数据结构与算法 2016 期末 王洪庆 中间有一些题漏掉了

一．是非判断题（正确:√，错误: X; 每题 2 分，共 20 分）

1. 因为顺序表是随机存储结构，所以其数据元素的检索效率为 O(1)。 (X)
2. 逻辑上考虑，静态链表属于链接存储结构，因此其结点需要动态分配。 (X)
3. 栈和队列都是操作受限的线性表，因此其插入和删除操作的效率低。 (X)
4. 农夫过河问题的求解，既可以使用广度优先搜索，也可以使用深度优先搜索。深度优先搜索需要栈控制，而广度优先搜索需要队列控制。 (√)
5. 已知二叉树的先根和中根遍历结果，就可唯一地确定其后根遍历结果。 (√)
6. n 个结点的二叉树中，度为 2 的结点数为 m，则度为 1 的结点数是 n-2m-1。 (√)
7. 描述折半检索的二叉判定树一定是一颗 AVL 树。 (√)
8. 在散列表中，采用线性探测的空地址法处理碰撞时容易产生二次聚集问题，因此应该避免使用该方法。 (X)
9. AOE 网中，增加非关键活动的持续时间，不会影响整个工程的时间。缩短关键活动的持续时间，也不一定能减少整个工程的时间。 (X)
10. 筛选法实现堆排序时，数据元素的存储结构是顺序表。基数排序中，数据元素的存储结构使用了单链表实现。 (√)

二．选择题（单选，每题 2 分，共 20 分）

1. 程序段{ i=0， s=0； while (s<N) {s=s+i; i++; } } 的时间复杂度为： (A)

   (A) $O(N^{1/2})$ (B) $O(\log_2 N)$ (C) O(N) (D) $O(N^2)$

2. KMP 模式匹配中，模式串“abcabbc”的 NEXT 数组值为： (B)

   (A) {-1, 0, 0, 0, -1, 1, 0 } (B) {-1, 0, 0, -1, 0, 2, 0 }

   (C) {-1, 0, 0, -1, -1, 2, 0} (D) {-1, 0, -1, 0, 0, 1, -1}

3. 循环队列 A[0..m]的头尾指针分别是 F、R，牺牲一个空间区分空满时，判满条件是：(C)

   (A) R+1 == F (B) F+1 == R (C) (R+1)%(m+1) == F (D) (F+1)%(m+1) == R

4. 设某 Haffman 树上共有 253 个结点，则其外部结点数目为： (C)

   (A) 125 (B) 126 (C) 127 (D) 128

5. 在一棵含有 n 个结点的树中，只有度为 k 的分支结点和度为 0 的终端结点，则该树中含有的终端结点的数目为： (B)

   (A) n-n/k (B) n-(n-1)/k (C) n-(n+1)/k (D) n-n/(k+1)

8. 十万个数据元素进行排序，在需求“稳定”的前提下，下列排序方法应该选择：(C)

(A) 起泡排序　(B) 快速排序　(C) 二路归并排序　(D) 堆排序

9. 下列图的运算，哪种不能用于检测图的连通性问题：(D)

(A) 图的遍历　(B) 最短路径　(C) 最小生成树　(D) 拓扑排序

10. n 个顶点的无向图邻接表存储结构中，所有顶点的边表结点总个数最多为：(C)

(A) n^2　(B) n(n+1)/2　(C) n(n-1)　(D) n(n-1)/2

三．填空题（每空 2 分，共 40 分）

1. 散列表的检索效率依赖于下面三个因素：散列函数，碰撞处理方法和装填因子。

设有关键字序列：32、13、18、22、38、17、71、99、42，使用散列函数 H(k)=k% 11 和线性探测法解决冲突，按上述序列的次序依次散列到初始为空的散列表 H[0..10]中。构造该散列表：

0	1	2	3	4	5	6	7	8	9	10
22	99	13			38	17	18	71	42	32

2. 给出一组关键字 T=（12，2，16，30，8，28，4，10，20，6，18）。写出用下列算法从小到大排序时第一趟结束时的序列：

希尔排序(起始增量为 5)　12，2，10，20，6，18，4，16，30，8，28

起泡排序　2，12，16，8，28，4，10，20，6，18，30

快速排序(选第 1 个记录为基准)　6，2，10，4，8，12，28，30，20，16，18

3. 阅读下列算法，回答后面的问题。

```
LinkList function1(LinkList L)        //L 是不带头结点的单链表的头指针
{   LinkList p, q;
    if (L && L->next)
    {   q=L; L=L->next; p=L;
        while(p->next) p=p->next;
        p->next=q; q->next=NULL;

    }
    return L;
}
```

该算法的功能为：原第一个结点成为最后一个结点，原第二个结点成为第一个。

4. 二叉树中的结点类型 BTNode 定义为：{ EType data; BTNode *left, *right; }。

```
int function1(BTNode * BT)
{   if (BT==NULL)  return 0;
    else if ( BT->left==NULL && BT->right == NULL )  return 1;
    else return (function1 (BT->left) + function1 (BT->right) );
}
```

该算法的功能为：统计 BT 为根的二叉树中终端结点个数

7. 完成下列包含 n 个记录的待排序序列 pvector 的堆排序算法。记录序列采用顺序存储，其定义如下：

```
typedef tagSortObject
{    int record[MAX_NUM], n;   //record 为记录数组，n 为记录个数(n<MAX_NUM)
} SortObject;
```

假定调整以 K_i 为根的筛选算法已经实现，其定义如下：

```
void SIFT(SortObject *pvector, int i, int n)。
```

另外定义 r1 和 r2 记录交换算法为 void SWAP(RecordNode *r1, RecordNode *r2)。

完成下列堆排序算法：

```
void headSort(SortObject *pvector)
{    int i, n = pvector->n;
     for ( i= n/2-1 ; i >= 0; i--)                    //建立初始堆
     {    SIFT(pvector, ____i, n____);    }
     for (i = n-1 ; i>0; i--)                         //各趟堆排序
     {    SWAP(&pvector->record[0],   &pvector->record[i]);
          SIFT(pvector, ____0, i____);
     }
}
```

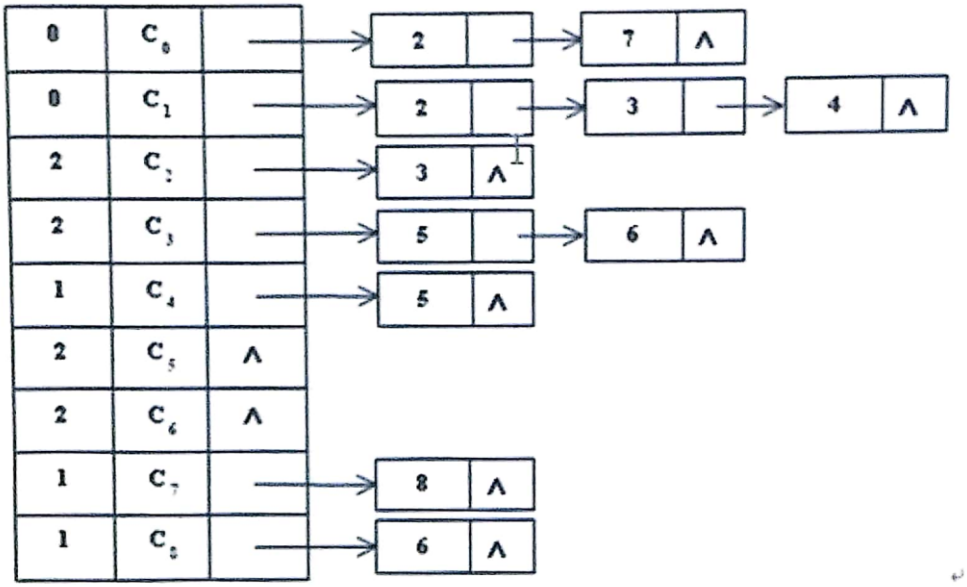
n 个记录，且有 $h=\log_2(n+1)$。则，初始堆构建过程中的比较次数 $C_1(n)$的计算公式为：

$$C_1(n)=\sum_{m=0}^{h-1} \quad \underline{\quad (h-m)*2^{m+1} \quad}$$

重新建堆总共的比较次数 $C_2(n)$的计算公式为：

$$C_2(n)<\sum_{j=1}^{n-1} \quad \underline{\quad 2\lfloor \log_2(n-j+1) \rfloor \quad}$$

8. AOV 网的邻接表（出边表）存储结构如下：



1） 按照深度优先遍历算法，写出从 C_0 开始的深度优先遍历结果：

C0 C2 C3 C5 C6 C7 C8 C1 C4

2） 按照广度优先遍历算法，写出从 C_1 开始的广度优先遍历结果：

C1 C2 C3 C4 C5 C6 C0 C7 C8

3） 按照拓扑排序算法，写出 AOV 网的拓扑序列：

C1 C4 C0 C7 C8 C2 C3 C6 C5

四．问答题（每题 10 分，共 20 分）

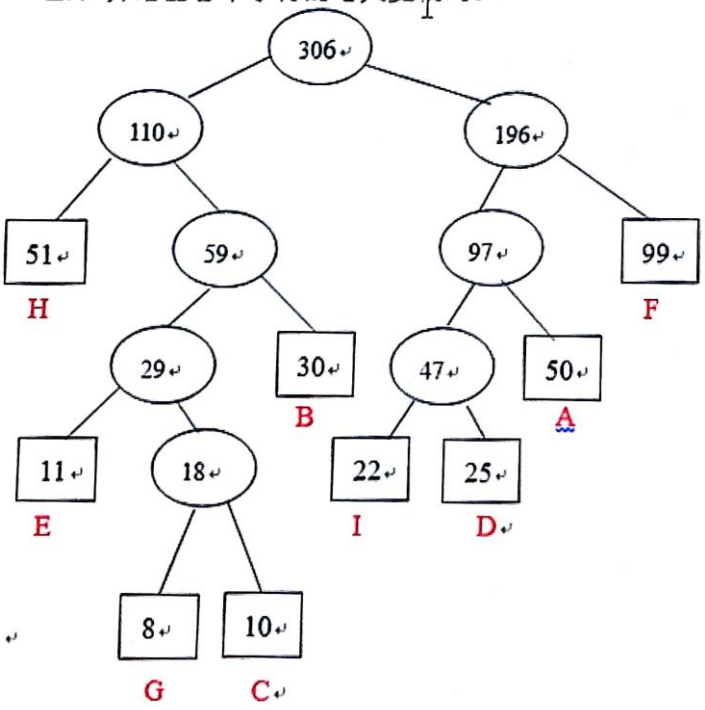
1. 回答下列哈夫曼树以及哈夫曼编码的相关问题（每小题 5 分）：

   1） 假定字符个数为 n，试简单描述哈夫曼编码算法实现时应该选择的合适存储结构。

三叉静态链表，m 个外部结点，必有 m-1 个内部结点。

2m-1 个元素，前 m 个外部结点，后 m-1 个内部构造结点，最后 1 个是根结点。

2）假定报文中各字符及其出现频率如下：A（50），B（30），C（10），D（25），E（11），F（99），G（8），H（51），I（22）。试画出对应的哈夫曼树（严格按照左小右大构造），并给出各个字符的哈夫曼编码。



A: 101
B: 011
C: 01011
D: 1001
E: 0100
F: 11
G: 01010
H: 00
I: 1000

2. 回答下列动态字典的相关问题（每小题 5 分）

1）为什么二叉排序树是动态字典的合适表达（从插入、删除和检索三个方面回答）？

假定 N 个记录，插入、删除：无数据元素的移动，$\log_2 N$

检索：类似折半检索（缩小区间的检索），效率高，$\log_2 N$

中序遍历：有序

2）等概率检索情况下，插入、删除记录后如何保持二叉排序树的高检索效率?

保持二叉排序树是 AVL 树（平衡二叉树），AVL 树的检索效率高。

每个结点增加平衡因子，判断最小不平衡子树，根据 4 中情况（LL， RR， LR， RL）之一进行调整。