考试科目名称　 　**数据结构**　　　（A1卷）

考试方式：闭卷　　考试日期 　年　月　　日　教师

系（专业）　　　计算机科学与技术系　　　　年级　二　　 　班级

学号　　　　　　　　　　　　　姓名　　　　　　　　　　　成绩

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 题号 | 一 | 二 | 三 | 四 | 五 | 六 | 七 | 八 | 九 | 十 |
| 分数 |  |  |  |  |  |  |  |  |  |  |

|  |  |
| --- | --- |
| 得分 |  |

**1、填空题。**（每小题**2**分，本题满分2**0**分）

(1) C++语言中，数组是按行优先顺序存储的，假设定义了一个二维数组A[20][30]，每个元素占两个字节，其起始地址为2140，则二维数组A的最后一个数据元素的地址为

2140+2\*(30\*20-1) = 3338(3338,3339) 。

(2) 若A，B是两个单链表，链表长度分别为n和m，其元素值递增有序，将A和B归并成一个按元素值递增有序的单链表，并要求辅助空间为O(1)，则实现该功能的算法的时间复杂度为 O(m+n) 。

(3) 快速排序的平均时间复杂度是\_\_\_\_n\*lg n\_\_\_\_\_\_\_\_\_\_\_。

(4) 假设有一个包含9个元素的最小堆，存放在数组A中，则一定比A[3]大的元素有\_ \_2 (A[7],A[8]) \_\_\_\_个；一定比A[3]小的元素有\_\_2 (A[0],A[1])\_\_\_\_个。（元素从第0个位置开始存放）

(5) 广义表(((A)),(B,C), D, ((A), ((E,F)))) 的长度是 4 ，深度是 4 。

(6) 有10个元素的有序表，采用折半查找，需要比较4次才可找到的元素个数为\_\_\_\_3\_\_\_\_\_。

(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 \_\_\_。

(8) 假设计算斐波那契数的函数Fib(long n)定义如下：

long Fib(long n){

if(n<=1) return n;

else return Fib(n-1)+Fib(n-2)

}

计算Fib(5)时的递归调用树（即指明函数调用关系的树）的高度是\_\_\_4 \_\_\_\_\_。假设叶子结点所在的高度为0。

(9) 完全二叉树按照层次次序，自顶向下，同层从左到右顺序从0开始编号时，编号为i的结点的左子结点的编号为\_\_\_2\*i+1\_\_\_\_\_\_。

(10) 假设用子女—兄弟链表方式表示森林，对应的二叉树的根结点是p，那么森林的第三棵树的根结点在二叉树中对应的结点是： \_\_\_p->rightchild->rightchild\_\_\_\_\_\_\_\_\_\_\_\_。假设二叉树的结点结构为：

leftchild data rightchild

|  |  |
| --- | --- |
| 得分 |  |

**2、选择题。**（每小题**2**分，本题满分**20**分）

(1) 如果能够在只知道指针p指向链表中任一结点，不知道头指针的情况下，将结点\*p从链表中删除，则这个链表结构应该是： ( B,C )（多选题）

A. 单链表 B. 循环链表 C. 双向链表 D. 带头结点的单链表

(2) 以下哪种矩阵压缩存储后会失去随机存取的功能?( A )

A. 稀疏矩阵 B. 对称矩阵 C. 对角矩阵 D. 上三角矩阵

(3) 下面哪一方法可以判断出一个有向图是否有环（回路）：( B ) (选A，B也对)

A. 广度优先遍历 B. 拓扑排序 C. 求最短路径 D.求关键路径

(4) n个结点的线索二叉树(没有头结点)上含有的线索数为（ B ）

A. 2n B. n－l C. n＋l D. n

(5) 循环队列存储在数组A[0..m]中，则入队时队尾指针rear的操作为（ D ）

A. rear=rear+1 B. rear=(rear+1) mod (m-1)

C. rear=(rear+1) mod m D. rear=(rear+1)mod(m+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

(8) 已知一算术表达式的后缀形式为ABC\*+DE/-，其前缀形式为：( D )

A. -A+B\*C/DE B. -A+B\*CD/E C. -+\*ABC/DE D. -+A\*BC/DE

(9) n个关键码排序，如果选用直接插入排序方法，则元素的移动次数在最坏情况下可以达到( B )。  
A. n\*n/2 B. n\*(n-1)/2 C. n/2 D. (n-1)/2

(10) 关键路径是AOE网络中　　A　C　。（多选）

A. 从源点到汇点的最长路径 B. 从源点到汇点的最短路径

C. 所有活动都是关键活动的路径 D. 最短回路

|  |  |
| --- | --- |
| 得分 |  |

**３、简答题。**（每小题**5**分，本题满分**20**分）

(1) 对如下无向图，按照Dijkstra算法，写出从顶点 1 到其它各个顶点的最短路径和最短路径长度。

1 3 5

10

5

8

6

9

11

7

7

2 4 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 结点编号 | 1 | 2 | 3 | 4 | 5 | 6 |
| 路径长度 | 0 | 11 | 7 | 12 | 14 | 15 |

最短路径 1 1-2 1-3 1-3-4 1-3-5 1-3-6

(2）请画出在如下图所示的5阶B树中插入一个关键码360后得到的B树。

100 200 300 400

20 40 150 180 240 260 310 320 350 370 420 430

300

100 200 350 400

20 40 150 180 240 260 310 320 360 370 420 430

(3) 假设有权值集合{16,40,15,4,25}，给出相应的huffman树。假设某类信息由符号a,b,c,d,e,组成，而上面的权值分别是符号a,b,c,d,e的出现频率。请给出各个符号的Huffman编码。

4

15

19

16

35

25

60

40

100

Huffman编码：

a: 001

b: 1

c: 0001

d: 0000

e: 01

注意：因为左右子树不同，

所以编码可以有多种，但是只要

其长度分别是3，1，4，4，2；且

相互之间不形成前缀关系就

是正确的。

(4)在AVL树的插入操作中，假设插入一个结点后，当前节点p的平衡因子是-2，其左子结点的平衡因子是+1，左子结点的右子结点的平衡因子是-1。如图所示，请给出旋转调整之后的结构。

p

0

C

-1

+1

A

B

t2

t1

t4

t3

A

B

C

t1

t2

t3

t4

-2

+1

-1

p

|  |  |
| --- | --- |
| 得分 |  |

**4、**已知输入关键码序列为（10,90,20,60,78,35,42,31,15），地址区间为0～11。

(1) 请设计一个散列函数，把上述关键码散到0～11中。画出散列表，冲突用线性探测法解决。（5分）

散列函数为：

f(x) = x % 12 允许其它的散列函数

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 60/1 | 31/7 |  | 15/1 |  |  | 90/1 | 78/2 | 20/1 | 42/4 | 10/1 | 35/1 |

(2) 搜索元素31需要比较的次数是多少？（2分）

7 (7-->8->9->10->11->0->1 (成功））

(3) 计算在等概率情况下查找成功的平均查找长度ASLsucc。（3分）

(1+7+1+1+2+1+4+1+1) / 9 = 19/9

|  |  |
| --- | --- |
| 得分 |  |

**5、 程序设计题。**（每小题**15**分，本题满分**30**分）

1. 设计一个算法，根据一棵二叉树的前序序列和中序序列，构造出这棵二叉树。 二叉树的结点都用字符表示。前序序列和中序序列都是字符串。二叉树的结点定义如下：

struct binTreeNode

{char data;

binTreeNode \*leftChild, \*rightChild;

}

**解：**

TreeNode \* tree(char \*preorder, \*midorder)

{

return

TreeRecursive(preorder, 0, strlen(preorder)-1, midorder, 0, strlen(midorder)-1);

}

TreeNode \* TreeREcursive(char \*pre, int preSt, int preEnd,

char\* mid, int midSt, int midEnd)

{

if (preEnd < preSt)

return NULL;

char rt = pre[preSt];

for(int j = midSt; j<=midEnd; j++)

if(mid[j] == rt) break;

if(j>midEnd){cout<<"Wrong input"; return NULL;}

TreeNode root = new binTreeNode( );

root->data = rt;

int lLen = j - midSt;

root->leftChild = treeRecursive(pre, preSt+1, preSt+lLen - 1,

mid, midSt,midSt+lLen-1);

root->rightChild=treeRecursive(pre, preSt+lLen+1, preEnd,

mid, midSt+lLen+1, midEnd);

return root;

}

要点在于根据preOrder的第一个字符，在midOrder中找到左右子树的分界；然后递归调用生成左右子树；做到这一点，就可以给出大部分分数。

可能有很多细节上不一样的地方，比如这里的preEnd指向的字符在相应的树中；但是有些人可能是preEnd指向下一个位置。或者传递参数时直接使用字符串传递。都是可以的。

*如果使用别的办法，参考其效率，酌情给分。只要效率过得去，也可以得满分。但是纯粹枚举则得分不高。*

2. 设计非递归算法实现图的深度优先遍历。（图用邻接表表示，已经定义了一个顺序栈stack[top]，top为栈顶指针，使用visit(node)来表示对顶点node的访问。）

图的邻接表结构定义如下：

struct Edge {

int dest;

Edge \*link; //下一条边链指针

}

struct Vertex {

int data;

Edge \*adj; //边链表的头指针

}

class Graph {

private:

Vertex \*Nodetable; //顶点表

int cnt

｝

**解：**

Graph : : DFS(int v)//从v开始搜索；

{

bool visited[MAXVert];

Edge nextEdge[MAXVert];

stack[0] = v;

nextEdge[0] = graph.Nodetable[v].adj;

top = 0;

visited(stack[top]);

visited[stack[top]] = true;

while(top >= 0)

{

while(nextEdge[top] && visited[nextEdge[top]->dest])

//寻找下一个尚未访问的邻接节点

nextEdge[top] = nextEdge[top]->link;

if(nextEdge[top] != NULL)

{

int nextVert = nextEdge[top]->dest;

visite(nextVert); //访问下一个邻接结点；保证了被压入栈中的顶点都被访问；

visited[nextVert] = true;

stack[top+1] = nextVert; //压栈，进入下一个结点；

nextEdge[top+1] = Nodetable[nextVert].adj;

nextEdge[top] = nextEdge[top]->link;

top ++;

}

else top --;

}

}