+1)}{2} = \frac{1}{2}\left(\sum_{i=1}^{n}i^{2}+\sum_{i=1}^{n}i\right)$

$$= \frac{1}{2}\left(\frac{n(n+1)(2n+1)}{6}+\frac{n(n+1)}{2}\right) = \frac{n(n+1)(n+2)}{6}$$

- **4.　现有两个不同的算法实现相同的功能，其时间复杂度分别为$T(n)=2^n$和$T(n)=n^{10}$。**
- **假设某计算机可连续运算的时间为10^7秒，每秒可执行基本操作10^5次。**
- **那么在这台计算机上运行程序，这两个算法可以求解的问题规模各是多少？选择哪个算法更好？请说明理由。**
- **答：第一个算法较好。虽然一般情况下多项式阶的算法优于指数阶的算法，但高次多项式算法在n的很大范围内不如某些指数阶的算法。**

- 这台计算机共能运行10^{12}次基本操作。
  - 对于$T(n)=2^n$的算法，$2^n=10^{12}$。n=39.9
  - 对于$T(n)=n^{10}$的算法，$n^{10}=10^{12}$。n=15.8.

n	2^n		n^{10}
10	1024	<<	10^10
50	1.1*10^15	<	9.8*10^16
60	1.2*10^18	>	6.0*10^17

- **5. 若长度为 n 的线性表采用顺序存储结构，在其第 i $(1 \le i \le n)$ 个位置之后插入一个新元素的算法时间复杂度是多少？**
- **答： O(n)**
- **6. 若采用带尾指针的单链表结构，则删除最后一个节点和第一个节点的复杂度分别为多少？**
- **答： O(n)和O(1)**

- 7. 已知长度为n的顺序表A=(a1, a2, …, an)。请将A转换为A’=(an, an-1, …, a1)。
- 基本思想：依次将a1和an，a2和an-1进行对换。

```
void SqReverse(L){
    m=n/2;
    ElemType tpElem;
    for( i=0; i< m; i++){
        tpElem = L. elem[i];
        L. elem[i] = L. elem[n-i-1] ;
        L. elem[n-i-1] = tpElem;
    }
}// SqReverse
```

- 9. 已知线性表中的元素以值非递减有序排列，并以顺序表为存储结构。试写一高效算法，删除表中所有值大于mink且小于maxk的元素 。
- 例如：删除（5-13）之间的元素

2	5	6	6	7	9	11	14	14	20

2	14	14	20						

- **10.** 已知一个顺序表的元素按元素值<u>非递减有序</u>排列，编写一个函数删除向量中多余的值相同的元素。

2	5	6	6	7	9	14	14	14	20

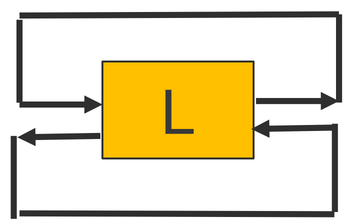
2	5	6	7	9	14	20			

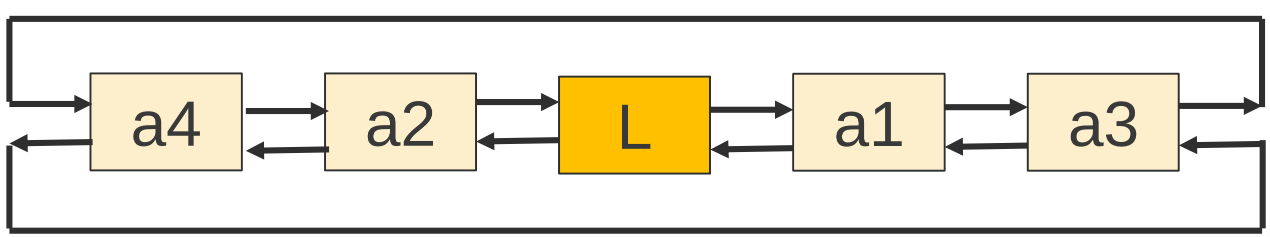
- **11.** 若线性表中的元素排列是随机的，并以顺序表为存储结构。试写一高效算法，删除表中所有值大于**mink**且小于**maxk**的元素。
- 例如：删除（**5-13**）之间的元素
- 方法同上题。

3	12	15	6	6	17	9	11	14	14	2

12	15	17	14	14						

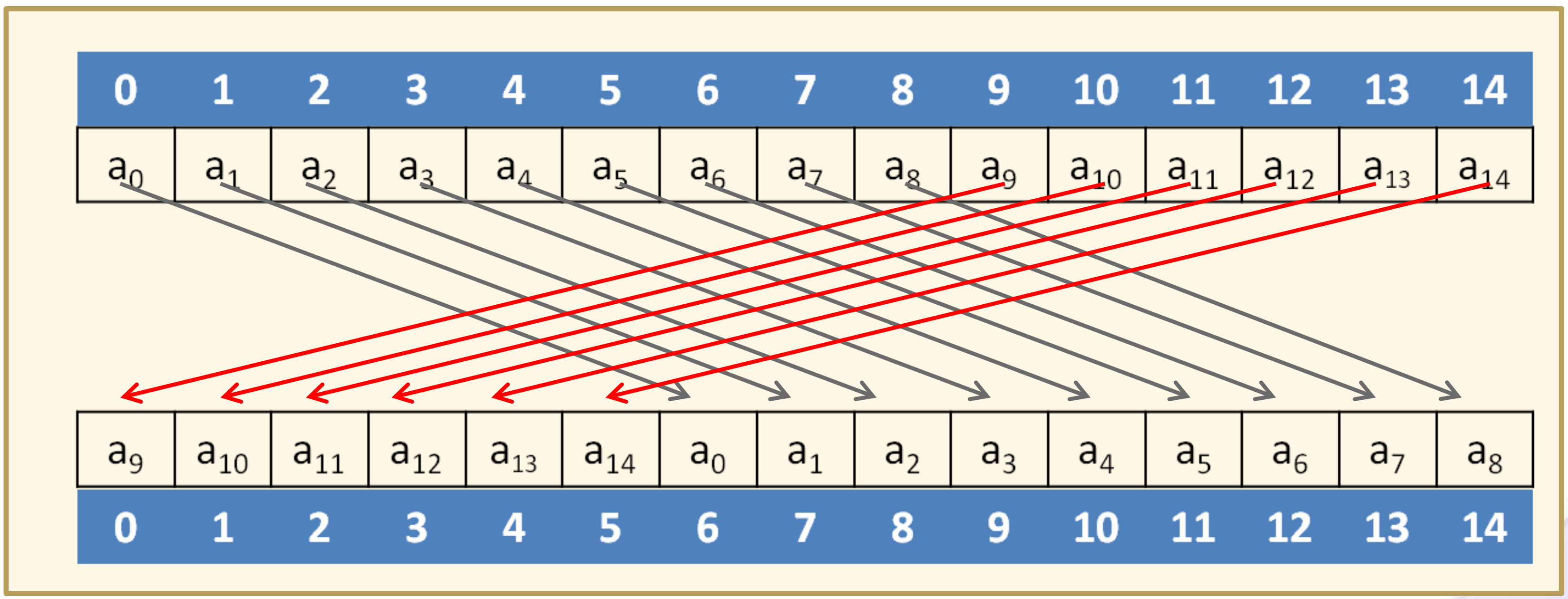
- 12. 用带头节点的双向循环链表表示线性表
- $L = (a_1, a_2, \ldots , a_n)$。请设计一个复杂度为O(n)的算法，将L中的节点按照下列顺序排列
- $L = (a_1, a_3, a_5, \ldots , a_n, \ldots,a_6,a_4, a_2)$。



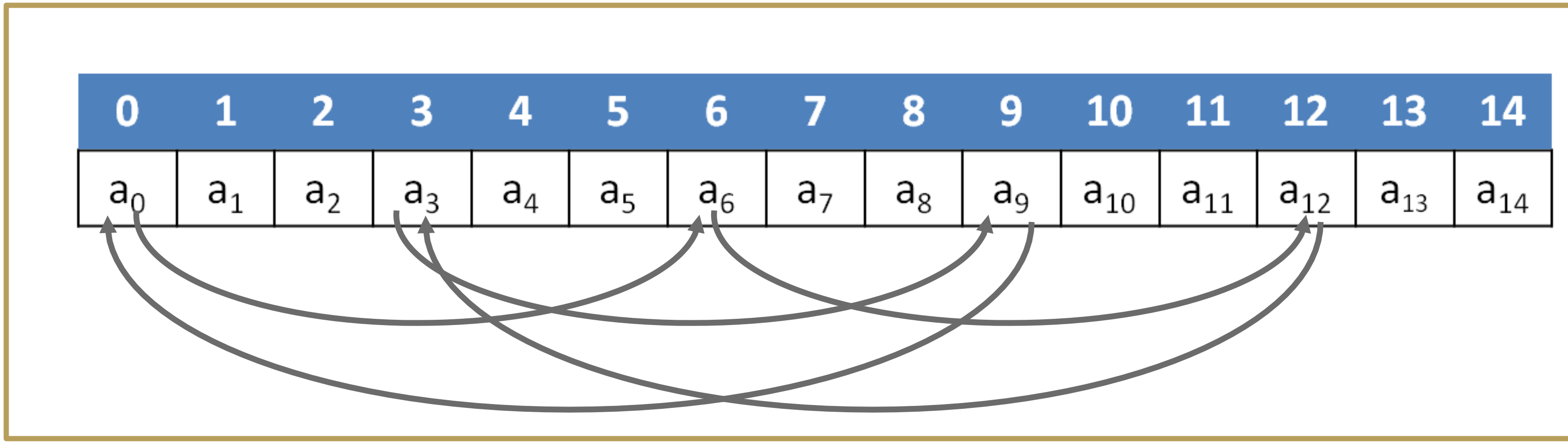


◆ **13 void RSh(int A[n], int k)//把数组A的元素循环右移k位，只用一个辅助存储空间。例如：**

n=15, k=6



- n=15, k=6

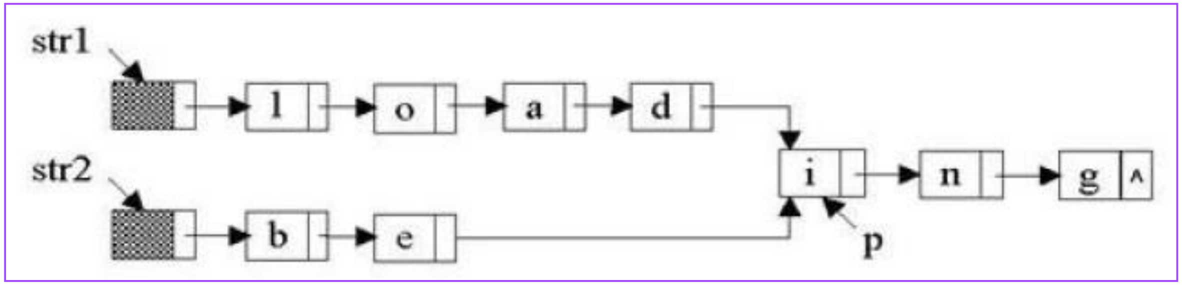


基本思想：
求出n和k的最大公约数gcn。上例中为3。
那么共需要gcn轮元素的循环移动。
用临时变量tmp保存下一个要移动的元素。

```
void RSh(int A[n], int k){
   k = k%n;//k有可能大于n
   if(k==0) return;//若k=0 移动了整数圈，不需要移动元素
   for( i=0; i<gcn; i++){
       t=i+k; tmp=a[i];  //
       while(t!=i) {
             tmp←→a[t];
             t=(t+k)%n; //指针循环向后移动k个位置
          }//while
    }//for
}//RSh
```

- **15. 识别读入的一个字符序列是否为反对称的字符序列(以字符’ & ’为对称轴)。**
- **a) 例如:abcd&dcba@ 是反对称字符序列;**
- **b) abc&@ 或 abc&abc@ 或 ab&bac@都不是反对称字符序列。**
- 答：用栈实现。如果当前还未读到&，则字符依次进栈。如果已经读到了&，则栈顶元素出栈，并与当前读入元素比较。若不相等则不是对称的。当字符尚未读取完毕而堆栈已为空，则说明是不对称的。当字符已经读取完毕而堆栈非空，则说明是不对称的。
- 若没有对称轴怎么办?

- 15 假定采用带头结点的单链表保存单词，当两个单词有相同的后时缀，则可共享相同的后缀存储空间，例如，“loading”和“being”，如下图所示。设str1和str2分别指向两个单词所在单链表的头结点，链表结点结构为（data, next），请设计一个时间上尽可能高效的算法，找出由str1和str2所指向两个链表共同后缀的起始位置（如图中字符i所在结点的位置p）。要求：
- （1）给出算法的基本设计思想。
- （2）根据设计思想，采用C或C++语言描述算法，关键之处给出注释。
- （3）说明你所设计算法的时间和空间复杂度。



- **17. 有两个栈 s1 和 s2 共享存储空间 c(1, m)，其中一个栈底设在 c[1]处，另一个栈底设在 c[m]处.**
- **假设top1 是 s1 的栈指针，top2 是s2 的栈指针，初始时两个的指针都指向各自的栈底。请问两个栈出现上溢和下溢的条件各是什么?**
- **答:**
- **当 top2=top1+1 时两个栈都满，此时若压栈则出现上溢。**
- **当 top1=1 时栈 1 为空，若出栈则出现下溢。**
- **当top2=m 时栈 2为空，若出现下溢。**

- 18. 设栈的输入序列是1、2、3、4，则可能是其出栈序列有哪些？
- n=1时， f(1)=1→1
- n=2时， f(2)=2→12, 21
- n=3时
  - （1）1第一个出栈：1 _ _， → 1 2 3， 1 3 2
  - （2）1第二个出栈：_ 1 _， → 2 1 3
  - （3）1第三个出栈：_ _ 1， → 2 3 1， 3 2 1
  - $f(3) = f(2) + f(1)f(1)+ f(2) = 5;$
- $f(4) = f(3) + f(1)f(2)+ f(2)f(1) + f(3) = 14;$
- 有n个不同的元素依次进栈，出栈的顺序共有：

- 令f(0)=1；
- 当n = 1时，f(1) = 1;
- 当n = 2时，f(2) = 2 = f(0)f(1) + f(1)f(0);
- 当n = 3时，f(3) = 5 = f(0)f(2) + f(1)f(1)+ f(2)f(0);
- 当n = 4时，f(4) = 14 = f(0)f(3) + f(1)f(2)+ f(2)f(1) + f(3)f(0);
- **f(n) = f(0)*f(n-1) + f(1)*f(n-2) + ... + f(n-1)*f(0)**

$$f(n)=\sum_{i=0}^{n-1} f(i)f(n-1-i)$$

此即catalan数。

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

- 假设S和X分别表示入栈和出栈操作，初态和终态都为空的入栈和出栈操作序列可以表示为仅由S和X组成的序列。则如果某序列表示的操作是不可实现的，则称该序列为非法序列，否则称为合法序列。
- 1）输入一个序列时，如何判断该序列是否是合法的？
- 2）长度为2n的序列，有多少种合法的样式呢？

SSSXXX
SXSXSX
SXXSSX

练习题

- 用静态链表实现的队列如右图所示:
- front = 9, rear = 7.
- 队列有头结点
- 当执行下列操作时,队列有什么变化?
- 1) 元素S入队
- 2) 队首元素出队

space	0	A	1
	1	B	3
	2	C	4
	3	D	6
	4	E	10
	5	F	7
	6	G	8
rear	7	H	0
	8	I	0
front	9	J	2
	10	K	5

ce	0	A	1
	1	B	3
	2	C	4
	3	D	6
	4	E	10
	5	F	7
	6	G	8
ear	7	H	0
	8	I	0
ont	9	J	2
	10	K	5

space	0	A	3
rear	1	S	0
	2	C	4
	3	D	6
	4	E	10
	5	F	7
	6	G	8
	7	H	1
	8	I	0
front	9	J	2
	10	K	5

1) 元素S入队

space	0	A	2
rear	1	S	0
C	2	C	3
	3	D	6
	4	E	10
	5	F	7
	6	G	8
	7	H	1
	8	I	0
front	9	J	4
	10	K	5

2) 队首元素出队

- 设有一个矩阵A[1..n, 1..2n-1]，将三个顶点分别为a[1, n]、a[n, 1]和a[n, 2n-1]的三角形区域内所有元素按行存放到一个一维数组B中。下图是n=3的存储示意图，问：
- (1) 一维数组*B*要有多少个元素？
- (2) 若在一维数组*B*中从0号位置开始存放，则矩阵A中的任一元素a_{ij}应存于一维数组B的什么下标位置？给出计算公式。

$$A=\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \end{bmatrix}$$

K	0	1	2	3	4	5	6	7	8
B[k]	a13	a22	a23	a24	a31	a32	a33	a34	a35

- **(1)1+3+5+…+2n-1 = n²**
- **(2)合法范围：** $n-(i-\mathbf{1}) \le j \le n+(i-\mathbf{1})$
- $a_{ij} = \begin{cases} (i-1)^2 + (j-(\mathbf{n}-\mathbf{i}+\mathbf{1})), \text{位置合法} \\ \text{无} \end{cases}$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \end{bmatrix}$$

A[1..n, 1..2n-1]

K	0	1	2	3	4	5	6	7	8
B[k]	a13	a22	a23	a24	a31	a32	a33	a34	a35

广义表的定义--练习

1. GetTail ((b, k, p, h))＝ (k, p, h) ；
2. GetHead (((c, d) ,(a, b))) ＝(c,d) ；
3. GetTail (((a, b), (c,d))) ＝((c, d)) ；
4. GetTail (GetHead(((a,b), (c,d)))) ＝(b) ；

   (a,b)

5. GetTail ((e))＝ ( ) ；
6. GetHead (( )) ＝ ( ) ；
7. GetTail (( )) ＝ ( ) 。

- 利用广义表的HEAD和TAIL操作，写出如上题的表达式，把原子banana从下列广义表中分离出来。
- 1）L1=(apple, (pear), ((banana)), (((orange))))
- 2）L2=(apple, (pear,(banana),orange));
- [解答]
- (1) HEAD(HEAD(HEAD(TAIL(TAIL(L1)))))
- (2) HEAD(HEAD(TAIL(HEAD(TAIL(L2)))))

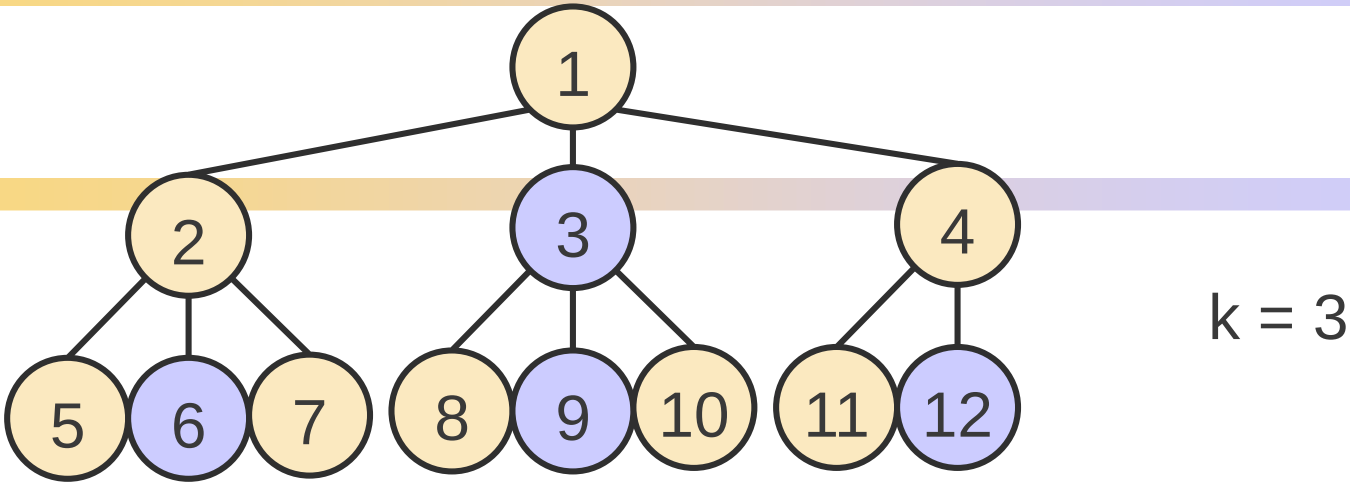
◆ 假设广义表采用头尾链表存储，试编写递归算法，判断两个广义表是否相等。

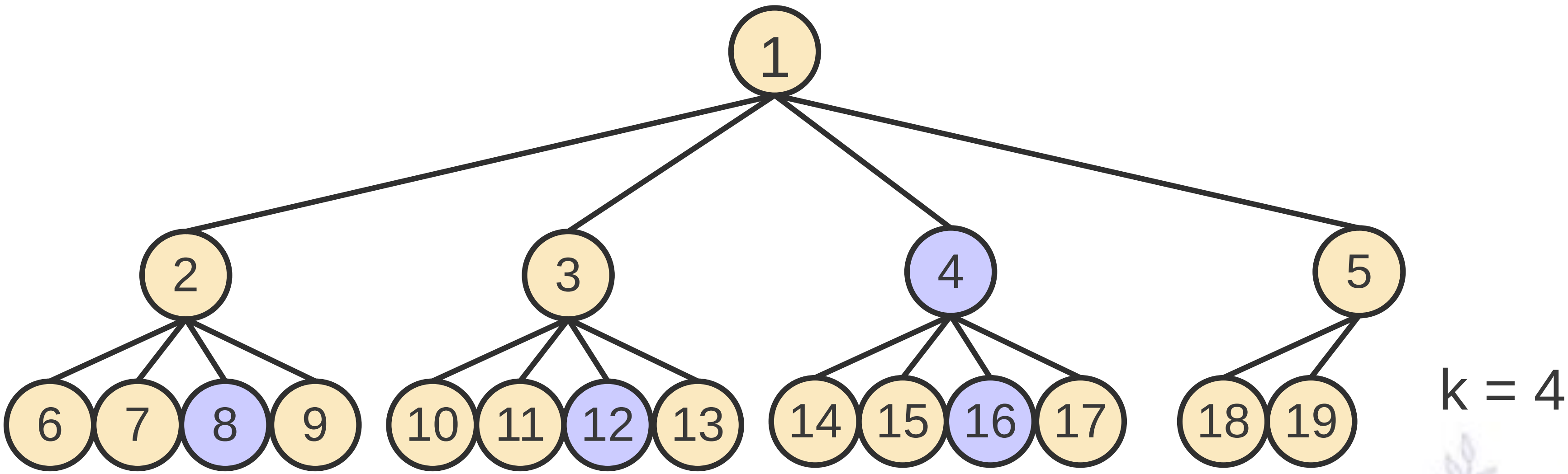
```
int CompGList ( GList L1, GList L2) {
   if (!L1 && L2 || L1 && !L2) return 0;
   if (!L1 && !L2) return 1;
   if (L1->tag != L2->tag) return 0;
   if (L1->tag ==ATOM && L1-> data != L2-> data)  return 0;
   if (L1->tag == LIST) {
      if(!CompGList (L1->ptr.hp, L2->ptr.hp) ) return 0;
      if(!CompGList (L1->ptr.tp, L2->ptr.tp) ) return 0;
   return 1;
}// CompGList
```

第5章 习题

- 1. 针对k叉树请回答下面问题:
- （1）k叉树的第 i 层上至多有多少个结点?
- （2）深度为t的k叉树上至多含多少个结点?
- （3）具有 n 个结点的完全k叉树的深度是多少?
- （4）若对含 n 个结点的完全k叉树从上到下且从左至右进行 1 至 n 的编号，则对完全k叉树中任意一个编号为 i 的结点，其父节点的编号是多少？其子节点的编号各是多少?
- （1） k^{i-1} 。
- （2） $k^0+k^1+ \cdots\cdots +k^{t-1} = (k^t-1)/(k-1)$;

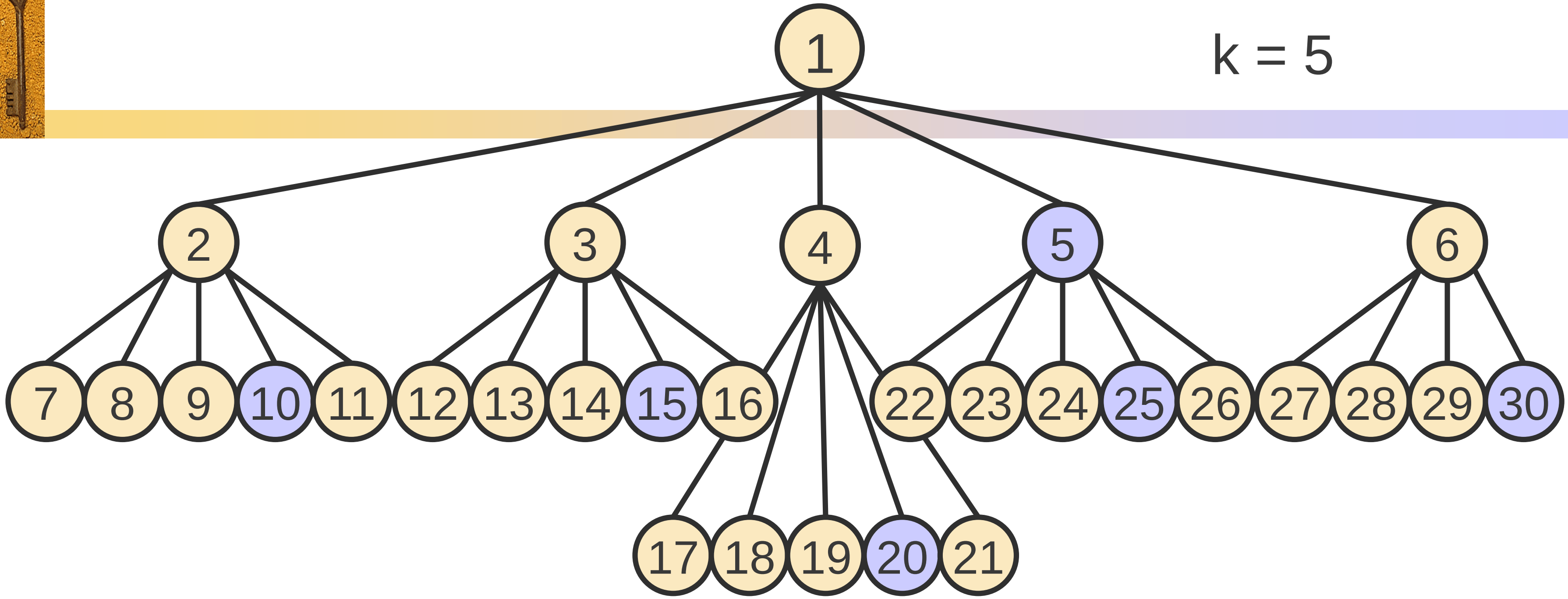
- （3）具有 n 个结点的完全k叉树的深度是多少？
- 设完全二叉树的深度为 t
- 则根据(2)得:
- $(k^{t-1}-1)/(k-1) < n \le (k^t-1)/(k-1)$
- $k^{t-1}-1 < n(k-1) \le k^t-1$
- $k^{t-1} \le n(k-1) < k^t$
- $t-1 \le \log_k n(k-1) < t$
- 又因为t只能是整数，因此， $t=\lfloor \log_k n(k-1) \rfloor + 1.$





i的父节点：若i=1，则无父节点，否则$\lceil (\boldsymbol{i}-\mathbf{1})/\boldsymbol{k} \rceil$

i的子节点：第k-1个子节点i*k；第k个子节点 i*k +1



第k-1个子节点i*k； 第k个子节点 i*k +1

第一个子节点：i*k-(k-2)

第j个子节点：i*k-(k-j-1)

若i*k-(k-2) > n 则节点i没有子节点；

若i*k-(k-j-1) = n 则节点i只有j个子节点

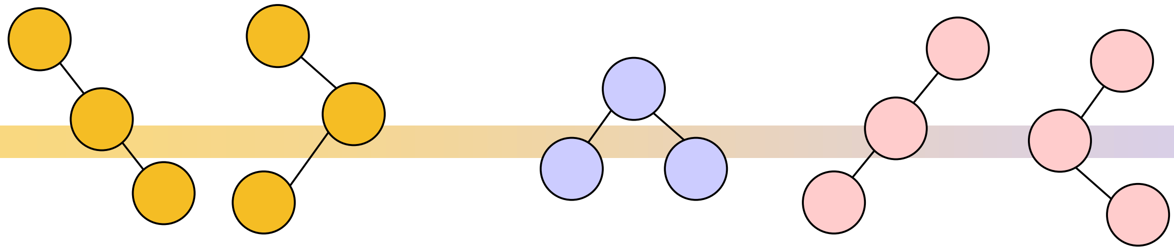
- 2. 请回答在下面的树中，各有多少叶子节点：
- （1）含有n个节点的正则k叉树：即只有度为k和度为0的节点的树。
- （2）含有n个节点的完全二叉树。

- （1）因为$n = n_0 + n_k$，
- 树的分支数$b = k*n_k = n – 1$
- 所以 $n_0 = n - n_k = n – (n-1)/k$

- （2）对应完全二叉树而言，度为1 的节点只能是1或0。根据二叉树的性质：
- $n0 = n2+1 \rightarrow n2=n0-1$
- $n = n0 + n1 + n2 = n0 + n1 + (n0 – 1) = 2n0 + n1 - 1$。
- 所以$n1 = 0$时，$n0 = (n+1)/2$
- $n1 = 1$时，$n0 = n/2$
- 又n1为0←→n为奇数，n1为1←→n为偶数，所以：
- n为奇数时，$n0 = (n+1)/2$
- n为偶数时，$n0 = n/2$

- 树的计数问题（不考虑结点的数据的异同）
- （1）有n个节点且高度为n的二叉树的数目有多少？
- （2）有n个结点的不同形态的树有多少种？

- （1）2^{n-1}
- （2）问题等价于有n个结点互不相似的二叉树的数目有多少种。
- 设所求结果为bn
- b0=1, b1=1, b2=2
- b3=？



- b0=1, b1=1, b2=2
- b3=b0*b2+b1*b1+b2*b0=5
- 同理：
- b4=b0*b3+b1*b2+b2*b1+b3*b0 =
- b5= b0*b4+b1*b3+b2*b2+b3*b1+b4b0

- b0=1， b1=b0*b0=1,
- b2=b0*b1+b1*b0=2,
- 递推关系$b_n = \sum_{i=0}^{n-1} b_i * b_{n-i-1}$

- 回答下列问题：
- 1）请画出满足下列条件的二叉树
- 先序序列为**EBADCFHGIKJ**
- 中序序列为**ABCDEFGHIJK**。
- 2）已知一棵二叉树的中序和后序序列如下：
- 中序序列：**GLDHBEIACJFK**
- 后序序列：**LGHDIEBJKFCA**
- （1）请画出这棵二叉树，并转换为对应的森林。
- （2）给出该森林的后根遍历序列。

- 已知一棵二叉树的中序遍历序列和层次遍历序列，请构造出该二叉树，并写一个算法实现这个过程。
- 中序序列：**GLDHBEIACJFK**
- 层次序列：**ABCDEFGHIJKL**

- 假设二叉树中共有n个节点，分别编号为1，2…，n，现在用两个数组L[i]和R[i](i=1,…,n)作为二叉树的存储结构。L[i]和R[i]分别指示二叉树中第i个结点的左孩子和右孩子结点，0表示空。试写判别结点u是否是结点v的子孙的算法。

```
Status descendent(int L[], int R[], int u, int v)
{
    if (u && v) {
        if (L[v]==u || R[v]==u) return TRUE;
        else if (descendent(L, R, u, L[v])) return TRUE;
        else return descendent(L, R, u, R[v]);
    }
    else return FALSE;
}
```

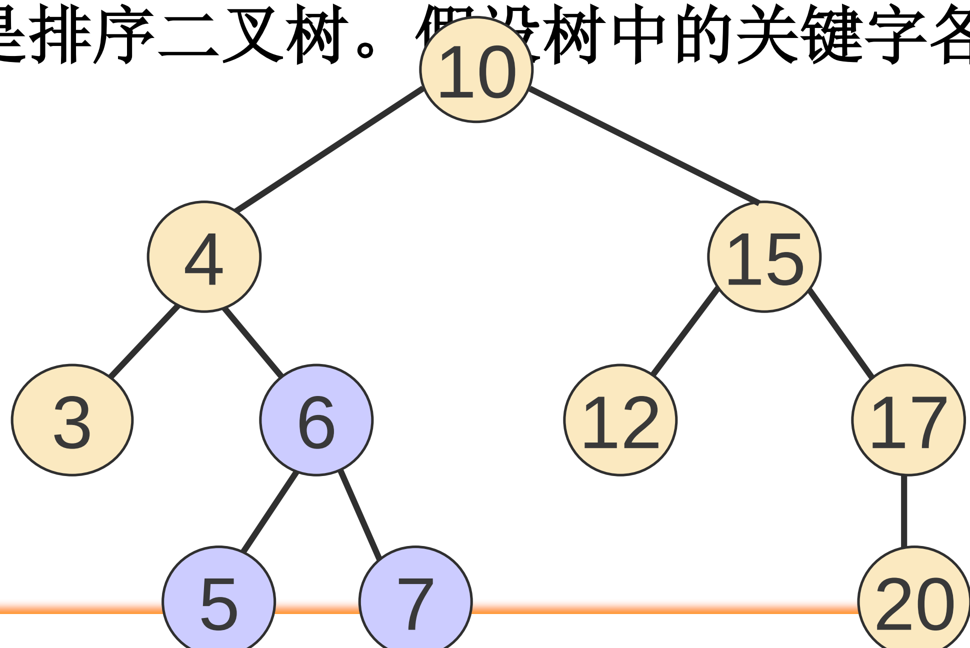
- 假设二叉树以二叉链表的形式存储。编写递归算法：对于二叉树中每一个元素值为x的结点，删去以它为根的子树，并释放相应的空间。

- **假设二叉树以二叉链表的形式存储。编写算法，求二叉树T中结点p和q“最靠近”的共同祖先.**

- 已知一棵具有n个结点的完全二叉树被顺序存储于一维数组的A[1]～A[n]元素中，试找出编号为i的结点的双亲和所有孩子。假设每一个元素用一个整数表示。完成下列程序：

```
void Request(int A[], int n, int i)
  {    //从数组A中打印出编号为i的结点的双亲和孩子
         if( i>n || i<1 )   exit(1);
         printf("current element:%f", A[i]);
         int j=i/2;
         if( j>0 )        printf( "parent:%f" , A[j] );
         else      prinf("No parent!");
         if( 2*i<n ){
              printf("left child:%f", A[2*i] );
              printf("right child:%f", A[2*i+1] );
          }
         else if(   2*i==n   ){
              printf("left child:%f", A[2*i] );
              printf("no right child!";
              }
         else printf("no children!");
  }
```

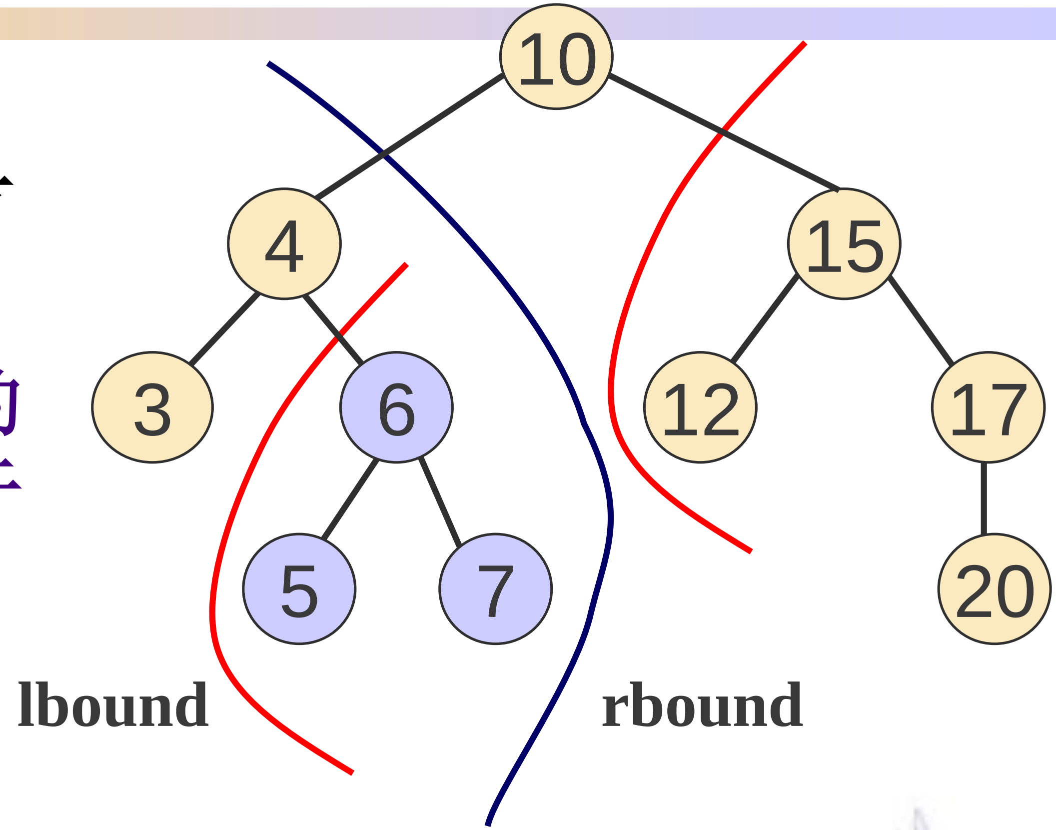
- 排序二叉树是指对于或者是一棵空树；或者是具有如下特性的二叉树：
    1. 若它的左子树不空, 则左子树上所有结点的值均小于根结点的值；
    2. 若它的右子树不空, 则右子树上所有结点的值均大于根结点的值；
    3. 它的左、右子树也都分别是二叉排序树。
- 图是一个排序二叉树的实例。请写一个算法判断给定的二叉树是否是排序二叉树。假设树中的关键字各不相同。



- 算法1：
  - 对二叉树进行中序遍历，将结果保存在一个数组中，然后判断该数组是否是非递减的序列。
- 算法2：（借用线索化二叉树的思想）
  - 对二叉树进行中序遍历，遍历过程中始终记录当前节点n的pre节点。因此只需要判断n的值是否小于pre的值即可。
- 算法3：判断条件改为：对于当前节点n：
  - 若左子树非空，则n的值要大于其左子树中的所有节点的值；
  - 若右子树非空，则n的值要小于其左子树中的所有节点的值；

方法3

- 当前节点n的值要大于其左子树中的所有节点的值；
  - 等价于左子树中的所有节点的值小于n的值
  - 即n的值为其左子树的右界；



n的值为其左子树的右界； 左子树继承节点n的左界；

n的值为其右子树的左界； 右子树继承节点n的右界；

```
Boolean  IsBST (BiTree T, int lbound, int rbound){ // 判断T是否是二叉排序树
    if ( !T )    return 1;
    if(T->data < lbound || T->data>rbound) return 0;
    if (T->lchild)
          if (! IsBST(T->lchild, lbound, T->data)) return 0;
    if (T->rchild)
          if ((! IsBST(T->rchild, T->data, rbound)) return 0;
    return 1;
}// IsBST
```

- 假设树用孩子-兄弟链表表示，请编写一个算法求树中叶子节点的个数。
- 解：树中的叶子节点是没有子节点的，所以在对应二叉树中没有左子树。因此只需要遍历二叉树，对没有左子树的节点计数即可。