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： 2014 年 1 月 1 日 **任课教师:**_____

题号	一	二	三	四	总分
分数					
阅卷人					

北京大学考场纪律

1、考生进入考场后，按照监考人员安排隔位就座，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、除必要的文具外，其它所有物品（包括空白纸张、手机、或有存储、编程、查询功能的电子用品等）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。

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。考试结束时间到，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。不准交头接耳，不准偷看、夹带、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。凡有违纪作弊者，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术规范条例》及相关规定严肃处理。

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。

以下为试题和答题纸，试题共 5 页。

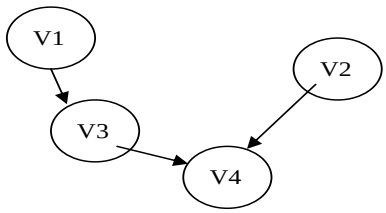
得分

一、填空（27 分）

1. （2 分）当各边上的权值 A 时，BFS 算法可用来解决单源最短路径问题。

   A．均相等　　B．均互不相等　　C．不一定相等

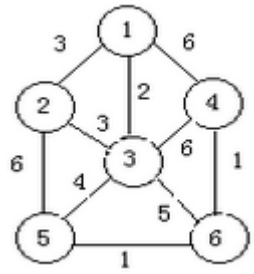
2. （4 分）有向图 G 如下图所示：



   （1）写出所有可能的拓扑序列： 1234, 1324, 2134。

   （2）添加一条弧 v_2v_1，或者 v_3v_2 之后，则仅有惟一的拓扑序列。

3. （2 分）设有如下所示无向图 G，用 Prim 算法构造最小生成树所走过的边的集合 E={(1，3)，(1，2)，(3，5)，(5，6)，(6，4)}（用边（2，3）代替（1，2）也可以）。



4. （2 分）如果只想得到 1000 个元素组成的序列中第 5 个最小元素之前的部分排序的序列，用 C 方法最快。

   A．起泡排序　B．快速排列　C．堆排序　D．简单选择排序

5. （2 分）若要求尽可能快地对序列进行稳定的排序，则应选 B

   A．快速排序　　B．归并排序　　C．冒泡排序　　D.归并排序

6. （2 分)在文件局部有序或文件长度较小的情况下，最佳的排序方法是 A 。

A.直接插入排序　　B. 冒泡排序　　C. 直接选择排序　　D.归并排序

7. （2 分）设输入的关键字满足 $K_1>K_2>\ldots>K_n$ ，缓冲区大小为 m ，用置换选择排序方法可产生 n/m+1 个初始归并段。

8. （2 分）设有 13 个初始归并段，其长度分别为 28，16，37，42，5，9，13，14，20，17，30，12，18。试计算最佳 4 路归并树的带权路径长度 WPL= 480 。

9. （3 分）设有序顺序表中的元素依次为 017, 094, 154, 170, 275, 503, 509, 512, 553, 612, 677, 765, 897, 908，则对其进行折半查找时，等概率下搜索成功的平均查找长度和不成功的平均查找长度分别为 45/14 和 59/15 。

10. （2 分)设散列表长度为 14（地址下标为 0…13)，哈希函数为 H(key)=key%11，表中已有数据的关键字为 15,38,61,84 共四个，现要将关键字为 49 的结点将加入表中，用二次探查法解决冲突，则放入的位置是 D 。

A．8　　B．3　　C．5　　D．9

11. （2 分）在一棵含有 1023 个关键字的 4 阶 B 树中进行查找（不读取数据主文件)，至多读盘 C 次。

A．7　　B. 8　　C. 10　　D. 9

12. （2 分）在一棵阶为 k 的红黑树中，内部结点最少有 2^k-1 个；从根到叶的简单路径长度的范围为 [k,2k]。

得分

二、 辨析与简答（30 分）

1. （8分）将关键字序列（7、8、30、11、18、9、14）散列存储到散列表中。散列表是一个下标从0开始的一维数组，散列函数为：H(key) = (key * 3) MOD 7，处理冲突采用线性探测法，要求装填（载）因子为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。

答案：

(1).根据题意，散列表长度为 L = 7/0.7 = 10；因此此题需要构建的哈希表是下标为 0~9 的一维数组。根据散列函数可以得到如下散列函数值表。

Key	7	8	30	11	18	9	14
H(Key)	0	3	6	5	5	6	0

采用线性探测法处理冲突，所构造的散列表为：

地址	0	1	2	3	4	5	6	7	8	9
关键字	7	14		8		11	30	18	9	

（2）等概率情况下查找成功平均查找长度：

根据（1）的构造过程，查找次数如下表所示：

Key	7	8	30	11	18	9	14
Count	1	1	1	1	3	3	2

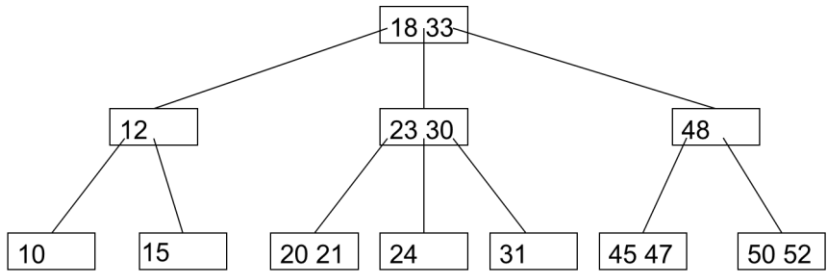
所以 $ASL_{success}$= （1+1+1+1+3+3+2）/ 7 = 12/7。

等概率情况下查找不成功的平均查找长度：见散列表，计算查找不成功的次数就直接找关键字到第一个地址上关键字为空的距离即可， 但根据哈希函数地址为 MOD7，因此初始只可能在 0~6 的位置。等概率情况下，查找 0~6 位置查找不成功的次数表如下表所示

Key	7	8	30	11	18	9	14
Count	3	2	1	2	1	5	4

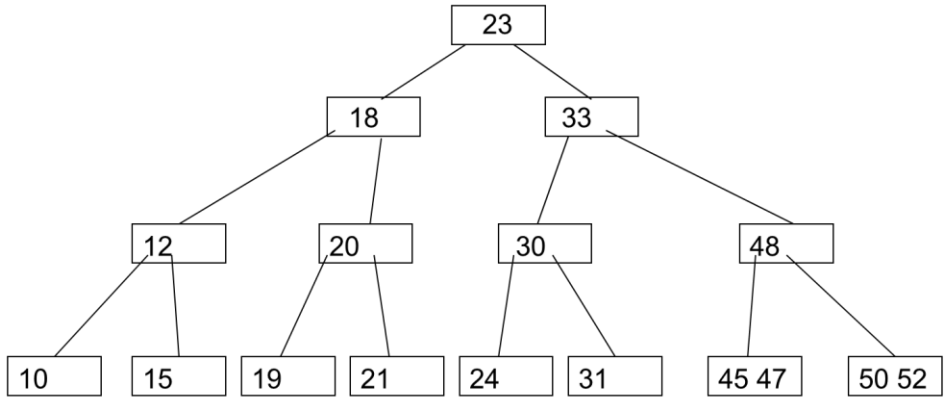
所以 $ASL_{unsuccess}$= （3+2+1+2+1+5+4）/ 7 = 18/7。

2. （6分）设有如下一棵3阶B树，请画出在其中插入关键码19后的B树，并给出插入过程中的访外（读写）次数；在此基础上再删除关键码48，请画出删除后的B树及删除过程中的访外（读写）次数。

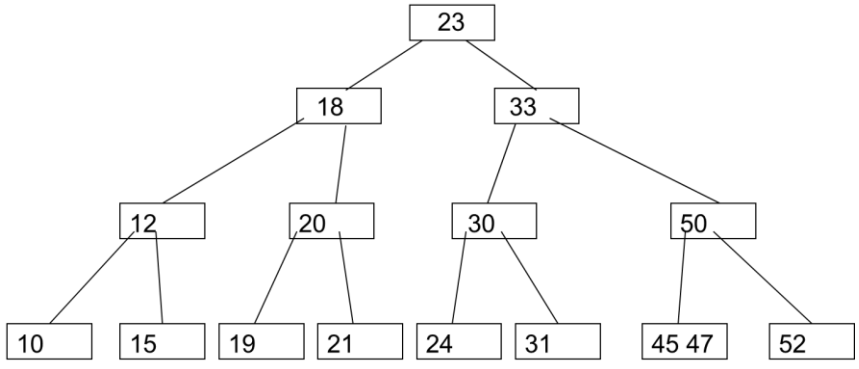


答案：

插入 19 后结点持续分裂到根，整个树高增 1，读盘 3 次，写盘 7 次，共读写 10 次。结果为：



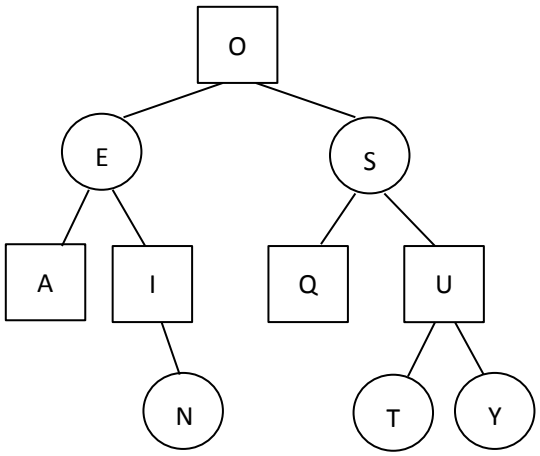
删除 48 时需先与后继 50 交换，后删除，删除后各结点依然满足阶的要求，故不再调整，删除过程中写盘 2 次，若假设之前操作读进的页块不再读的话，读盘 2 次，或者从根开始重新读盘则 4 次，结果如下：



3. （6分）将下列字符序列 E A S Y Q U E S T I O N 依次插入到初始为空的红黑

树（RB-tree）中，请画出最终得到的红黑树。（用圆圈表示红色结点，方框表示黑色结点，外部空叶结点可省略不画）。

答案：最终得到的红黑树：（用圆圈表示红色结点，方框表示黑色结点，外部空叶结点省略不画）



4. （4分）利用广义表的Head和Tail运算，把原子d分别从下列广义表中分离出来，L1＝(((((a),b),d),e))；L2＝（a,(b,((d)),e））。

答案：head(tail(head(tail(LS))))

5. （6分）将关键码1，2，3，...，（2k-1）依次插入到一棵初始为空的AVL树中，试证明结果是一棵高度为k的完全满二叉树。

答案：答案：

采用数学归纳法。

1. 当$k=1$时，命题现任成立（只有一个点的情况）

2. 假设，当$k=n$时命题成立（即依次插入$2^{n-1}-1$个关键码递增的结点构成一棵高度为n的完全满二叉树），下面讨论$k=n+1$的情况。

由于关键码的插入顺序单调递增，因此每次插入的新结点的初始位置一定是最右链的叶子的右儿子。

按照插入顺序，当插入到关键码$2^{n-1}-1$时，构成一棵高度为n的完全满二叉

树（归纳假设）。接着，考虑从关键码 2^{n-1} 到 $2^{n-1}+2^{n-2}+1$（共 2^{n-2} 个关键码），它们和根结点的右子树构成了 $2^{n-1}-1$ 个连续递增的关键码，根据归纳假设，插入关键码 $2^{n-1}+2^{n-2}+1$ 后，根结点的右子树成为一棵高度为 n 的完全满二叉树。然后，下一个关键码 $2^{n-1}+2^{n-2}+2$ 破坏了根结点的平衡（右子树比左子树深度大 2），根据 AVL 树的旋转要求，树将以 2^n 为根（之后根结点不再参与旋转，因此这就是最终的根结点），左子树是一棵高度为 n 的完全满二叉树；而右子树几乎是一棵高度为 $n-1$ 的完全满二叉树，唯一差别在于它的最右边还挂着 $2^{n-1}+2^{n-2}+2$ 这个结点。最后，关键码 $2^{n-1}+2^{n-2}+3$ 到 2^n-1 插入到树中，根据归纳假设，它们将把右子树构造成一棵高度为 n 的完全满二叉树。此时，整棵树便是一棵高度为 $n+1$ 的完全满二叉树。

综上，结论成立。

得分

三、算法填空 （18 分）

阅读和完成下面的代码（每空可不限一条语句）。

1. （10 分）以单向链表为存储结构，实现选择排序算法（得到非递减序列）。

```
struct node {
      int key;
      node *next;
};

node* sort_linked_list(node* head) {
   i = head;
   while (i!=NULL) {
         填空 1 node *j = i->next          ;
```

```
        while (j!=NULL) {
          if ( 填空 2 i->key>j->key )  填空 3 swap(i->key,j->key)  ;
            填空 4 j = j->next          ;
          }
              填空 5 i = i->next          ;
        }
        return head;
    }
```

2. （8 分）下面是败者树在内部结点从右分支向上比赛的成员函数实现，请将空缺部分补充完整。

```
template<class T>
void LoserTree<T>::Play(int p, int lc, int rc, int(*winner)(T A[], int b, int c),
int(*loser)(T A[], int b, int c)) {
    B[p] = loser(L, lc, rc);
    int temp1, temp2;
    （1）temp1 = winner(L, lc, rc);
   while (p>1 &&  （2）  p%2   ) {
    temp2 = winner(L, temp1, B[p/2]);
      （3）B[p/2] = loser(L, temp1, B[p/2]);
          temp1 = temp2;
          p/=2;
      }
    （4）B[p/2] = temp1;
}
```

3. （8 分）下述算法实现在图 G 中计算从顶点 *i* 到顶点 *j* 之间长度为 len 的简单路径条数。图的 ADT 如下：

```
Class Graph {
public:
    int VerticesNum();
    int EdgesNum();
```

```
        Edge FirstEdge(int oneVertex);
        Edge NextEdge(Edge preEdge);
        bool IsEdge(Edge onEdge);
        int FromVertex(Edge oneEdge);
        int ToVertex(Edge oneEdge);
    };

int visited[MAXSIZE];                    // 初始化为 0
int GetPathNum_Len(Graph& G, int i, int j, int len)  {
    if (填空 1   i == j && len == 0) return 1;
    sum = 0;                             // sum 表示通过本结点的路径数
    visited[i] = 1;
    for (Edge e = G.FirstEdge(i); G.IsEdge(e); e = G.NextEdge(e)) {
        int v = G.ToVertex(e);
        if (!visited[v])
            填空 2 sum += GetPathNum_Len(G, v, j, len - 1)
    }// for
    visited[i] = 0;          //本题允许曾经被访问过的结点出现在另一条路径中
    return sum;
} // GetPathNum_Len
```

得分

四、算法设计与分析 （25 分）

请尽量按照试题中的要求来写高效率的可读算法。应该申明算法思想，在代码种加以恰当的注释。

1. 伸展树是一种自平衡的BST树，它可以通过旋转来实现树结构的动态调整。

伸展树结点的数据结构定义如下：

```
struct TreeNode{
    int   data;
    TreeNode    * father,* left,* right;
```

};

其成员函数包括：

1）Delete zig(TreeNode* x); 其功能是删除以 *x* 结点为根的子树；

2）Splay(TreeNode* *x* , TreeNode* f)；其功能是将 *x* 旋转为 f 的子结点（f 是 *x* 的祖先）。特殊的，把 *x* 旋转到根结点即 Splay(*x*, NULL)。

请运用上述给定的成员函数，实现删除树rt中所有大于u小于v的结点（假设u, v是Splay树中的结点，且树种所有的节点值不重复）。

函数原型为：void DeleteSubTree(TreeNode* rt, TreeNode* u, TreeNode* v)

解答：把 u 结点旋转到根；把 v 旋转为 u 的右儿子；删除 v 结点的左子树前面几句各 3 分，最后一句 1 分。如果思路、注释各 1 分，不写则扣。

```
void DeleteUV(TreeNode* rt, TreeNode* u, TreeNode* v ) {
    Splay(u, NULL);
    Splay(v, u);
    Delete(v->lchild);
    v->lchild = NULL;
}
```