期中考试讲解

填空题

填空题

- 1、算法“int i=0, s=0; while (s<=n) { i++; s=s+i; }”的时间复杂度为*O*(_____)。
- 答：由于S变化是1+2+3+……+k = k(k+1)/2

  而i变化是1+1+1+…… = k

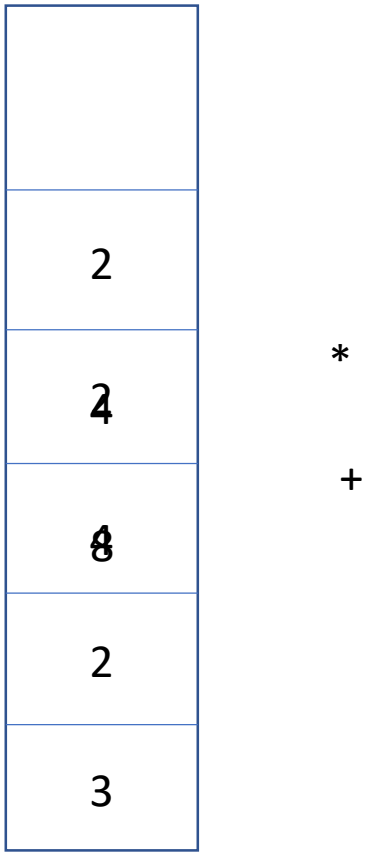
  s变化比i快，所以用s会提前终止循环。

  循环次数是k(k+1)/2=n

  解得复杂度是O($\sqrt{n}$)

- 2、表达式3*2^(4+2*2-6*3)-5求值过程中当检查到6时，数据栈栈顶的数据值为_____。
- 解：首先把表达式转化成逆波兰后缀表达式

$$32422*+63*-^*5-$$



- 3、单循环链表表示的队列长度为*n*，若只设头指针，则进队的时间复杂度为_____。
- 答：只设头指针

因为进队是从队尾进入，而只知道头指针，想找到尾指针的时间复杂度是O(n),找到之后插入操作复杂度是O(1)所以进队复杂度是O(n)

- 4、已知广义表*LS*=(*a*, (*b*, *c*, *d*), *e*)，假设求表头操作为*Head*，求表尾操作为*Tail*，则*b*=_____。
- 答：广义表第一个元素为表头，其余元素组成的表为表尾。
- 1)　tail(LS)=((b,c,d),e)
- 2)　tail(tail(LS)) = (b,c,d)
- 3)　head(tail(tail(LS)))=b

- 5、已知三对角矩阵*A*[1..9, 1..9]的每个元素占用2个单元，现将其三条对角线上的元素逐行存储在起始地址为1000的连续内存单元中，则元素*A*[7][8]的存储地址为_____；如按列优先顺序进行存储，则*A*[7][8]的存储地址为_____。
- 答：由于矩阵较小，方法一可以用数的办法。
- 对于本题从1开始计数的三对角矩阵，共有2+（i-2）*3+j-i+2=2*i+j-2个数据需要存储，存储地址为2*i+j-3
- A[7][8]=1000+2*19=1038
- A[8][7]=1000+2*20=1040

- 6、已知完全二叉树的第7层有10个叶子节点，则整棵二叉树的结点数最多是_____。
- 答：根节点为0层

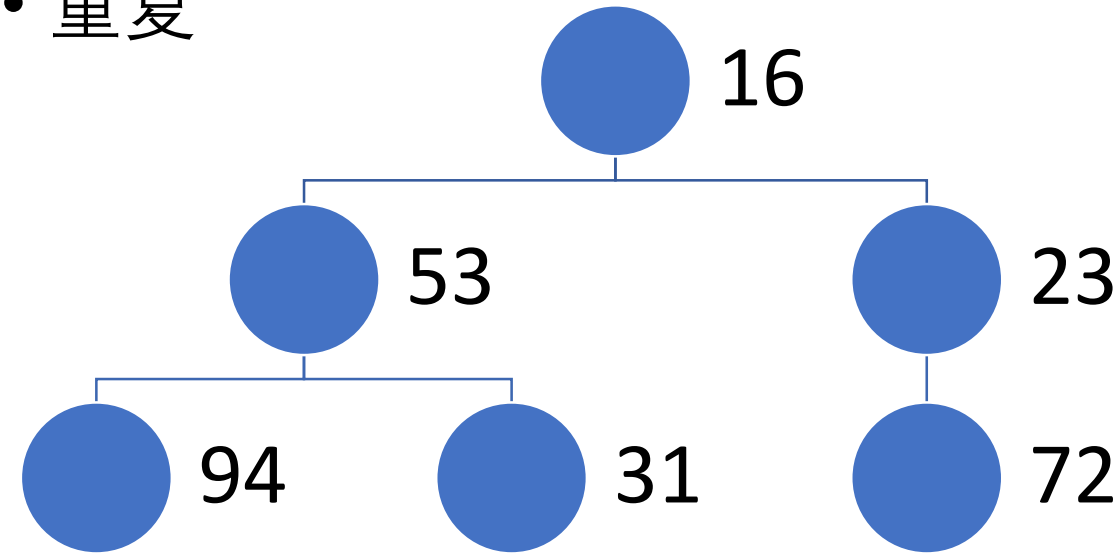
节点数最多的情况是，第七层全满，第八层最后缺少20个结点。

$$2^{7-0+1} + 2^{8} - 1\text{-}20 = 491$$

- 7、一棵二叉树的中序遍历序列为*ABCDEFG*，后序遍历序列为*GFEDCBA*，则这棵二叉树根结点左子树中的结点数目为_____。
- 答：后序遍历最后一个为根节点，所以根节点是A，中序遍历根节点将树分为左右两个子树。*ABCDEFG，所以可以看出根节点左子树节点数目为0*

- 8、将一个有*n*个结点的森林用左孩子-右兄弟链表示时，其中空链域的个数为_____。
- 答：n+1，只跟二叉树有关，跟什么二叉树没有关系

- 9、将关键码序列{16, 53, 23, 94, 31, 72}转换为最大堆后所对应的关键码序列为_____。
- 答：从（n/2）处开始进行判断，如不满足进行交换，交换下去的结点递归判断是否满足最大堆，如都满足则，n=n-1.
- 重复



算法填空

算法填空题

- restore函数用于重构树
- postorder函数用于生成后序遍历序列
- 答案：
- 1)　*ppos
- 2)　rpos=ipos
- 3)　rpos-ipos
- 4)　ipos
- 5)　ppos+1

```
TNODE *restore(char *ppos, char *ipos, int n) {
    TNODE *ptr;
    char *rpos; int k;
    if (n<=0)    return NULL;
    ptr->info=(1)_____;
    for ((2)_____; rpos<ipos+n; rpos++)
      if (*rpos==*ppos) break;
    k=(3)_____;
    ptr->llink=restore(ppos+1, (4)_____, k );
    ptr->rlink=restore ((5)_____+k, rpos+1, n-1-k);
    return ptr;
}
```

问答题

问答题2

- 设目标串*T*=“*xxyxxxyxxxxyxyx*”，模式串*P*=“*xxyxy*”。如何用最少的比较次数找到*P*在*T*中第一次出现的位置？相应的比较次数是多少？请给出具体的分析过程。（6分））
- 答：首先写出next数组[-1,0,1,0,1]
- *x x y x x x y x x x x y x y x*
- *x x y x y* *5*
- *x x y* *2*
- *x x y x y* *4*
- *x x y* *2*
- *x x y* *2*
- *x x y x y* *4*

标准答案

设目标串 T="xxyxxxyxxxxyxyx"，模式串 P="xxyxy"。如何用最少的比较次数找到 P 在 T 中第一次出现的位置？相应的比较次数是多少？请给出具体的分析过程。（6 分））

答： 穷举模式匹配算法的时间复杂度为 $O(m*n)$（其中，m 为模式串长度，n 为目标串长度）。KMP 算法有一定改进，时间复杂度达到 $O(m+n)$。本题也可采用从后面匹配的方法，即从右向左扫描，比较 6 次成功。另一种匹配方式是从左往右扫描，但是先比较模式串的最后一个字符，若不等，则模式串后移；若相等，再比较模式串的第一个字符，若第一个字符也相等，则从模式串的第二个字符开始向右比较，直至相等或者失败。若失败，模式串后移，再重复以上过程。按这种方法，本题比较 18 次成功。

算法二

Total = 18次

x	x	y	x	x	x	y	x	x	x	x	y	x	y	x
				bf										
					bf									
		bf				bt								
							bf							
								bf						
									bf					
										bf				
							b	b	bf		bt			
												bf		
									b	b	bt	bf	bt	

问答题1

某算法时间复杂度的递推关系如下，其中 n 为求解问题的规模，设 n 是 3 的正整数幂。请给出该算法时间复杂度的大 O 表示法及推导过程。（6 分）

出错的同学较多

2、最后结果错误

解题方法：
递归树法
*主方法

$$T(n)=\begin{cases}1 & (n=1)\\ 5T(n/3)+n & (n>1)\end{cases}$$

答：$T(n)=5T(n/3)+n=5(5T(n/9)+n/3)+n=\ldots$

$=5^{\log_3 n}+5^{\log_3 n-1}*3+5^{\log_3 n-2}*3^2+\ldots+5*n/3+n$

$T(n)*5/3=5/3*5^{\log_3 n}+5^{\log_3 n}+5^{\log_3 n-1}*3+\ldots+5*n/3$

$T(n)*5/3-T(n)=5/3*5^{\log_3 n}-n$

$T(n)=(3/2)(5^{\log_3 n+1}/3-n)=O(5^{\log_3 n}-n)$

问答题3

利用 n 个从左到右顺序排列的容量足够大的栈，初始时有 2^n 个不同元素位于这 n 个栈的左侧限定位于 n 个栈左侧的 2^n 个不同元素是按顺序进入 n 个栈，从 n 个栈出来后则在右边按出栈顺序排列。证明通过一系列的进栈和出栈操作，要求操作的方向从左向右，最终在这 n 个栈的右边可以形成这 2^n 个元素的 $2^n!$ 种不同的元素排列顺序。（提示：利用数学归纳法）（6 分）

本题问题不大。90%的同学都做对了，部分同学被扣分，是因为没有描述清楚，部分同学留白。

1、只要证明产生的输出序列是输入序列中的元素的任意排列组合即可。

2、n = 1 时显然成立

3、假设n=k+1时，元素有$2*2^k$个，先把前面2^k个入栈，这时候可以先让第一个栈中的元素通过剩下的n个栈，共有$2^k!$种不同的元素排列顺序。

4、也可以先让不在第一个栈的元素通过剩下的n个栈，也有$2^k!$种不同的元素排列顺序。

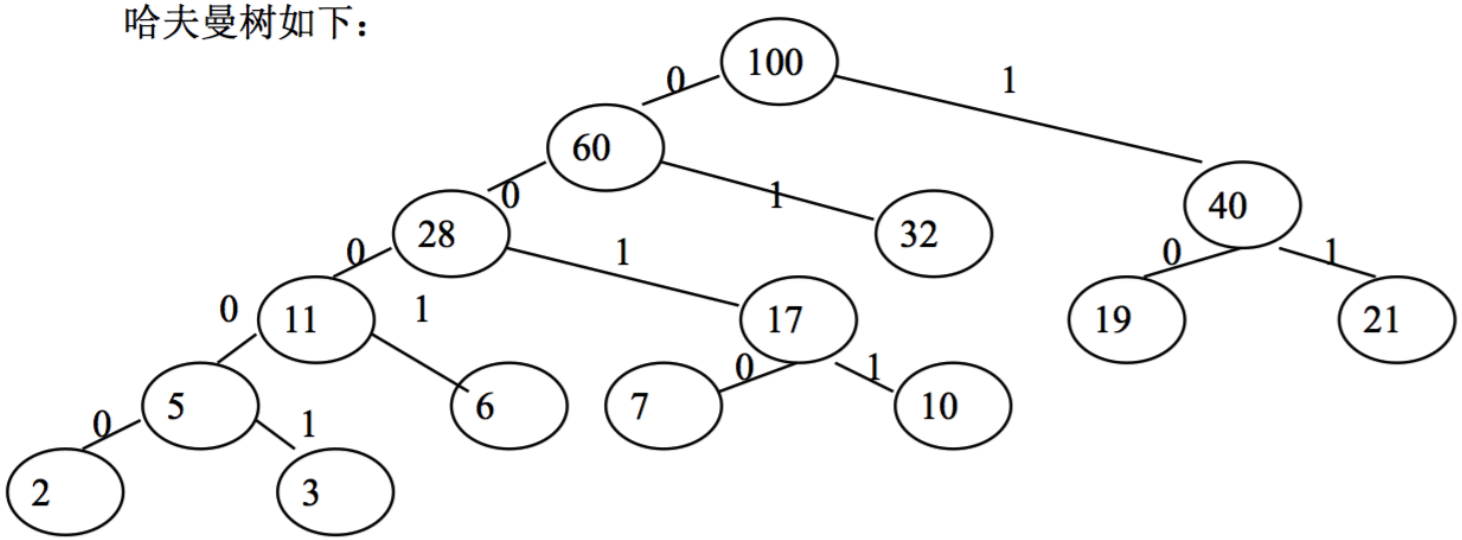
5、这两个$2^k!$种不同的排列相互之间可前可后也可以交错形成任意顺序，所有共有$(2*2^n)!=2^{n+1}!$种不同的排列。

问答题4

用于通信的电文由 8 个字符组成，字符在电文中出现的频率分别为 0.07, 0.19, 0.02, 0.06, 0.32, 0.03, 0.21, 0.10。试为这 8 个字符设计哈夫曼编码。使用 0~7 的 3 位等长二进制表示形式是另一种编码方案，对于上述实例比较两种方案的优缺点。（6 分）

答：哈夫曼编码方案的带权路径长度为 2.61，而用 0~7 的 3 位等长二进制表示形式的路径长度为 3。显然前者降低了信息编码的平均长度，可以提高信道利用率，提高信息传输速度，节省存储空间。两种编码的对照表如下（字符的出现频率转换成整数）：

哈夫曼树如下：



频数	7	19	2	6	32	3	21	10
哈夫曼	0010	10	00000	0001	01	00001	11	011
等长码	000	001	010	011	100	101	110	111

哈夫曼 缺点

- 第一，每一个树的节点都要储存有关它们的父节点与子节点等等相关资讯，如果符号集合的数量包含许多不同机率的符号，记忆体的负荷量会明显增大许多。
- 第二，哈夫曼树的追踪需要耗费极大的运算量。

- 所以基于以上两个论点，传统的哈夫曼编码是一种极为消费储存空间且没有效率的方式。

问答题5

已知指向二叉树中某结点的指针 p，并且已知该二叉树为中序线索化二叉树，规定二叉树的中序第一个结点的左孩子指针为空，最后一个结点的右孩子指针为空。请问仅根据这些已知条件什么情况下能找到 p 的双亲结点，什么情况下无法找到？（6 分）

- 主要存在的问题
  - 20%左右的同学没做
  - 大部分同学作答不全(两条路)

1、先检查p的左孩子的指针，再检查左孩子的左孩子，一直到遇到线索，如果线索指针为空，说明到底第一个结点，则需要从p的右孩子找。

2、如果线索非空，则到达p的一个祖先结点q，先找到q的右孩子r，检查r是否是p，如果是的话则q为p的双亲，否则沿着r的左孩子一路找到p，在p之前找到的就是p的双亲。

3、从p的右孩子找也类似。如果p是树根，从p的左孩子找到第一个结点，从p的右孩子找到最后一个结点，则无法找到p的双亲。

4、因为从根结点沿左子女链一定走到中序序列的第一个结点，沿右子女链一定走到中序序列的最后一个结点，根结点无父节点，因此无法找到。

算法设计题

算法设计1

- 一个数组中存放着未经排序的一组正整数，设计算法调整数组中的数据的顺序，使得所有奇 数存放在所有偶数之前。要求尽量少的辅助空间和尽量快的速度。

要求:(1)描述算法的基本思想并分析时间和空间复杂度;(2)给出程序代码。(10 分)

答:(1)一遍扫描，用类似于快排的交换过程。时间复杂度 O(n)，空间复杂度 O(1)

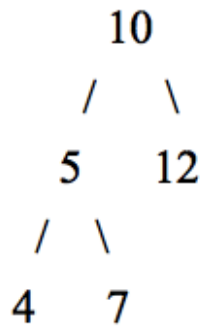
算法设计1

```
void move(int A[ ], int n) {
        int i=0, j=n-1, tmp;
        while (i<j) {
                while (i<j&&A[i]%2==1) i++;
                while(i<j&&A[i]%2==0) j--;
                if(i<j) {
                    tmp=A[i]; A[i]=A[j]; A[j]=tmp;
                    i++;  j--;
                  }
        }
} //算法结束
```

算法设计2

- 输入一个整数和一棵二叉树。从树的根结点开始往下访问一直到叶结点所经过的所有结点形 成一条路径。打印出和与输入整数相等的所有路径。例如，输入整数 22 和如下二叉树:

则打印出两条路径:10, 12 和 10, 5, 7。二叉树结点的数据结构定义为:




```
struct BinaryTreeNode {
    int m_nValue; //value of node
    BinaryTreeNode *m_pLeft; //left child of node
    BinaryTreeNode *m_pRight; //right child of  node
};
```

要求:(1) 描述该算法的基本设计思想;(2) 给出程序代码。(15 分)

算法设计2

- 主要利用回溯和递归，需要一个栈辅助路径的回溯。

```
struct TreeNode {
    int data;
   TreeNode *left;
   TreeNode *right;
};
void printPaths(TreeNode
*root, int sum) {
    int path[MAX_HEIGHT];
    helper(root, sum, path, 0);
}
```

算法设计2

```
void helper(TreeNode * root, int sum, int path[ ], int top) {
      path[top++]=root.data;
       sum-=root.data;
       if(sum>=0) { //剪枝
         if (root->left==NULL && root->right==NULL)
             if (sum==0)  printPath(path, top);
          else {
                    if (root->left!=NULL)
                        helper(root->left, sum, path, top);
                    if (root->right!=NULL)
                         helper(root->right, sum, path, top);
           }
```

算法设计题3

- 在二叉树中查找值为 *x* 的结点，试编写算法打印值为 *x* 的结点的所有祖先(假设值为 *x* 的结 点不多于一个)，并分析算法的渐进时间复杂度(不加分析直接写出结果得零分)。(15 分)

程序设计思想——可以利用栈来解答本题。通过前序遍历树，把路经的结点都放入栈中，如 果遇到结点 *x* 就输出栈里面的元素;否则，利用前序遍历，递归处理其它结点。

算法设计题3

```
class BTree {
  private:
    BTreeNode *root; //树的根结点
 public:
    void ancestor(BTreeNode *current);
    stack S;
 };
```

算法设计题3

```
class BTree {
  private:
    BTreeNode *root; //树的根结点
 public:
    void ancestor(BTreeNode *current);
    stack S;
 };
```

算法设计题3

```
void BTree :: ancestor(BTreeNode * current) {
    if(current= =NULL) return false;
    S.push(root);
    if (root->left= =x)
        { 把栈中的元素依次退出栈,
        作为结果输出;
     } else {
       ancestor (current->left);
      ancestor (current->right);
     }
}
```