**2020年电信学院《数据结构》云考试电子试卷**

1. **考试时间：**5月10日（周日）上午8:30-11:00 （正式考试时长2.5小时，提前半小时进入考场，11:00交卷）
2. **考试方式：**开卷。可以看书，但不能上网，不能用电子书和电子设备。
3. **考试题型：**简答题（共58分）和算法题（共72分），总分130分。
4. **得分规则：**按照规定选择题目作答；期终考试卷面成绩满分130分，不折算，若卷面成绩高于100分则按100分计算。
5. **答题形式：**手写回答，不必抄题，但一定要注明题号。请参考《数据结构答题模板》

如：页眉 班级：电信190X班；姓名：张三; 学号：U2019XXXX

页尾 页码1/12

最后一题答完后，增加一段-----------------------------end---------------------表示结束

1. **交卷形式：**每张答题纸拍一张照片，命名格式为“班号-姓名-n”（n是每张答题纸的编号，从1开始），文件格式是Jpeg（Jpg），所有文件压缩为rar文件，rar文件命名为“班号-姓名-Z.rar”（Z是试卷总张数）。
2. **交卷方式：11:00**考试结束开始提交答卷，考生须停止作答，把答题纸拍照，考试结束后15分钟之内同时发给主考老师和助教各一份，发送方式为QQ私信传输文件，交卷时间以QQ上文件传输时打的时间戳为准。在考试结束后15分钟交卷者，每延迟一分钟，卷面成绩扣一分；考试结束后25分钟还未交卷者，本次考试成绩无效。

**数据结构考试题目**

**2020.5.10**

**【简答题 ：共58分。】**

**简答题共8题，分数安排如下：**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **章节** | **1** | **2** | **3** | **4,5** | **6** | **7** | **9** | **10** | **总分** |
| **满分** | **6** | **8** | **8** | **8** | **8** | **6** | **8** | **6** | **58** |

**【第1章：绪论（满分6分）】**

**【题号：JD-1】**

1. 通过数据结构的学习，你在算法设计方面的最大收获是什么？请说明理由。
2. 以下算法的时间复杂度为多少？ 【王邦，答案 O(n2log5n】

void fun(int n){

int i,k;

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

for(j=1; j<=n; j++) {

k=1;

while (k<=n)

k=5\*k;

}

}

**【第2章：线性表（满分8分）】**

**【题号：JD-2】**

阅读下列程序，回答问题：

LinkList mynote(LinkList L) {//L是不带头结点的单链表的头指针

if(L&&L->next){

q=L；L=L－>next；p=L；

S1： while(p－>next) p=p－>next；

S2： p－>next=q；q－>next=NULL；

}

return L；

}

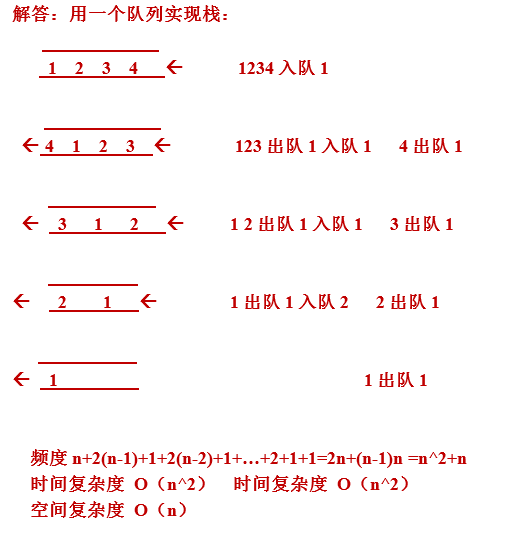
请回答下列问题：

1. 说明语句S1的功能； 【查询链表的尾结点】
2. 说明语句组S2的功能； 【将第一个结点链接到链表的尾部，作为新的尾结点】
3. 设链表表示的线性表为（a1,a2, …,an），写出算法执行后的返回值所表示的线性表。【返回的线性表为（a2,a3,…,an,a1）】

**【第3章：栈和队列 （满分8分）】**

【**题号：JD-3**】

以1，2，3，4顺序输入，如何得到4，3，2，1为例，说明如何用一个队列实现栈？给出以上操作的时间、空间复杂度。



**【第4，5章：串、数组和广义表（满分8分）】**

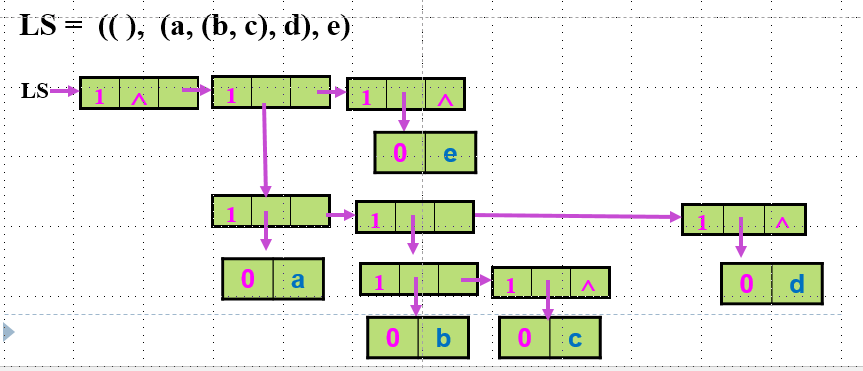
【**题号：JD-4**】

1. 在KMP算法中，求子串bababbbababb 的next数组；

【答案011234223456】

1. 画出广义表LS=((), (a, (b,c),d), e)的头尾链表存储结构。

解答：



**【第6章：树与二叉树（满分8分）】**

【**题号：JD-5**】

1. 完全二叉树的结点总数为2020个，那么，这颗树有多少条边（即结点之间的连线）？叶子结点、度为1的结点和度为2的结点各有多少个？

**【答案：边有2019条，叶子结点，度为1的点，度为2的点各有2018/2+1，1，2018/2个】**

1. 设p是二叉树的根结点，下面算法功能是将二叉树所有左右子树交换。请问有何错误？若有错，请改为正确的算法, 并分析算法的平均时间和空间复杂度。

void Exchange（bitreptr p）{

while p≠NULL {

Exchange (p->Lchild);

x=p-> Lchild; p-> Lchild =p->Rchild; p-> Rchild=x;

Exchange (p->Rchild);

}

}

答： 正确的算法应为：

ABC（bitreptr p）

{ if p≠NULL then

{ x=p-> Lchild; p-> Lchild =p->Rchild; p-> Rchild=x;

Exchange (p->Lchild);

Exchange (p->Rchild);

}

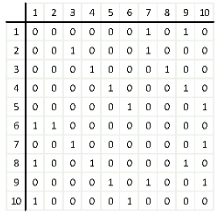
}

该算法的时间复杂度是O(n)，空间复杂度是O(logn)

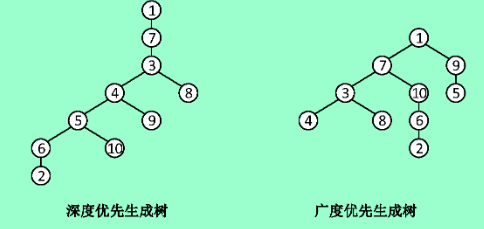
**【第7章：图（满分6分）】**

【**题号：JD-6**】

已知以二维数组表示的图的邻接矩阵如下图所示。试分别画出自顶点1出发进行遍历所得的深度优先生成树和广度优先生成树。



解答：



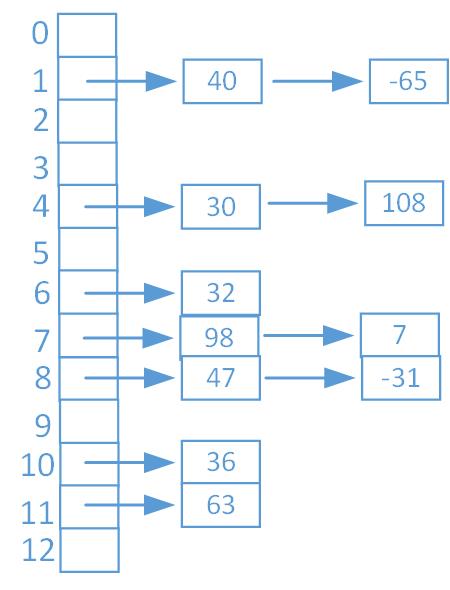
**【第8章：查找，满分8分】**

**【题号：JD-7】**

已知一组关键字为（7，36，32，108，-31，30，-65，47，40，63，73，98，60），设哈希函数H（key）＝key Mod 13，Mod表示取余。用链地址法处理冲突。

1. 请画出所构造的哈希表。
2. 在等概率情况下，求出其查找成功时的ASL。

**解答：构建哈希表如下：**

****

60

73

**等概率情况下平均查找长度为：**

**ASL=(7\*1+4\*2+3\*1+4\*1)/13=22/13**

**【第9章: 排序，满分6分】**

**【题号：JD-8】**

设有10万个无序的元素，需排出前10个最大的元素，请问用什么排序方法最快？排出前10个最大元素的总比较次数是多少？

解答：

用竞标赛排序最快，排除最大元素比较次数为100000-1=99999，第2-10个元素比较次数为,因此，总的比较次数为：99999+9\*（）

**【算法题：每题12分，共72分】**

**所有算法题都要求设计尽可能高效的算法，即时间、空间复杂度尽可能低。算法题解答要求分为三个部分：**

* 1. **解题思路（中文描述）**
  2. **算法流程（类C代码描述）**
  3. **时间复杂度（大O表示，加上适当解释）**

**【题号：SF-1】**

**题干：**假设有2个班级，每个班级都有N个学生，2个班学生的数据结构成绩表（分数表）分别放在两个数组X1和X2中， X1[n]、X2[n]为0到100之间的整数,0nN。请设计一个高效算法，找到2个班级的成绩表中共同出现且出现频率最高的所有分数。例如：

X1 = [75, 80, 68, 91, 88, 91]

X2 = [91, 66, 81, 81, 81, 81]

2个班级的成绩表中共同出现且频率最高的分数为91。

**问题：**（1）叙述算法思路；（2）给出算法的伪代码；（3）分析算法的时间复杂度和空间复杂度。

解答：

思路是利用hash函数，将分数作为地址，然后扫描两个班的学生分数，每次对应的分数的地址加一，得到两个数组，再次扫描这两个数组，相同地址的值相加，满足和最大的，并且两个数组的值都不为零的，即是说要找的分数。

**【题号：SF-2】**

**题干：**对于输入序列1，2，…，n，n>2，n为整数,使用栈通过任意的进栈和出栈次序，可以得到的输出序列为T1 T2 … Tn 。例如输入序列123，输出序列为 123、132、213、231、321共5种**。**

**问题：**设计一个算法输出第二个元素是2的所有可能的序列（如上例，输出序列为123、321两种），并分析该算法的时间复杂度和空间复杂度。

解答：

算法如下：

OutputStack(String t, int i, int j)

//输出序列t中的第i个到第j个的元素

{

if (j > i) print("\n"); // 换行打印

else

{

e = GetElem(t,i); // 取t中的第i个元素

push(s, e);

OutputStack(t, i+1, j);

if (s栈不空)

{

pop(s, e);

print(e);

OutputStack(t, i+1, j);

}

}

}

时间复杂度：O(n) =O( n！）＝ O(2n)。

空间复杂度：由于需要栈保存递归参数，因此空间复杂度是递归操作中栈的深度，故为O(n)。

**【题号：SF-3】**

**题干：**有N把锁和N把钥匙，N>1，只有对应的钥匙能开对应的锁。锁和钥匙按顺序编号，第一对锁和钥匙是(0, 1)，第二对锁和钥匙是(2, 3)，以此类推，最后一对锁和钥匙是(2N-2, 2N-1)。钥匙和锁都随机的放置在连续的2N个位置上，位置用0到2N-1的整数表示，要想将锁打开只能是钥匙和锁处在相邻的位置上，为了让锁和钥匙能相邻使得锁被打开，可以一次选择任意交换两个位置上的锁或者钥匙，即让它们交换位置。补充说明：

1. 数组lock[2N]存放锁和钥匙的序号，这些钥匙和锁的初始位置是随机的；
2. 数组lock是序列 0..2N-1 的一个全排列。

**问题：**计算最少交换位置的次数，以便每个锁都被打开。

**示例1：**

若: lock = [3, 0, 1, 2] 则需要1次调整

解释: 这时需要交换lock[0]和lock[2]的位置才能将所有锁打开。

**示例2：**

若: lock = [1, 0, 3, 2] 则需要0次调整

解释:所有的锁已经都可以打开了。

解答：

1.首先初始状态把这N对情侣分别构成一个连通块

2.考虑k对情侣相互错开的情况，他们形成一个环，可以知道需k-1次交换使排列正确

3.这样相互错开的情况，分别构成连通块，最后用N - 连通块个数即为答案

时间复杂度O(n)。

class Solution {

public int minSwapsCouples(int[] row) {

int n=0;

//只需要遍历下标为偶数的座位，每次+2，奇数的座位可由交换座位匹配好

for(int i=0;i<=row.length-2;i+=2){

int j;

//判断下标i的情侣是多少

j=row[i]%2==0?row[i]+1:row[i]-1;

//若下一个座位的不是该情侣

if(row[i+1]!=j){

//需要进行交换座位

n++;

//找到该情侣，进行交换

for(int m=i+2;m<row.length;m++){

if(row[m]==j){

swap(row,m,i+1);

break;

}

}

}

}

return n;

}

//交换

private void swap(int[] a,int i,int j){

int tmp=a[i];

a[i]=a[j];

a[j]=tmp;

}

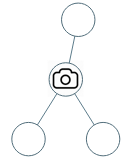
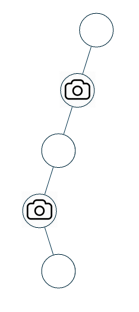
}

**【题号：SF-4】**

**题干：**给定一个二叉树，我们希望能看到所有的结点，方式是在树的某些结点上安装探测器（安装探测器的结点不影响此结点的任何功能）。结点上的每个探测器都可以看到其父结点、自身及其直接孩子结点。

**问题：**给定一颗二叉树，设计一个高效算法，计算覆盖该树所有结点所需的最小探测器数量，使得所有的结点都可以被看到。分析算法的时间复杂度。

**示例：**（下图中4个结点的二叉树安装1个探测器，5个结点的二叉树安装2个探测器）

**解答：**当遍历到一个节点时，我们可以定义三种状态：

0 ： 初始状态，如果节点为null可以返回，也就是不影响其他节点，当两个节点都是0时，我们直接设置当前节点为未监控状态

1： 未监控状态，如果子节点含有该状态，则此节点必须添加摄像头，同时返回当前状态为监控态

2： 监控态，表明此节点已经被监控，当子节点为此状态时，父节点不需要添加摄像头，可以返回初始态

**代码：**

**Int traveltree(TreeNode node)**

**{**

**If(node == null) return 0;**

**Int l,r;**

**l= traveltree(node.left);**

**r= traveltree(node.right);**

**if(l+r==0) return 1;**

**else if (l ==1 || r==1)**

**{**

**Cameras ++;**

**Return 2;**

**}**

**else return 0;**

**时间复杂度为O(n)，空间复杂度为O(h)，h为树的高度，近似为log2(n)**

**【题号：SF-5】**

**题干：**现有一个打印机要在一张白纸上打印物体，可以将白纸划分为二维的网格，打印过的格子显示黑色，记为1，没有打印过的格子显示白色，记为0。打印完成以后我们定义所有被白色格子包围的黑色格子形成一个物体，并且每个物体只能由水平方向或竖直方向上相邻的黑色块连接形成。此外，假设该纸张的四条边外均是白色的。

**问题：**设计一个高效算法，给出任意指定的“1/0”矩阵表示的二维网格中物体的数量。

|  |  |
| --- | --- |
| **示例1：** | **示例2** |
| 输入:  11100  11100  11000  01110 | 输入:  11110  11000  00110  11000 |
| 输出: 1 (个物体) | 输出: 3 (个物体) |

解释: 每个物体只能由水平和/或竖直方向上相邻的黑色格子连接而成。

解答：我们需要定义一个数组visited来判断当前的元素是否已经被访问了，如果被访问了则为true,如果没有被访问则为false. 那么我们判断当前元素是否为1，如果为1，则看它的相邻位置是否为‘1’，一直到把所有连通域里面的‘1’访问结束后，岛屿个数+1

public class IslandCount {

public static int dfs(int[][] island,int[][] isvisit) {

int count = 0;

for (int i = 0; i < island.length; i++) {

for (int j = 0; j < island[0].length; j++) {

if(isvisit[i][j]!=1) {

if(island[i][j]==1) {//表示是第一个1，进行加一

count++;

}

//进行深度优先搜索

dfsVisit(island,isvisit,i,j);

System.out.println("---------------");

}

}

}

return count;

}

public static void dfsVisit(int[][] island,int[][] isvisit,int i,int j) {

if(i<0 ||j<0 || i>island.length-1 || j>island[0].length-1) {

//表示已经数组越界了

return;

}

if(island[i][j]==0) {

//表示不是岛屿

return;

}

if(isvisit[i][j]==1) {

//已经遍历过了，

return;

}else {

//表示没有遍历过

isvisit[i][j]=1;

System.out.println("i="+i+" j="+j);

dfsVisit(island,isvisit,i-1,j);//上

dfsVisit(island,isvisit,i+1,j);//下

dfsVisit(island,isvisit,i,j-1);//左

dfsVisit(island,isvisit,i,j+1);//右

}

}

public static void main(String[] args) {

//岛屿的数量

int[][] island = {{1,1,0,1,1},

{1,1,0,0,0},

{0,0,1,0,0},

{1,0,0,1,1}};

//isvisit表示是否遍历过

int[][] isvisit =new int[island.length][island[0].length];

System.out.println(dfs(island,isvisit));

}

}

时间复杂度O(m\*n)，m，n为二维网络的长和宽

**【题号：SF-6】**

**题干：**给定如下的m×n的数字方形，第一行和第一列元素都是1，其他元素等于其上面元素加上左边元素和左上角元素，如：

1 1 1 1 1

1 3 5 7 9

1 5 13 25 41

1 7 25 63 129

**问题：**请设计一个高效算法求出处在第i行第j列的元素的值，给出时间和空间复杂度。

注意：实际的矩阵是不存在的，我们只需计算数字方形处在第i行第j列的元素的值。

**解：**

**算法思想：**

**a[i][j]= a[i-1][j-1]+a[i-1][j]+ a[i][j-1]**

**使用递归法即可。时间复杂度 O(3^mn), O(3^mn)**

**--------------------------The end------------------------------**