北京大学信息科学技术学院

2006－2007 秋季学期

数据结构与算法 A 参考答案

任课教师: 张铭、赵海燕、王腾蛟

一、填空（28 分）

1. （3 分）两空各自 1.5 分： n*(n-1)/2，n-1
2. （5 分）V0（1.5 分），V1 和 V3（1.5 分），（用数字表示结点）02413,02431,04213,04231（2 分）
3. （3 分）对字符串数组进行快速排序，第一趟选 ff 为轴值，分割后得到的序列是(A)。
4. （7 分）

1. （7 分）散列表长度 m=13，散列函数为 h1(key) = key % m。处理冲突的方法为双散列法，探查函数为：

p(key，i)=（key %（m-2）+ i * h1（key））% m，i =1，…，m-1。

依次将关键码{34, 96, 12, 8, 37, 35, 22 } 填入散列表。

(1) 解法 1：

（3 分）下表每错一个扣 0.5 分，扣完 3 分为止。

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
37	22				96			34	35		8	12

对散列表中现有元素等概率查找时查找成功的平均查找长度是 1.57 。

计算过程：

h1(34)=8; 查找 1 次

h1(96)=5; 查找 1 次

h1(12)=12; 查找 1 次

h1(8)=8; p(8,1)=3; d1(8)=11; 查找 2 次

h1(37)=11; p(37,1)=2; d1(37)=0; 查找 2 次

h1(35)=9; 查找 1 次

h1(22)=9; p(22,1)=9; d1(22)=5; p(22,2)=5; d2(22)=1; 查找 3 次

(2) 解法 2 如果是下面这个解答（该解答的探测值没有加上原基地址，算半对），则扣 1.5 分

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
22			8		96			34	35		37	12

对散列表中现有元素等概率查找时查找成功的平均查找长度是 1.57 。

平均查找长度为(1 * 5 + 2 * 1 + 4 * 1)/7 = 11 / 7 = 1.57

查找元素 34, 96,12, 37, 35 需要 1 次

查找元素 8 需要 2 次

查找元素 22 需要 4 次

注意：如果不是上述两种解答，而且(1)错了，(2)对了，要注意严查抄袭，并根据学生(1)的结果运算一下。

(1) （3 分）请把关键码填写到下面散列表中相应的位置（散列槽）：

（4 分）对散列表中现有元素等概率查找时查找成功的平均查找长度是_____11/7 （1.57）_____。

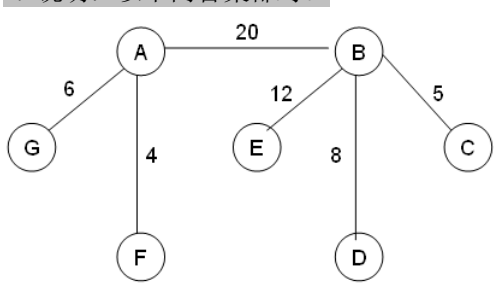
5. (3 分) 外排序是针对___磁盘文件__所进行的排序操作，其算法的主要目标是__尽量减少读写磁盘的次数___。 （每个空各 1.5 分）。

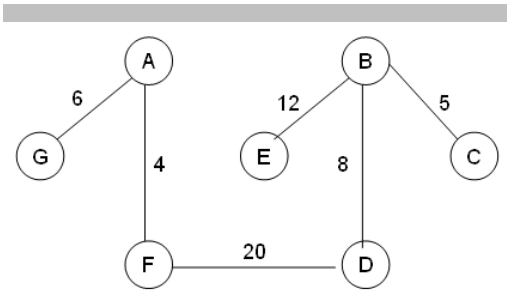
6.（3 分）在一个二级索引中，每一条二级线性索引记录对应于一个一级线性索引文件的磁盘块。磁盘块的大小是 1024 字节，每个关键码 / 指针记录需要 8 个字节。假设线性文件索引中包含 10000 条记录，那么相应的二级线性索引文件中有____79_____项记录。

7.（ 4 分）设某文件经内排序后得到 100 个初始顺串，若使用多路归并排序算法，并要求三趟归并完成排序，问归并路数最少为____5_______。

二、辨析题（26 分）

1. 说明：以下两答案都对。





2.

答案：

	a	b	c	d	e	f	g
初始状态	0 a	∞ a	∞ a	∞ a	∞ a	∞ a	∞ a
进入 a	0 a	15 a	2 a	12 a	∞ a	∞ a	∞ a
进入 c	0 a	15 a	2 a	12 a	10 c	6 c	∞ a
进入 f	0 a	15 a	2 a	11 f	10 c	6 c	16 f
进入 e	0 a	15 a	2 a	11 f	10 c	6 c	16 f
进入 d	0 a	15 a	2 a	11 f	10 c	6 c	14 d
进入 g	0 a	15 a	2 a	11 f	10 c	6 c	14 d

3.（4 分）请简述置换选择排序的基本思想。

评分准则：利用内存的(1)**堆**，对(2)**外存数据块预排序**，以得到(3)**若干个顺串**。为加快内存和外

存交换数据，需要利用(4)**输入/输出缓冲区**。上述关键点都提到，或者虽然不是这样表述，但意思齐全，都给满分。不完整，一个点酌情扣 1 分。参考解答：

作为堆排序的一个变体，置换选择排序的核心思想是利用最小值堆来对读进内存的数据块进行排序，把外存待数据排成若干个顺串文件。在此过程中，需要 2 个辅助的缓冲区：输入缓冲区和输出缓冲区。从输入文件读取记录（一整块磁盘中所有记录），进入输入缓冲区；在 RAM 中放入待排序记录；记录被处理后，写回到输出缓冲区；输出缓冲区写满的时候，把整个缓冲区写回到一个磁盘块；当输入缓冲区为空时，从磁盘输入文件读取下一块记录。在处理过程中，新加入的数据小于刚输出到缓冲区的数据时，该数据不能进入堆，堆的规模缩小 1。当堆规模缩小为 0 时，要重新建立堆，开始新的顺串输出。

还有学生写了整个置换选择排序的算法思想或者伪码，而没有上述 4 点，也可以给满分。

（1）首先 初始化堆

(a) 从磁盘读 M 个记录放到数组 RAM 中

(b) 设置堆末尾标准 LAST ＝ M - 1

(c) 建立一个最小值堆

（2）然后 重复以下步骤，直到堆为空（LAST < 0 ）：

(a) 把具有最小关键码值的记录（根结点）送到输出缓冲区

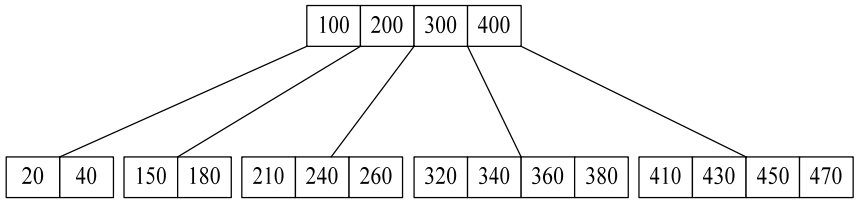
(b) 设 R 是输入缓冲区中的下一条记录。如果 R 的关键码值大于刚刚输出的关键码值……

i. 那么把 R 放到根结点

ii. 否则使用数组中 LAST 位置的记录代替根结点，然后把 R 放到 LAST 位置。设置 LAST = LAST - 1

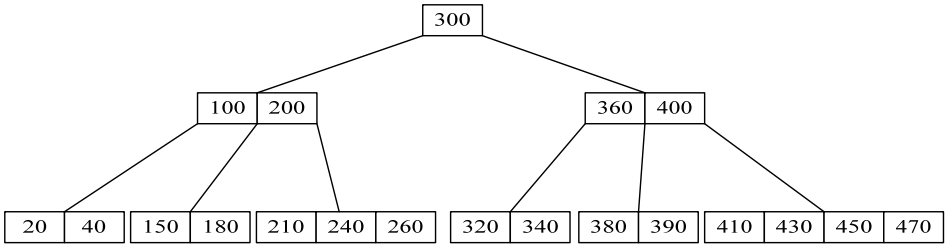
(c) 重新排列堆，筛出根结点

4.（8 分）请画出往下图的 5 阶 B 树中插入一个关键码 390 后得到的 B 树，以及再删除关键码 150 后得到的 B 树。并请分别分析插入和删除时的读写页块次数。（假设在操作之前 B 树所有的结点页块均在外存，删除关键码时，如果需要从兄弟结点借关键码的话，先查看左边的、再查看右边的兄弟。）

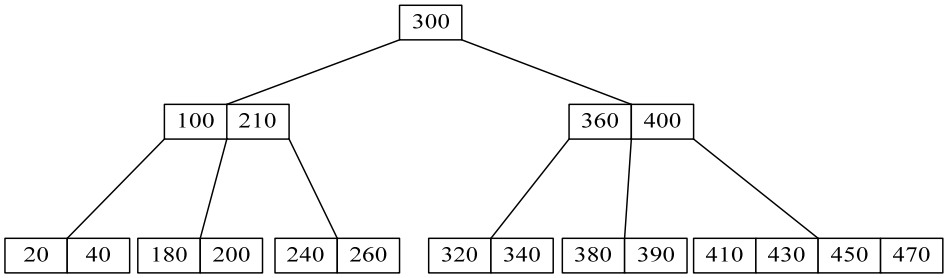


评分准则：

插入 390 后的 B 树 （ 2 分）：



删除 150 后的 B 树 （2 分）：



插入 390 时的读写页块次数： 读 2 次，写 5 次 （ 2 分，读写次数各 1 分）

删除 150 时的读写次数：读 5 次（如果从根开始，假设之前插入时读进，则读 3 次）；写 3 次（ 2 分，读写次数各 1 分）

三、算法填空（16 分），每个空 4 分

① delta/=3

② &Array[j], n-j, deltae

③ ModifiedInsertSort(Array, n, 1)

④ swap(Array, j, j-delta)　　// swap(Aarry[j], Array[j-delta])也可以

四、算法设计（30 分）

本试卷对于算法改进、算法设计都有质量要求，请尽量按照试题中的要求来写算法。否则将酌情扣分。您所写的算法应该申明算法思想，并在算法段加以恰当的注释。

1. 答案：设 C 是有向图 G 的邻接矩阵：

利用 Floyd 算法求出每对顶点之间的最短路径距离矩阵 A

对矩阵 A 求出每列 i 的最大值，得到顶点 i 的偏心度；

在这 n 个顶点的偏心度中，确定具有最小偏心度的顶点 k，这便是有向图 G 的中心点。

```
int    CenterOfGraph( Graph& G )
{
       int** D ;                              //传入 Floyd 算法的参数，求出每个结点之间的最短距离
       int max[G.VertecesNum()];   //记录每个结点到别的结点最短距离的最大值
       int min = 0;                           //记录 max 数组中的最小值
       int pos = 0;                        //记录这个最小值的位置
       for( int i = 0 ; i < G.VertecesNum()    ; i++ )          //初始化 max 数组
              max[i] = -1 ;
       Floyd( G , D );                                          //用 Floyd 算法求每个结点之间的最短距离
       for( int j = 0 ; j < G .VertecesNum() ; j++ )     //找每列的最大值
       {
              for( i = 0 ; i< G. VertecesNum() ; i ++ )
              {
                     if( D[i][j] > max[j] )
                            max[ j ] = D[i][j] ;
              }
       }
       min = max[0];
       for (    i = 0 ; i < G. VertecesNum() ; i++ )          //找 max 数组的最小值
              if( max[ i ] < min )
              {
                     min = max[i];
                     pos = i;
              }
       return pos;                                              //返回的是那个中心点结点的标号
}

void Floyd(Graph& G,int **D)       //Floyd算法
{
       int i,j,v;                          //i,j是计数器，v记录相应顶点
       //初始化D数组
```

```
    for(i=0;i<G.VerticesNum();i++)
    {
        for(j=0;j<G.VerticesNum();j++)
        {
            if(i==j) D[i][j]=0;
            else D[i][j]=INFINITY;
        }
    }
    //D=adj[0]，将边的值填入D数组
    for(v=0;v<G.VerticesNum();v++)
        for( Edge e=G.FirstEdge(v) ; G.IsEdge(e) ; e=G.NextEdge(e) )
            D[v][G.ToVertex(e)]=G.Weight(e);

    //如果两个顶点间的最短路径经过顶点v，则有
    //D[i][j]>(D[i][v]+D[v][j])，通过对v的循环，
    //相当于将图一个顶点一个顶点的扩大
    for(v=0;v<G.VerticesNum();v++)
        for(i=0;i<G.VerticesNum();i++)
            for(j=0;j<G.VerticesNum();j++)
                if(D[i][j]>(D[i][v]+D[v][j]))
                    D[i][j]=D[i][v]+D[v][j];
}//end of Floyd()
```

2. （15 分）有一种简单的排序算法，叫做计数排序（rank sorting），该算法用一个新数组存储原数组各元素排序后的正确下标位置（结果下标）。假设所有待排序的关键码互不相同，算法对每个记录都扫描一趟待排序数组，统计表中有多少个记录的关键码比该记录的关键码小。假设针对某一个记录，统计出的计数值为 c，那么，这个记录在新有序表中的合适的存放位置即为 c。例如，对于排序码{34, 96, 12, 8, 37}，计数排序后的结果如下表所示。

下标	0	1	2	3	4
排序码	34	96	12	8	37
结果下标	2	4	1	0	3

得到这个结果下标后，还需要一个整理的过程，在原来传入待排数据的数组 array 中返回排序后的正确结果，而且把计数结果在 rank 数组中返回。对于上例，按照排序码的顺序依次为 { 8, 12, 34, 37, 96}。

(1) 编写实现计数排序的算法；函数原型为：

void Rank(Record* array, int *rank, int n);

(2) 与简单选择排序相比较，这种方法是否更好？为什么？

注释：待排序的数据类型 Record 是“（key, info）”对。不能简单认为待排数据就是整型。

评分标准：

(1) 算法思想 2 分；注释 2 分；

(2) rank 排序部分 4 分；调整 4 分；

(3) 算法时间、空间代价分析 2 分，结论时空代价“比简单选择排序差”1 分。

（1） 算法部分

算法思想：

1、首先对每个元素计数，数组中所有元素只需两两比较一次即可。

2、按 rank 值调整每个元素的位置，有两种方法：

A：一种是循环调整。循环调整思想：因为 rank 数组中保存着每个元素应在位置，则只需把 array 数组中每个元素按正确位置放置即可，处理时，需要先把正确位置的元素拿出来，依次处理被拿出来的元素，直到所有元素已放到正确位置。

B：一种是交换调整。交换调整思想：把 array 数组中 i 位置的元素与该元素应在位置的元素交换，当被交换元素也不在正确位置时，还需要继续交换，直至所有元素都在正确位置。

A、B 算法本质上没有差别，A 算法是先腾出位置，在逐个填空；B 算法每次都交换。

参考算法：

```
void Rank(Record* array, int* rank, int n)
{
     //初始化下标数组
     for(int i=0; i<n; ++i)
          rank[i]=0;
     //排序
     for(int i=0; i<n; ++i)
       for(int j=i+1; j<n; ++j)
         if(Compare::gt(array[i], array[j]))
          rank[i]++;
         else
          rank[j]++;

        //使用下述两种调整算法之一，调整元素位置（用一种就可以得全分）
     AdjustRecord(array, rank, n);
    // 或者SwapRecord(array, rank, n);
}

//循环调整算法
void AdjustRecord(Record* array, int* prank, int n)
{
     //拷贝一份rank数组
     int* rank=new int[n];
     for(int i=0; i<n; ++i)
          rank[i]=prank[i];

     for(int i=0; i<n; ++i)
     {
          if(rank[i]==i)              //已经在正确位置
               continue;
          int k=rank[i];              //array[i]应在的位置
          Record lastRec=array[i];
          Record nextRec;
          int temp;
          while(rank[k] != k)         //元素不在正确位置
          {
               //保存该位置元素，放入正确元素
               nextRec=array[k];
               array[k]=lastRec;
               lastRec=nextRec;

               temp=rank[k];          //下一步处理该元素
               rank[k]=k;             //标记该位置元素已在正确位置
               k=temp;
          }
     }
     delete []rank;
}
```

```
//交换调整算法
void SwapRecord(Record* array, int* prank, int n)
{
    //拷贝一份rank数组
    int* rank=new int[n];
    for(int i=0; i<n; ++i)
        rank[i]=prank[i];
    for(int i=0; i<n; ++i)
    {
        if(rank[i]==i)              //已经在正确位置
            continue;
        //交换到正确位置
        swap(array[i], array[rank[i]]);
        swap(rank[i], rank[rank[i]]);
        while(rank[i] != i)         //元素不在正确位置
        {
            swap(array[i], array[rank[i]]);
            swap(rank[i], rank[rank[i]]);
        }
    }
    delete []rank;
}
```

(2) 算法分析

数组中所有元素都需两两比较一次，比较时间代价为 O(n^2)。

以上两种调整算法对每个元素（除了第一个元素）都需要移动三次，其中包括移动到临时变量中 2 次，移动到正确位置一次。元素移动时间代价是 O(n)。

辅助空间代价为 O(n)。

与简单选择排序相比较，元素比较次数和交换次数都没有减少，空间代价有增无减，因此从时空代价来说该方法不如简单选择排序。

下述说法不扣分：如果修改一下算法，允许重复关键码，该算法可以达到稳定，应此它比简单选择排序算法好。

阅卷情况：

1. 算法思想和注释各占 2 分，如没写酌情扣分（有的注释写的好的，没算法思想也没扣分）。
2. 计算 rank 数组值部分 4 分；（基本都得到该 4 分）
3. 调整记录部分 4 分，如不完全正确扣 1－3 分。如使用临时记录数组的，如果正确都扣 2 分，不正确扣 4 分。（基本都得了 2 分左右）
4. 没有返回 rank 数组正确值的扣 1 分。（如使用临时数组的，这 1 分肯定得了）
5. 明显程序语法错误的最多扣 1 分，大部分没扣分。
6. 第二问中时间、空间复杂度共 2 分，分析不全的扣 1 分。没有比较该算法与简单选择排序算法优劣的或比较不对的扣 1 分。（这部分得 2 分和 3 分的人数相当）

总的得分情况是，大部分得 11－13 分，个别人得 14 分或者 10 以下。没有得满分的。