哈工大 *2010* 年春季学期
数据结构与算法　　A **试 卷**

| 班号 | |
|---|---|
| 学号 | |
| 姓名 | |

| 题号 | 一 | 二 | 三 | 四 | | | 总分 |
|---|---|---|---|---|---|---|---|
| 分值 | **15** | **15** | **20** | **20** | | | **70** |
| 得分 | | | | | | | |

注意行为规范　遵守考场纪律

一、填空题（每空 1 分，共 15 分）

1. 在顺序存储的二叉树中，编号为 i 和 j 的两个结点处在同一层的条件是_____。
2. 某二叉树的前序遍历序列是 ABCDEFG，中序遍历序列是 CBDAFGE，则其后序遍历序列是_____。
3. 在有 n 个叶子的哈夫曼树中，分支结点总数为_____个。
4. 对于含有 n 个顶点 e 条边的连通图，利用 Prim 算法求最小生成树的时间复杂度为_____。
5. 表达式 a*(b+c)-d 的后缀表达式是_____。
6. 假定一棵二叉树的结点数为 18，则它的最小深度为_____，最大深度为_____。
7. 设有一个 n 阶的下三角矩阵 A，如果按照行的顺序将下三角矩阵中的元素（包括对角线上元素）存放在 n(n+1)个连续的存储单元中，则 A[i][j]与 A[0][0]之间有_____个数据元素。
8. 设一组初始记录关键字序列为(20，18，22，16，30，19)，则根据这些初始关键字序列建成的初始堆为_____。
9. 磁盘文件的归并技术有_____、_____、_____。
10. 设有向图 G 中有向边的集合 E={<1，2>，<2，3>，<1，4>，<4，2>，<4，3>}，则该图的一种拓扑序列为_____。
11. 设一组初始记录关键字序列为(345，253，674，924，627)，则用基数排序需要进行_____趟的分配和回收才能使得初始关键字序列变成有序序列。
12. 利用 Dijkstra 算法求从有向图顶点 v1 到其他各顶点的最短路径要求边上权值_____。

主管领导签字

二、选择题（每题 1 分，共 15 分）

1. 若某线性表最常用的操作是存取任一指定序号的元素和在最后进行插入和删除运算，则利用_____存储方式最节省时间。

A. 顺序表　　B.双链表

C. 单循环链表　　D. 带头结点的双循环链表 0

2．在一个具有 n 个单元的顺序栈中，假定以地址低端（即下标为 0 的单元)作为栈底，以 top 作为栈顶指针，当出栈时，top 的变化为_____。

A. 不变　　B. top=0;　　C.top=top-1;　　D. top=top+1;

3．设一组初始关键字记录关键字为(20，15，14，18，21，36，40，10)，则以 20 为基准记录的一趟快速排序结束后的结果为_____。

A、 10，15，14，18，21，36，40，20

B、 10，15，14，18，20，40，36，21

C、 10，15，14，20，18，40，36，2l

D、 15，10，14，18，20，36，40，21

4．任何一棵二叉树的叶子结点在前序、中序、后序遍历序列中的相对次序_____。

A.肯定不发生改变　B.肯定发生改变

C.不能确定　　D.有时发生变化

5．用有向无环图描述表达式(A+B)*((A+B)/A)，至少需要顶点的数目为_____。

A.5　　B. 6　　C. 8　　D. 9

6. 对线性表进行二分查找时，要求线性表必须_____。

A、以顺序方式存储

B、以链接方式存储

C、以顺序方式存储，且数据元素有序

D、以链接方式存储，且数据方式有序

7. 设散列表表长 m=14，散列函数 H(k)=k mod 11。表中已有 15、38、61、84 四个元素，如果用线性探侧法处理冲突，则元素 49 的存储地址是_____。

A. 8　　B. 3　　C. 5　　D. 9

8. 若需在 $O(n\log_2 n)$的时间内完成对数组的排序，且要求排序是稳定的，则可选择的排序方法是_____。

A.快速排序　　B. 堆排序

C.归并排序　　D. 插入排序

9. 下面关于 m 阶 B 树的说法正确的是_____。

① 每个结点至少有两株非空子树

② 树中每个结点至多有 m-1 个关键字

③ 所有的叶子都在同一层上

④ 当插入一个记录引起 B 树分裂后，树增高一层

A. ①②③　　B.②③　　C. ②③④　　D. ①③

10. 已知一个有序表为（12，18，24，35，47，50，62，83，90，115，134），当折半查找值为 90 的元素时，经过_____次比较后查找成功。

A.2　　B.3　　C.4　　D.5

11.能有效缩短关键路径长度的方法是_____。

A．　缩短任意一个活动的持续时间

B．　缩短关键路径上任意一个关键活动的持续时间

C．　缩短多条关键路径上共有的任意一个关键活动的持续时间

D．　缩短所有关键路径上共有的任意一个关键活动的持续时间

12.在采用线性探测法处理冲突所构成的闭散列表上进行查找，可能要探测多个位置，在查找成功的情况下，所探测的这些位置的关键字值_____。

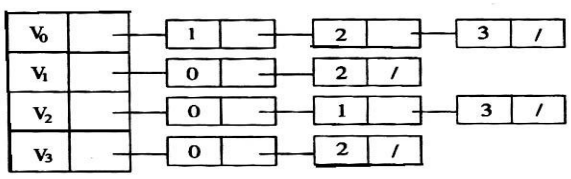
A.一定都是同义词　　B.一定都不是同义词

C.不一定都是同义词　　D.都相同

13.设哈夫曼编码的长度不超过 4，若已对两个字符编码为 1 和 01，则最多还可以对_____个字符编码。

A．2　　B．3　　C．4　　D．5

14.已知图的邻接表如下所示，根据算法，则从顶点 0 出发深度优先遍历的结点序列是_____。




A.0 1 3 2　　B. 0 2 3 1　　C. 0 3 2 1　　D. 0 1 2 3

15. 在具有 n 个结点的有序单链表中插入一个新结点并仍然有序的时间复杂度是_____。

A.O(1)　　B.O(n)　　C.O($n^2$)　　D.O($n\log_2 n$)

三、简答题：**每题 10 分，共 20 分**

. 1．一个按数组元素有序的一维数组一定是堆吗？请说明理由。

2. 设有一组初始记录关键字为(45，80，48，40，22，78)，可以构造出一棵二叉排序树，若不是平衡树则调整平衡，并给出其前序遍历该树的序列，并写出右旋转函数算法。

四、**算法设计：每题 10 分，共 20 分**

要求：

⑴描述算法设计的基本思想

⑵描述算法的详细实现步骤

⑶根据设计思想和实现步骤，采用程序设计语言描述算法（使用 C 或 C++ 或 JAVA 语言实），关键之处请给出简要注释。

（栈、队列的存储结构、基本操作可以直接引用）

1．对给定的序号 j（1＜j＜n），要求在无序记录 A[1]~A[n]中找到按关键码从小到大排在第 j 位上的记录，试利用快速排序的划分思想设计算法实现上述查找。

2．设计算法，判断以邻接表存储的有向图中是否存在由顶点 vi 到顶点 vj 的路径（i≠j）。

参考答案：

一．填空：1.logi=logj　2.CDBGFEA　　3. n-1　4. O(n2)　5.abc+*d-

6.　5　　18　　7.i(i+1)/2+j-1　　8. 16 18 19 20 30 22　　9.多路归并、I/O 并行处理　初始归并段产生　10.1,4,2,3　11.　3　　12.非负

二、单选：1A2C3A4A5A6C7A8C9B10A11D12C13C14D15B

三、简答：

1. 按数组元素有序的一维数组一定是堆。（4 分）

非升序数组一定是最小堆为例说明如下：假设非升序数组为 $K_1, K_2, \ldots, K_n$，则满足 $K_1 \leq K_2, \leq \ldots \leq K_n$，则一定满足：$K_i \leq K_{2i}$ 且 $K_i \leq K_{2i+1}$，即满足最小堆的定义。同理可知，非降序数组一定是最大堆。因此，按数组元素有序的一维数组一定是堆。（6 分）

2. 二叉平衡树为 48，40，80，22，45，78　（3 分）

前序：48，40，22，45，80，78　　（3 分）

右旋转函数：**void R_Rotate(BSTree &p)**

```
{   lc=p->lchild;
    p->lchild=lc->rchild;
    lc->rchild=p;
    p=lc;
}
```

**四、算法**

**1.** 1、本算法不要求将整个记录进行排序，而只进行查找第 j 个记录。

(1)基本思想：改进划分算法，是一次划分将基准元素定位于 k，如果 k==j，则找到第 j 小的元素；否则，递归地在 k 的左边或右边进行划分，直到 k==j 为止。

(2) 算法详细步骤：略

(3) 算法如下：

```
int Search( int A[ ], int n, int j )
{
    s = 1;   t = n ;
```

```
    k = Partition( A, s, t);
    while ( k ！= j )
        if ( k < j )    k = Partition (A, k+1， t);
        else       k = Partition (A,    s，k-1 );
    return    A[j];
}
int   Partition (int A[ ], int low ,int high )
{
    i = low; j = high; pivot = A [ low ];
    while ( i < j )
    {
        while ( A[j] >= pivot && i < j )    j --;
        if (i < j )
            A[ i++ ] = A[ j ];
        while ( A[j] < pivot && i < j )    i ++;
        if (i < j )
            A[ j-- ] = A[ i ];
    }
    A[ i ] = pivot;
    return    i;
}
```

2. int visited[MAXSIZE]; //指示顶点是否在当前路径上

```
int exist_path_DFS(ALGraph G,int i,int j)
{
if(i==j) return 1; //i 就是 j
else
{     visited[i]=1;
   for(p=G.vertices[i].firstarc;p;p=p->nextarc)
   {
      k=p->adjvex;
      if(!visited[k])
            return exist_path(k,j);//i 下游的顶点到 j 有路径
   }//for
   return 0;
}//else
}//exist_path_DFS
```