# 1-5章试题 ：

* 1. 二维数组M的每个元素是6个字符组成的串，行下标i的范围从0到8，列下标j的范围从1到10，则存放M至少需要\_\_\_540\_\_\_个字节；M的第8列和第5行共占\_\_108\_\_\_\_个字节。

解：每个元素占6个字符，一个字符一个字节，所以每个元素占6个字节；

I：共有9个，j：共有10个

第一个空：9\*10\*6=540

第八列共有9个元素，第五行共有10个元素，但是有一个元素被计算了两遍，所以，第八列和第五行共有18个元素，18\*6=108

* 2. 设有一批数据元素，为了最快地存取某元素，宜用\_\_\_顺序\_\_\_结构存储，为了方便的插入一个元素，宜用\_\_\_链式\_\_\_结构存储．
  3. 设有一个空栈，现输入序列为1、2、3、4、5。经过push，push，pop，push，pop，push，pop，push后，输出序列是\_\_\_\_2 3 4\_\_\_\_\_\_\_\_。
  4. 设输入序列为1、2、3，则经过栈的作用后可以得到\_\_5\_\_\_\_种不同的输出序列。

1. 设F和R分别表示顺序循环队列的头指针和尾指针，则判断该循环队列为空的条件为\_\_\_\_\_\_\_ F == R\_\_\_\_\_，判满的条件：(R + 1) % MAXQSIZE == F。
2. 设有一个n阶的下三角矩阵A，如果按照行的顺序将下三角矩阵中的元素（包括对角线上元素）存放在n(n+1)个连续的存储单元中，则A[i][j]与A[0][0]之间有\_\_\_(1+i)\*i/2 + j \_\_\_个数据元素。

本题目当年的答案是：(1+i)\*i/2 + j 但是有学生考虑A[i][j]与A[0][0]之间应该有(1+i)\*i/2 + j -1 个更合适

1. 设有一个顺序共享栈S[0：n-1]，其中第一个栈项指针top1的初值为-1，第二个栈顶指针top2的初值为n，则判断共享栈满的条件是\_\_\_ top1 + 1 == top2\_\_\_\_\_\_\_\_\_。
2. 不论是顺序存储结构的栈还是链式存储结构的栈，其入栈和出栈操作的时间复杂度均为\_\_O（1）\_\_\_\_。
3. 链表不具备的特点是 （ A ）。
4. 可随机访问任一结点
5. 插入删除不需要移动元素
6. 不必事先估计存储空间
7. 所需空间与其长度成正比
8. 下列哪种数据结构用于执行递归调用（ B ）。

A. 队列 B. 栈 C. 串 D. 链表

1. 二维数组A[5][6]的每个元素占5个存储单元，从首地址LOC(0,0)开始将其按行优先顺序存储在起始地址为1000的连续的内存单元中，则元素A[4][4]地址为（ A ）。

A. 1140 B. 1145 C. 1120 D. 1125

1. 设循环队列中数组的下标范围是1~n，其头指针为f，尾指针为r，则其当前元素个数为（C ）。
2. r-f
3. r-f+1
4. (r-f+n) mod n
5. (r-f) mod n +1
6. 程序段s=i=0；do { i=i+1; s=s+I;}while(i<=n);的时间复杂度为（A ）。

A. O(n) B. O(nlog2n) C. O(1) D. O(n^2)

1. 设指针变量p指向单链表中结点A，若删除单链表中结点A，则需要修改指针的操作序列为（ A ）。
2. q=p->next；p->data=q->data；p->next=q->next；free(q)；
3. q=p->next；q->data=p->data；p->next=q->next；free(q)；
4. q=p->next；p->next=q->next；free(q)；
5. q=p->next；p->data=q->data；free(q)；
6. 设输入序列为1、2、3、4、5、6，则通过栈的作用后可以得到的输出序列为（ B ）。
7. 5，3，4，6，1，2
8. 3，2，5，6，4，1
9. 3，1，2，5，4，6
10. 1，5，4，6，2，3
11. 设某链表中最常用的操作是在链表的尾部插入或在尾部删除元素，则下列、存储方式中最节省运算时间的是（ D ）。
12. 单向链表
13. 单向循环链表
14. 双向链表
15. 双向循环链表
16. 建立一个长度为n的有序单链表的时间复杂度为（C ）

A O(n) B. O(1) C. O(n^2) D. O(log2n)

1. 设指针变量top指向当前链式栈的栈顶，则删除栈顶元素的操作序列为（D ）。



A. top=top+1; B. top=top-1;

B. top->next=top; D. top=top->next;

19.设二维数组A[8][10]从首地址LOC（0,0）开始按行优先顺序存放，每个元素占用 存储

单元，则元素a[6][5]的起始地址为 。

1. 广义表(((a), b), c)的表头是 ((a), b) ，表尾是 (c) ，深度 3 长度 2 。
2. （10分）现有5\*5矩阵，要求根据提示信息使用C语言编写函数isSymmetricMatrix()来判断矩阵是否为对称矩阵，如果是对称矩阵返回TRUE，否则返回FALSE。

|  |
| --- |
| #define TRUE 1  #define FALSE 0  int IsSymmetricMatrix(int M[5][5]); |

#include <stdio.h>

#include <stdlib.h>

#define TRUE 1

#define FALSE 0

int IsSymmetricMatrix(int M[5][5])

{

int i,j;

for(i=0;i<=4;++i)

{

for(j=0;j<=4;j++)

{

if(M[i][j]==M[j][i])

{

printf("此矩阵不是方阵");

return FALSE;

break;

}

}

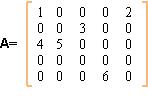
}

printf("此矩阵是方阵");

return TRUE;

}

1. 对于任意稀疏矩阵M,均可压缩存储以减少存储空间的消耗，如三元组顺序存储等。

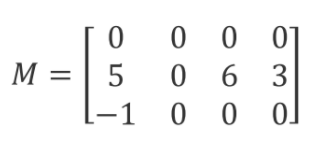


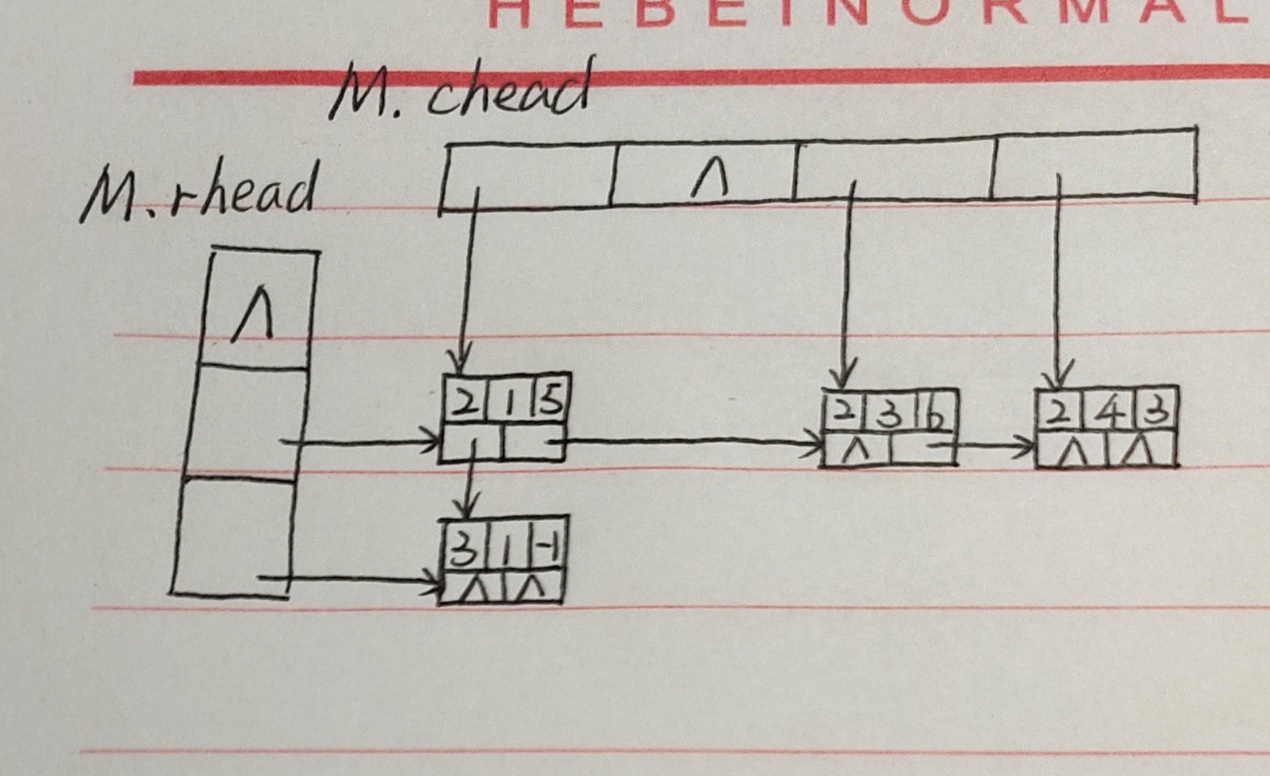
（1）设计并写出稀疏矩阵的存储结构类型，并画出矩阵A相对应的存储示意图。

（2）写出类c语言算法描述，通过键盘输入来实现稀疏矩阵的初始化（注意：矩阵形式类似例子A,但不是针对此特定矩阵A求解）函数原型如下所示：

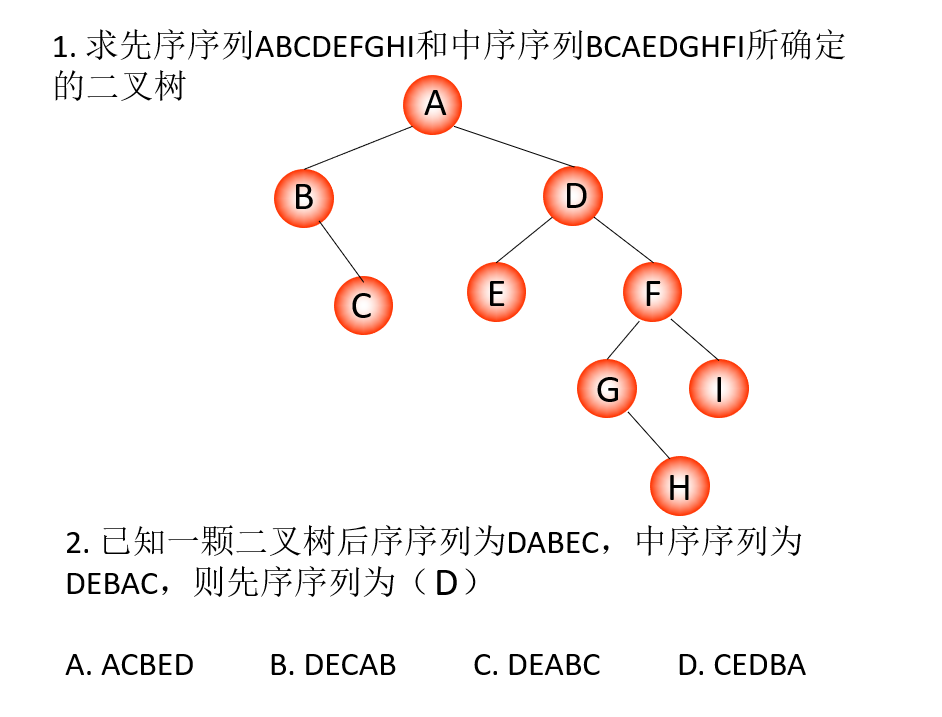
int InitMatrix (Matrix M)；// Matrix 为（1）中定义的类型

1. 矩阵M的**十字链表**存储方式结构图如下：





​第6章 ​​树和二叉树



【例1**】**、已知一棵完全二叉树的第6层（设根为第1层）有8个叶子结点，则该完全二叉树的结点个数最多是（ ）。 【全国统考 2009】

A．39 B. 52 C. 111 D. 119

答案：C

【**解析**】。完全二叉树的形状要在考生的头脑中有个清晰的映像，这样这个题目就比较好做了。题中说第6层有8个叶子，又要求总结点数最多，而完全二叉树的叶子只能在最后两层，所以，叶子除了第6层有8个之外，第七层也有，第七层的叶子是第六层的32-8=24个结点“生”出来的，所以第七层有48个，而前6层最多是26-1=63个，最终选C。

【例2**】**、在一棵度为4的树T中，若有20个度为4的结点，10个度为3的结点，1个度为2的结点，10个度为1的结点，则树T的叶结点个数是（ ）。 【全国统考 2010】

A．41 B. 82 C. 113 D. 122

答案：B.

【**解析**】 二叉树的性质3：n0=n2+1，考生只要会证明这个性质，那么本题也就不难得出答案了。在此不在赘述。

【例3**】**、已知一棵完全二叉树有768个结点，则该二叉树中叶结点的个数是（ ）。

【全国统考 2011】

A．257 B．258 C．384 D．385

答案：C.

【**解析**】 这个题目没有直接考二叉树的五个基本性质，拐了个弯，主要考察考生对完全二叉树的熟悉程度，考生可以回忆一下，完全二叉树主要是由度为2的结点和叶子组成，度为1的结点在完全二叉树中最多有一个。考生可以先假设没有度为1的结点，这样根据二叉树的性质3：n0=n2+1得n2+1+n2=768，这是不可以能的，会出现半个结点，所以假设错误，即该二叉树必然有一个度为1的结点。这样有n2+1+n2=768-1，得n2=383。所以叶结点为384。

### 二、二叉树的存储结构

**【**例1**】**、已知一棵有2011个结点的树，其叶子结点个数为116，该树对应的二叉树中无右孩子的结点个数是（ ）。【全国统考 2011】

A.115 B.116 C.1895 D.1896

答案：D

【解析】 此题较复杂，原因是不符合正常思维，也就是说，考生不能够根据题意画出树和二叉树的对应关系，从而找出通项公式解答。当年考生得分率很低。

如果换种思路就比较简单了，根据树到二叉树的转换规则，树中只有非叶子结点在二叉树中对应的结点肯定有左孩子，换句话说，只有叶子结点在其对应的二叉树中才没有左孩子，其余结点都有左孩子。这样，该树对应的二叉树只有116个没有左孩子的结点，也即有116个结点的左指针域为空。另一方面，该二叉树一共有2011个结点，4022个指针域（2012个为空，2010个非空），所以，右指针为空的个数是：2012-116=1896个，也即没有右孩子的结点个数是1896个。

**【**例2**】**、若一棵二叉树的先序遍历序列是a,e,b,d,c，后序遍历序列为b,c,d,e,a，则根结点的孩子结点（ ）。

【全国统考 2012】

A． 只有e B. 有e、b C. 有e、c D. 无法确定

答案：A

**【解析】**每个考生基本上都知道这个解题模式：根据二叉树的先序序列和中序序列或者后序序列和中序序列画出二叉树，然后在根据二叉树解题。而本题却偏偏不给中序序列。这样考生就只能用分析法寻找正确答案了。

根据先序序列，可以确定这棵二叉树的根结点是a。一方面，该二叉树如果有左子树，则左子树的根是e，但是根据后序序列就会发现，右子树不可能存在了，因为e之后，直接到a了。另一方面，该二叉树如果没有左子树，则e只能是其右子树的根结点了。综合上面分析，只能选答案A。

### 线索二叉树

【例1】、下列线索二叉树中（用虚线表示线索），符合后序线索树定义的是（ ）。

【全国统考 2010】



答案：Ｄ.

**【解析】**这个题目很简单，第一步，写出后序遍历序列；第二步，让左空指针指向后序序列的前驱结点，右空指针指向其后继结点。所以答案D正确。

**【**例1**】**、对n(n大于等于2)个权值均不相同的字符构成哈夫曼树，关于该树的叙述中，错误的是（ ）。 【全国统考 2010】

A. 该树一定是一棵完全二叉树

B. 树中一定没有度为1的结点

C. 树中两个权值最小的结点一定是兄弟结点

D. 树中任一非叶结点的权值一定不小于下一层任一结点的权值

答案：A

4、对于给定的一组权值 W＝{1, 3, 7, 8, 14, 20, 28} 建立哈夫曼树，并计算带

权路径长度。 196

1、一棵哈夫曼树有 19 个结点，则其叶子结点的个数是( 20 )。

### 树的双亲表示法

// **双亲表示法及其实现代码重点看存储结构及其树的深度**

**树的叶子、度求法实现：**

#include <stdio.h>

#include <stdlib.h>

#define Status int

#define OK 1

#define ERROR 0

#define OVERFLOW -2

#define TElemType char

//----------- 树的双亲存储表示 -----------//

#define MAX\_TREE\_SIZE 100

typedef struct //结点结构

{

TElemType data;

int parent; //双亲的位置域

}PTNode;

typedef struct //树的结构

{

PTNode nodes[MAX\_TREE\_SIZE];

int r,n; //根的位置和结点数

}PTree;

// 根据书上的修改了下叶子结点的位置

const int parentArr[] = {-1, 0, 0, 0, 1, 3, 5, 5, 1, 5};

const char dataArr[] = {'R', 'A', 'B', 'C', 'D', 'F', 'G', 'H', 'E','K'};

const int m = 10; //dataArr 数组长度

void CreateTree(PTree &P)

{

int i;

P.n = m;

for(i = 0; i < m; ++i) {

P.nodes[i].data = dataArr[i];

P.nodes[i].parent = parentArr[i];

if(-1 == P.nodes[i].parent)

P.r = i;

}

}

int DepthTree(PTree P)

{

int max = 0, i, j, temp = 0;

for(i = 0;i < P.n; ++i){

// j是双亲的下标，只要 j != -1 那么深度不为0，因此temp从1开始

for(j = P.nodes[i].parent, temp = 1; -1 != j; j = P.nodes[j].parent){

temp++;

}

if(max < temp)

max = temp;

}

return max;

}

int CountLeaf(PTree P) //defnett

{

int i, k, length = 0;

int\* temp = (int\*)malloc(P.n \* sizeof(int));

temp[length++] = P.nodes[0].parent;

for(i = 0; i < P.n; ++i) {

// 在temp[]中找到与parent相等的break

for(k = 0;k < length;k++)

if(P.nodes[i].parent == temp[k])

break;

// 若没有相等的

if(k >= length)

temp[length++] = P.nodes[i].parent;

}

free(temp);

// 因为根结点中双亲是-1，不是树的结点，因此length再减1

length -= 1;

return P.n - length;

}

Status Visit(TElemType e)

{

if(!e)

return ERROR;

printf("%c ",e);

return OK;

}

// 若 cur\_e 是 T 的非根结点，则返回它的双

// 亲所在数组中的下标，否则函数值为“空”。

int Parent(PTree P,TElemType cur\_e) {

int i;

for(i = 0;i < P.n && P.nodes[i].data != cur\_e; ++i);

// 若是没有cur\_e或者cur\_e所在结点为根结点返回 ERROR

if(i >= P.n || P.nodes[i].parent == -1)

return ERROR;

return P.nodes[i].parent;

}

Status Assign(PTree &P,TElemType cur\_e,TElemType value)

{ /\* 初始条件: 树T存在,cur\_e是树T中结点的值。操作结果: 改cur\_e为value \*/

int i;

for(i = 0;i < P.n;i++)

{

if(P.nodes[i].data == cur\_e)

{

P.nodes[i].data = value;

return OK;

}

}

return ERROR;

}

int CountDegree(PTree P) //defnett

{

int i, j, max = -1;

int temp[20] = { 0 };

for(i = 0; i < P.n; ++i) {

for(j = 0; j < P.n; ++j){

if(i == P.nodes[j].parent)

temp[i]++;

}

}

for(i = 0; i < P.n; ++i)

if(temp[i] > max)

max = temp[i];

// 因为根结点中双亲是-1，不是树的结点，因此length再减1

return max;

}

// 树的层次遍历

Status LevelOrderTraverse(PTree P,Status (\*pvisit)(TElemType e)){

int i, j, rear = 0, front = 0;

//类似于一个队列，用来存储每层的结点，由于队列为P.n足够长，不可能放满

int\* temp = (int\*)malloc(P.n \* sizeof(int));

temp[rear] = P.r;

rear = (rear + 1) % P.n;

while(rear != front){

i = temp[front];

front = (front + 1) % P.n;

if(!pvisit(P.nodes[i].data))

return ERROR;

// 一般情况下，双亲肯定在孩子的前面，因为若

// 双亲在后面的话，孩子结点的parent域没法写

// j 可以从i开始

for(j = i; j < P.n; ++j)

{

if(P.nodes[j].parent == i){

temp[rear] = j;

rear = (rear + 1) % P.n;

}

}

}

free(temp);

printf("\n");

return OK;

}

int main(void)

{

PTree p;

CreateTree(p);

LevelOrderTraverse(p ,Visit);

Assign(p ,'F','X');

LevelOrderTraverse(p ,Visit);

printf("树的深度：%d \n" ,DepthTree(p));

printf("树的叶子：%d \n" ,CountLeaf(p));

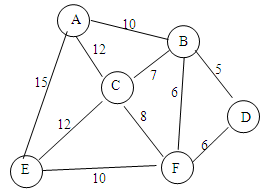
printf("树的 度：%d \n" ,CountDegree(p));

return 0;

}

### 图的存储及图的遍历

1. 有一个带权无向图G，如下所示：



1. 画出该图的邻接矩阵**和**邻接链表存储结构。（注意权值如何添加）
2. 并从顶点A开始进行深度优先**和**广度优先遍历。
3. 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和广度优先遍历。

