《数据结构》考试题（闭卷）　　A 卷

（电信系本科 2016 级　 2017 年 5 月 16 日）

姓名 _____　　班级 _____ 学号 _____

题　号	一	二	三	总分
题　分	40	32	40	112
得　分				

注：总分 112 分，不折算，超过 100 分按 100 分计

得　分

一、回答下列问题　(每题 5 分，共 40 分)

1．以下关于静态链表，说法错误的是（　　）

(1) 静态链表既有顺序存储的优点，又有动态链表的优点。所以，它存取表中第 *i* 个元素的时间与 *i* 无关。

(2) 静态链表中能容纳的元素个数的最大数在表定义时就确定了，以后不能增加。

(3) 静态链表与动态链表在元素的插入、删除上类似，不需做元素的移动。

A．(1)，(2)　　B．(1)　　C．(1)，(2)，(3)　　D．(2)

解答：B

2．(*Stanford University*) 阅读下列程序代码，请从 A-F 中选择对应的复杂度填写在程序左侧的横线上。

```
___B___     int f1(int N){
                int x = 0;
                for(int i = 0; i < N; i++)
                    x++;
                return x;
            }
______      int f2(int N){
                int x = 0;
                for(int i = 0; i < N; i++)
                    for(int j = 0; j < i; j++)
                        x++;
                return x;
            }
______      int f3(int N){
                if(N == 0) return 1;
                int x = 0;
                for(int i = 0; i < N; i++)
```

```
                x += f3(N-1);
        return x;
}
_____   int f4(int N){
        if(N == 0) return 0;
        return f4(N/2)+f1(N)+f4(N/2);
}
_____   int f5(int N){
        int x = 0;
        for(int i = N; i > 0; i = i/2)
                x += f1(i);
        return x;
}
_____   int f6(int N){
        if (N == 0) return 1;
        return f6(N-1)+f6(N-1);
}
_____   int f7(int N){
        if(N == 1) return 0;
        return 1+f7(N/2);
}
```

A. Log_2N　　B. N　　C. $Nlog_2N$　　D. N^2　　E. 2^N　　F. N!

解答：　B, D F C B E A

i. Single loop.

ii. Similar to insertion sort.

iii. Similar to enumerating all permutations.

iv. Similar to mergesort since f1(N) takes linear time.

v. N + N/2 + N/4 + . . .

vi. Similar to enumerating all subsets.

vii. Similar to binary search.

3. 已知待搜索的主字符串为：asjfsglklkabbcdesdfks， 寻找其子符串为：abbcde，求用 KMP 匹配算法的 NEXT 数组的值

解答：主要考察学生是否知道next数组和待搜索的字符串无关的结论，以及next数组的计算。

Next只和abbcde有关，值为012111

4．对下列关键字序列用快速排序法（第一个作为基准）进行从低到高排序时，速度最快的

情形是(　　)

A.（21, 25, 5, 17, 9, 23, 30）

B.（25, 23, 30, 17, 21, 5, 9）

C.（21, 9, 17, 30, 25, 23, 5）

D.（5, 9, 17, 21, 23, 25, 30）

解析：A

最后排序结果是：（5, 9, 17, 21, 23, 25, 30）

一趟快速排序越等分越好：

A 一趟后是：（9, 5, 7, 21, 25, 23, 30），左右都只需再需一趟即可

C 一趟后是：（5, 9, 7, 21, 25, 23, 30），左边还需两趟，右边还需一趟。

5. 假定有 *k* 个关键字的 Hash 函数值相同，处理冲突用线性探测法把这 *k* 个关键字存入 Hash 表中，至少要进行多少次比较？

解答：

1+2+3+…+k-1=k*(k-1)/2

6. 定义函数 f 如下：

```
int     f(int x)
{
      if (x > 0)
            return x * f(x-1);
      else
            return 2;
}
```

执行完　i　= f(f(1))；语句后，变量 i 值为（　　）

A. 2　　**B.** 4　　**C.** 8　　**D.** 无限递归

解答：　B

7.一颗高度为 h 的二叉树，若只有度为 0 和 2 的节点，则该树包含的节点数最多是多少？最少是多少？为什么？

8. 为 *n* 个关键字建初始堆，什么情况下比较次数最少？对应的比较次数和移动次数分别为多少？

解：当已是一个堆，比较次数最少，比较次数是 2*n/2，移动次数是 0.

得　分

二、综合题（每题 8 分，共 32 分）

1. 用递归算法实现从尾到头输出单链表的结点。

解答：

思路：每访问到一个结点的时候，先递归输出它后面的结点，再输出该结点自身，这样链表的输出结果就反过来了

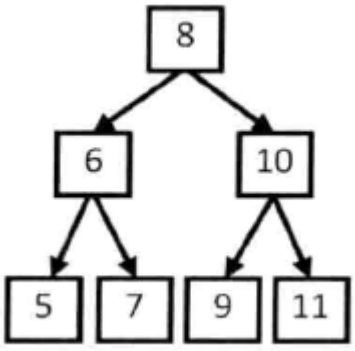
```
void PrintListReversely(ListNode* pListHead)
{
    if(pListHead != NULL)
    {
        // Print the next node first
        if (pListHead->m_pNext != NULL)
        {
            PrintListReversely(pListHead->m_pNext);
        }

        // Print this node
        printf("%d", pListHead->m_nKey);
    }
}
```

2.现有数组{ 5,7,6,9,11,10,8 }，判断该数组是不是某二叉排序树的后序遍历的结果？如果是，则画出该二叉搜索树，并给出该二叉排序树的先序遍历和中序遍历结果。

解析：

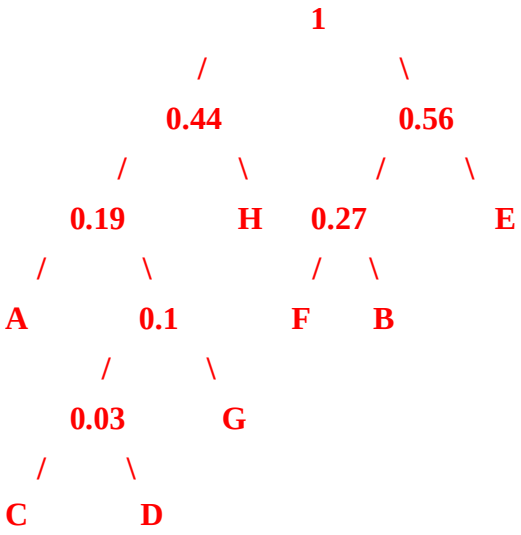
是二叉搜索树的后序遍历结果，该二叉搜索树如下：



先序遍历结果为：{8，6，5，7，10，9，11}
中序遍历结果为：{ 5, 7 , 6，9，11, 10, 8 }

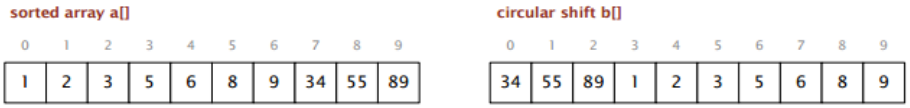
3. 设 A～H 8 个字符出现的概率为: w={0.09， 0.16， 0.01, 0.02， 0.29， 0.11， 0.07， 0.25}，设计最优二进制码并计算平均码长。如果收到的序列为：11100011110011011011，则对应的报文是什么？（假设霍夫曼树编码的原则是左 0 右 1）

答：哈夫曼树为



因此：A的编码：000
B的编码：101
C的编码：00100
D的编码：00101
E的编码：11
F的编码：100
G的编码：0011
H的编码：01
平均码长：3*0.09+3+*0.16+5*0.01+5*0.02+2*0.29+3*0.11+4*0.07+2*0.25=2.59
译码：EFHEFEHB

4. （***Stanford University***）给定由小到大排列的有序数组 ***a***，其中 ***a*** 中元素均不相等。数组 ***b*** 是 ***a*** 的一个循环移动结果，移动了多少位未知。如下图所示



请设计一个快速算法确定 ***x*** 是否存在于数组 ***b*** 中（不可以使用数组 ***a***）。要求算法复杂度不高于 $\log_2 N$。请先写出算法的设计思路，再用伪 C 代码进行描述。

答：

方案一：采用类似二分查找的形式，找到 b 中最小的数字。然后 b 变成两个有序数组，在这两个有序数组中的一个用二分查找确定 x 是否存在。找最小元素位置 r 的算法如下：

If either $N \le 1$ or $b[0] < b[N-1]$, then the array is sorted and $r = 0$. Otherwise, finding the index r is similar to problem 6 on the Fall 2014 midterm because r is the unique index for which $b[r-1] > b[r]$. We maintain the invariant that $b[lo] > b[hi]$ for two indices $lo < hi$. Initially, we set $lo = 0$ and $hi = N-1$. Now, pick $mid = (lo + hi)/2$. There are three cases:

- if $hi = lo + 1$, return hi
- else if $b[mid] < b[hi]$, then set $hi = mid$
- else if $b[mid] > b[hi]$, then set $lo = mid$

方案二：分六种情况讨论

Solution 2. This solutions searches for the search key x without necessarily finding the crossover index r. We maintain the invariant that if x is in the array b, then it is in $b[lo..hi]$. Initially, we set $lo = 0$ and $hi = N-1$. Now, pick $mid = (lo + hi)/2$. There are six cases:

- if $hi < lo$, return false
- else if $b[mid] = x$, return true
- else if $b[lo] \le x < b[mid]$, do a regular binary search for x in $b[lo..mid-1]$.
- else if $b[mid] < x \le b[hi]$, do a regular binary search for x in $b[mid+1..hi]$
- else if $b[mid] < b[hi]$, then set $hi = mid - 1$
- else, set $lo = mid + 1$

The number of 3-way compares is $\sim 2 \lg N$ in the worst case.

We also remark that there is no way to solve the problem in logarithmic time if duplicate keys are permitted. To see why, consider an array b containing all 0s, but with a single 1 somewhere in the interior. Note that b is a circular shift of a sorted array. Now, there is no efficient way to search for the key 1.

得　分

三、　算法设计题（每题 8 分，共 40 分）

1. 一个有 *n* 个结点的双向循环链表 *H*，随机存放奇数和偶数，试设计一高效算法实现链表结点中的奇偶数相邻存放（不要求有序），多余的奇数和偶数放在最后。
   a. 给出算法思想
   b. 编写伪 C 语言算法
   c. 分析该算法的时间和空间复杂度。

解答：

可设两指针 P1=H 和 P2= H→next 指向结点分别存放奇数和偶数（或偶数和奇数）两指针分别以步长 2 向后滑动，当两指针指向的结点的奇偶内容相反时进行交换，结束条件为 P1 或 P2 任何一个等于 H，时间复杂度 O（n），空间复杂度 O（1）,该算法思路不唯一。

2. 给定函数 *d*(*n*)为数 *n* 及 *n* 的各位之和，且 *n* 为正整数，如 *d*(78)=78+7+8=93。这样这个函数可以看成一个生成器，如 93 可以看成由 78 生成。
   定义数 *A*：数 *A* 找不到一个数 *B* 可以由 *d*(*B*)=*A*，即 *A* 不能由其他数生成。现在要写程序，找出 1 至 10000 里的所有符合数 *A* 定义的数。

解答：

算法思路：

申请一个长度为 10000 的 bool 数组，每个元素代表对应的值是否可以有其它数生成。开始时将数组中的值都初始化为 false。

由于大于 10000 的数的生成数必定大于 10000，所以我们只需遍历 1 到 10000 中的数，计算生成数，并将 bool 数组中对应的值设置为 true，表示这个数可以有其它数生成。

最后 bool 数组中值为 false 的位置对应的整数就是不能由其它数生成的

```
Main()
{
    int bool[1000];
    int n,m;
    for (n=1; n<=1000; n++) bool[n]=false;
   for (n=1; n<=1000; n++){
        m=find(n);
        Bool(m)=ture;
   }
   for (n=1; n<=1000; n++){
        if(bool[n] == 'false') printf('find data:    %d \n ',n);
   }
```

```
}

int find( k )
{
     int m, o;
    m=k;
     o=k;
     for (p=1; p<=3; p++) {
          o=o+m%10;
          m=m/10;
     }
     return(o);
}
```

3. 在一个字符串中找到第一个只出现一次的字符。如输入 **abaccdeff**，则输出 **b**。试用伪 **C** 语言编写实现该功能的算法。

解答：创建一个长度为 256 的数组，每个字母根据其 ASCII 码值作为数组的下标对应数组的对应项，而数组中存储的是每个字符对应的次数。这样我们就创建了一个大小为 256，以字符 ASCII 码为键值的哈希表

第一遍扫描这个数组时，每碰到一个字符，在哈希表中找到对应的项并把出现的次数增加一次。这样在进行第二次扫描时，就能直接从哈希表中得到每个字符出现的次数了

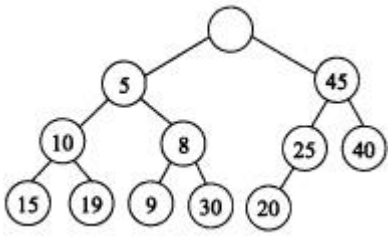
```
#include<stdio.h>
char firstsingle(char *arr)
{
  char asc[255] = {0};
  int i = 0;
  for(i = 0; arr[i] != '\0'; i++)
  {
    asc[arr[i]]++;
  }
  for(i = 0; arr[i] != '\0';i++)
  {
    if(asc[arr[i]] == 1)
    {
       return arr[i];
    }
  }
```

```
  return '\0';
}
int main()
{
  char arr[10];
  char ret;
  scanf("%s",&arr);
  ret = firstsingle(arr);
  printf("%c\n",ret);
  return 0;
}
```

4. 数据结构 DEAP（双堆）的定义如下：DEAP 是一棵完全二叉树，它或者是一棵空树，或者满足下列特性：（1）树根不包含元素．（2）其左子树是一小堆（MINHEAP），其右子树是一大堆（MAXHEAP）．（3）若右子树非空，设 *i* 是左子树的任一结点，*j* 是右子树中与 *i* 相应的结点．若这样的 *j* 结点不存在，则取 *j* 为右子树中与 *i* 的父结点相应的结点；结点 *i* 的关键字总是小于或等于结点 *j* 的关键字值。一个 DEAP 的例子如下图所示，与结点 15 相对应的结点为 20，与结点 19 对应的结点为 25．

   a. 给出在该 DEAP 中插入结点 3 后的结果．

   b. 写出在 DEAP 中插入新结点的算法思路（不要求用伪 C 语言描述）．

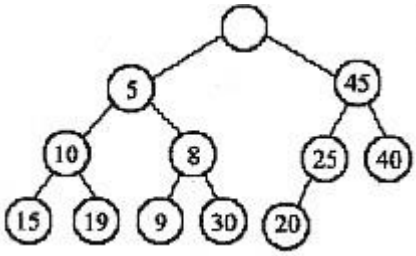


解答：

插入情况：当均为空二叉树或满二叉树（m=2h-1）时，应在小堆插入；小堆满（二叉树）后在大堆插入。即当 m>=n 且 m<>2h-1 且 log2m-log2n<=1 在小堆插入，否则在大堆插入。

最后分析调堆情况：在小堆 m 处插入结点 x 后，若 x 的值不大于大堆的 m/2 结点的值，则在小堆调堆；否则，结点 x 与大堆的 m/2 结点交换，然后进行大堆调堆。在大堆 n 处插入结点 x 后，若 x 不小于小堆的 n 结点，则在大堆调堆；否则，结点 x 与小堆的 n 结点交换，然后进行小堆调堆。

在 DEAP 中插入结点 4 后的结果如图：



4 先插入到大堆，因为 4 小于小堆中对应位置的 19，所以和 19 交换。交换后只需调整小堆，从叶子到根结点。这时，大堆不需调整，因为插入小堆 19 时，要求 19 必须小于对应大堆双亲位置的 25，否则，要进行交换。

5. （*Stanford University*）给定一个 $N \times N$ 实数矩阵，找出在每行都至少出现过一次的最大的数（如果没有则返回 NULL）。比如以下矩阵，找到的数是 5。

9	6	3	8	5
3	5	1	6	8
0	7	5	3	5
3	5	7	8	6
4	3	5	7	9

要求在最坏的情况下，算法的时间复杂度是 O（$N^2 \log N$），空间复杂度是 O(N^2)。

a. 请用首先用中文描述算法思路，然后再用伪 C 代码描述你的算法。

b. 分析你的算法的时间复杂度是多少？

解答：

(a)

1. Sort each row using heapsort.

2. For each number in row 0, from largest to smallest, use binary search to check if it appears in the other N − 1 rows.

3. Return the first number that appears in all N rows. The order of growth of the running time is $N^2 \log N$, with the bottleneck being steps 1 and 2. Correctness follows because the largest common number must appear in row 0. Scanning the numbers in row 0 from largest to smallest ensures that we find the largest common number.

(b) $N^2 \log N$