装订线内

不要答题

北京大学信息科学技术学院考试试卷

考试科目：数据结构与算法 A **姓名：**_____ **学号：**_____

考试时间： 2018 年 1 月 3 日 **任课教师:** _____

题号	一 10	二 15	三 5	四 10	五 30	六 30	总分
分数							
阅卷人							

北京大学考场纪律

1、考生进入考场后，按照监考老师安排隔位就座，将学生证放在桌面上。无学生证者不能参加考试；迟到超过 15 分钟不得入场。在考试开始 30 分钟后方可交卷出场。

2、除必要的文具和主考教师允许的工具书、参考书、计算器以外，其它所有物品（包括空白纸张、手机等）不得带入座位，已经带入考场的必须放在监考人员指定的位置，并关闭手机等一切电子设备。

3、考试使用的试题、答卷、草稿纸由监考人员统一发放，考试结束时收回，一律不准带出考场。若有试题印制问题请向监考教师提出，不得向其他考生询问。提前答完试卷，应举手示意请监考人员收卷后方可离开；交卷后不得在考场内逗留或在附近高声交谈。未交卷擅自离开考场，不得重新进入考场答卷。考试结束监考人员宣布收卷时，考生立即停止答卷，在座位上等待监考人员收卷清点后，方可离场。

4、考生要严格遵守考场规则，在规定时间内独立完成答卷。不准旁窥、交头接耳、打暗号，不准携带与考试内容相关的材料参加考试，不准抄袭或者有意让他人抄袭答题内容，不准接传答案或者试卷等。凡有严重违纪或作弊者，一经发现，当场取消其考试资格，并根据《北京大学本科考试工作与学习纪律管理规定》及其他相关规定严肃处理。

5、考生须确认自己填写的个人信息真实、准确，并承担信息填写错误带来的一切责任与后果。

学校倡议所有考生以北京大学学生的荣誉与诚信答卷，共同维护北京大学的学术声誉。

以下为试题（共 5 单页）和 答题纸（共 4 页）

得分

一、选择填空题（7 小题，每空 1 分，共 10 分）

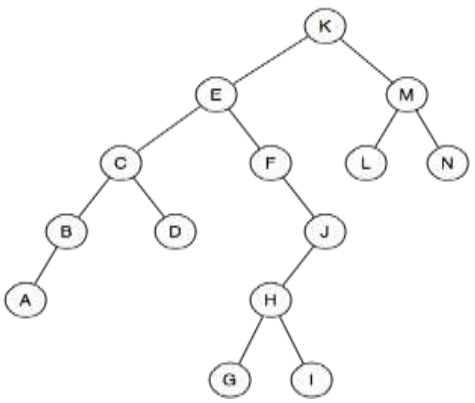
1. 下列说法中正确的是 _____。
   A. 图的广度优先搜索中邻接点的寻找具有“先进后出”的特征。
   B. 连通图的生成树是该图的一个极大连通子图。
   C. 图的广度优先搜索是一个递归过程。
   D. 在非连通图的遍历过程中，每调用一次深度优先搜索算法都得到该图的一个连通分量。

2. 基于比较的内排序算法中，平均时间复杂度为 O(nlogn)的算法 有 _____， 这其中稳定的排序算法 有 _____ 。

3. 在包含 n 个关键码的线性表中从后向前进行顺序检索（0 下标为监视哨）。若第 i 个关键码的检索概率为 p_i ，且满足如下分布 ：
   $p_1 = 1/2,\ p_2 = 1/4,\ \ldots,\ p_n = 1/2^n$,
   则成功检索的平均检索长度为 _____， 失败检索的平均检索长度为 _____ 。

4. 给定一组输入关键码 {15, 1, 3, 14, 11, 2, 12, 10, 4, 17}，采用大小为 3 的最小堆进行置换选择排序的输出为 _____ 。

5. 按照 AVL 定义，下图所示的 BST 中没有达到平衡状态的结点有



_____ 。

6. 假设磁盘块大小是 1024 字节，每个索引项需要 8 个字节。 一个包含 20000 条记录的数据文件，若采用二级索引， 则一级索引文件和二级索引文件共需占用 _____ 磁盘块。

7. 给定两个广义表 S = ((a, (b), c), ((d), e)), T = (f, (g, ((h)), i, j)); 利用广义表的 Head 和 Tail 运算， 则有

   d = _____ ;

   h = _____ 。

得分

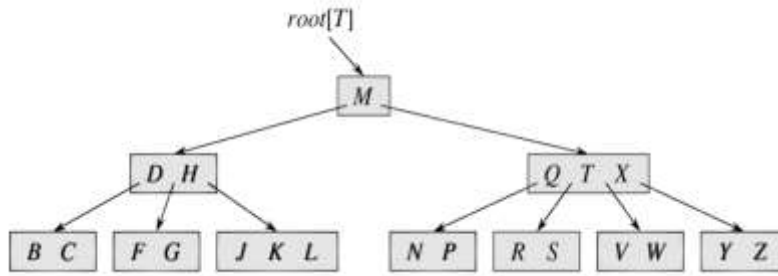
二、简答辨析题（5 小题，共 15 分）

1. （2 分） 将关键码序列（7, 8, 30,11, 18, 9, 14）存储到一个散列表中。散列表是一个下标从 0 开始的一维数组， 若散列函数采用: H(key) = (key * 3) MOD 7， 处理冲突采用线性探测法，装填（负载）因子要求为 0.7。

   (1) . 请画出所构造的散列表;

   (2) . 分别计算等概率情况下查找成功和查找不成功的平均查找长度。

2. （4 分）下图是一棵关键码为英文字符的 m 阶 B 树，依据字典序组织。 请回答下述问题：



   (1) . 这是一棵合法的 B 树，则 m 可以取哪些值，并说明原因;

   (2) . 若 m 取所有可能值中的最小值，则

   (A) 给出查找字符 V 的过程，并说明需要的读盘次数（假设根结点也需读盘操作）;

   (B) 给出插入字符 I（H < I < J）的过程，并画出插入 I 后的 B 树;

   (C) 在原始 B 树（即进行第(B)问的操作之前）的基础上，给出删除 H 的过程，并画出删除 H 后的 B 树。

3. （4 分） 现有 8 个顺串，每个顺串的第一个关键码为 S1[1..8] = {17, 10, 2, 9, 13, 15, 12, 4}，第二个关键码是 S2[1..8] ={23, 34, 38, 25, 27, 40, 36, 39}，进行从小到大的归并。

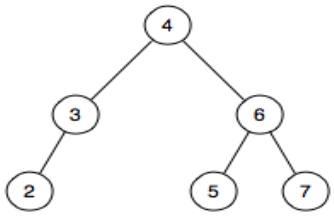
   (1) . 请写出初始构建的赢者树内部结点的数组 A[1..7] 和 败者树内部结点

的数组 B[0..7]（数组 A 和 B 存储的是 1 到 8 的顺串的下标，B[0]表示最终冠军）；

(2)． 在输出第一个全局赢者并重构赢者树或败者树后，写出此时赢者树内部结点的数组 A[1..7]和败者树内部结点的数组 B[0..7]。

4. （3 分）请按照 2, 1, 4, 5, 9 的插入顺序，建立一棵红黑树，并画出每插入一个元素后的红黑树。（用 B 和 R 表示结点的颜色）

5. （2 分）给定下面的一棵 splay 树， 按照结点序列 2, 3, 4, 5, 6, 7 访问所有结点之后，请画出访问后的 splay 树形结构。



得分

三、算法填空题（2 小题，每空 1 分，共 5 分）

1. 下面是考虑墓碑问题的散列表插入算法，算法中不允许有重复关键码插入。

```
template <class Key, class Elem, class KEComp, class EEComp> class
hashdict  {
private:
    Elem* HT;                          // 散列表
    int M;                             // 散列表大小
    int currcnt;                       // 现有元素数目
    Elem EMPTY;                        // 空槽
    int p(Key K, int i)                // 探查函数
    int h(Key K) const ;               // 散列函数
    Key getKey(Elem e);                // 获取元素 e 的关键码值
public:
    hashdict(int sz, Elem e){          // 构造函数，e 用来定义空槽
        M = sz;
        EMPTY = e;
        currcnt = 0;
        HT = new Elem[sz]
        for (int i=0; i<M; i++){
            HT[i] = EMPTY;
        }
    }
    ~hashdict() { delete []HT; }
```

```
    bool hashSearch(const Key&, Elem&) const;
    bool hashInsert(const Elem&);
    Elem hashDelete(const Key& K);
    int size() { return currcnt; }    // 散列表中现有元素数
};

// 墓碑用常量 TOMB 表示，以下为插入函数
template<class Key,class Elem,class KEComp,class EEComp>
bool hashdict<Key,Elem,KEComp,EEComp>::hashInsert(const Elem& e){
    int insplace;
    int i = 0;
    int pos  = home = h(getkey(e));   // 找到新结点基地址
    bool tomb_pos = false;
    while (_____ 填空 1 _________)  {
        if (EEComp::eq(e,HT[pos]))
            return false;
        if (EEComp::eq(TOMB,HT[pos]) && !tomb_pos) {
            _________ 填空 2 ________
            tomb_pos = true;
        }
        i++;
        ________ 填空 3 _________  // 探查
    }
    if (!tomb_pos)
        insplace = pos;           // 若无墓碑，则 insplace 位于空槽位置
    HT[insplace] = e;
    return true;
}
```

2. 设有一个仅由红、白、蓝 3 种颜色的条块组成的序列，各种色块的个数和分布是随机的，但 3 种颜色的总数为 n。下面的代码实现了一种时间复杂度为 O(n)的排序，使得这些条块按照红、白、蓝的顺序排好。请将其补全。

```
const int Red = 1, White = 2, Blue = 3;
void ColorSort(int Array[], int n) {
    int Insert_Red = 0, Insert_White = 0, i;
    for (i=0; i<n; i++) {
        if (Array[i] == White) {
            Array[i] = Array[Insert_White];
            Array[Insert_White] = White;
            Insert_White++;
        }
        if (Array[i] == Red) {
            ________ 填空 4 _________
            ________ 填空 5 _________
            Array[Insert_Red] = Red;
```

```
                Insert_Red++;
                Insert_White++;
            }
        }
    }
```

得分

四、设计分析题（2 小题，共 10 分）

1. （6 分）请设计一个算法求有向无环图（DAG）中每对顶点间的“最长简单路径”（所谓“最长简单路径”，是指该简单路径包含的边数最多）。以一个有向无环图作为输入，对于每对顶点，如果它们之间存在简单路径，则输出其中路径长度最长（边数最多）的简单路径，否则输出空。试分析你所设计算法的时间复杂度。

2. （4 分）利用证明基于比较的排序问题的复杂度下限的方法，试证明在已排序序列上基于比较的检索问题的复杂度下限是 Ω(logn)。

得分

五、期中考试题（共 30 分，成绩由助教登记）

得分

六、上机考试题（共 30 分，成绩由助教登记）