一、**回答下列问题（14 分）**

1．下列几种排序法中，要求空间最大的是_____。

（A） 插入(insertion)排序； （B） 选择(selection)排序；

（C） 快速(quick)排序； （D） 归并(merge)排序。

2．将 5 个字母 ‘ooops’ 按此顺序入栈，则有_____种不同的出栈顺序可以仍然得到 ‘ooops’。

（A） 1 （B） 3 （C） 5 （D） 6

3．已知一棵二叉树的前序遍历结果为 ABCDEF，中序遍历结果为 CBAEDF，则后序遍历的结果为：

（A） CBEFDA （B） FEDCBA （C） CBEDFA （D） 不定

4．在下列查找的方法中，平均查找长度与结点个数 n 无关的查找方法是_____。

（A） 顺序查找；（B） 二分法；（C） 利用二叉搜索树；（D） 利用哈希(hash)表。

5．一棵树有 n_1 个孩子数为 1 的结点，n_2个孩子数为 2 的结点，……，n_m个孩子数为m的结点，则该树的叶结点数为_____。

6．前序遍历和中序遍历结果相同的二叉树为_____；前序遍历和后序遍历结果相同的二叉树为_____。

（A）一般二叉树 （B）只有根结点的二叉树

（C）根结点无左孩子的二叉树 （D）根结点无右孩子的二叉树

（E）所有结点只有左子树的二叉树 （F）所有结点只有右子树的二叉树

二、**右图所示为一有向图，请给出该图的下述要求：**

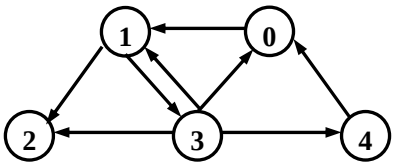
(1)每个顶点的入/出度。（2 分）

(2)邻接矩阵。（3 分）

(3)邻接表。（3 分）

(4)逆邻接表。（2 分）

(5)强连通分量。（2 分）



三、．**已知无向图如下，试给出以 A 为顶点访问序列起点（同一顶点的多个邻点，按字母顺序访问）的，**

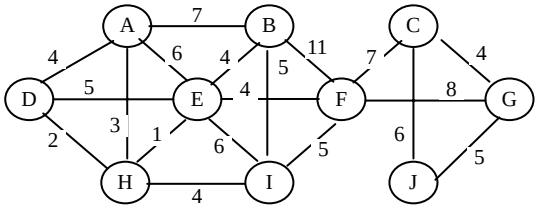
（1）深度优先(depth-first) 遍历序列；（3 分）

（2）深度优先生成树；（3 分）

（3）广度优先(width-first) 遍历序列；（3 分）

（4）广度优先生成树。（3 分）

（5）最小成本生成树。（3 分）



四、**已知线性表{42，26 ，8，70，102，6，56，2，88，80，35}**

（1） 按表中元素顺序依次插入一棵初始为空的二叉排序树，请画出插入完成后的二叉排序树；（8 分）

（2） 按表中元素顺序构造一棵二叉平衡树（AVL 树）。（8 分）

五、**堆排序 heapsort 是通过不断地调整堆（heap）来达到排序目的，其函数如下**

```
voide heapsort (element list[ ], int n)
{
    int    i, j;
    elment temp;
    for ( i= n/2 ; i> 0;    i--)
        adjust(list, i, n);
    for ( i= n-1; i> 0;    i--) {
        swap (list[1], list[i+1],    temp)
        adjust (last, 1, i);
    }
}
```

请写出调整堆的函数 void adjust (element list[ ], int root, int n) (16 分)

六．**阅读如下两个程序，指出其功能，并在空格处填上适当语句。**(12 分)

（1）

```
void test2(element item, list_pointer ht[])
    {
        int hash_value = hash( item.key) ;
        list_pointer ptr, trail = NULL, lead = ht[hash_value];
        for ( ; lead; trail = lead, lead = lead->link) {
            if (!strcmp(lead->item.key, item.key)) {
                fprintf(stderr, "The key is in the table\n");
                exit(1);
            }
      }
        ptr = (list_pointer)malloc(sizeof(list));
        if (IS_FULL(ptr)) {
            fprintf(stderr, "The memory is full\n);
            exit(1);
        }
        ptr->item = item;
        ptr->link = NULL;
        if (trail)
        _____________________    ;
        else
        _____________________________    ;
    }
```

（2）**void merge_pass** (element list[], element sorted []. int n, int length)

```
{ int i, j :
  for (i =0; i <= n-2*length; i += 2*length)
        merge (list, sorted, i ,_____, i +2*length -1);
  if ( i+length < n)
        merge (list, sorted, i, i+length-1; n - 1);
  else
      for ( j= i; j< n, j++)    sorted[j] = list[j];
 }
  void    merge_sort (element list[], int n)
  { int length=1;
    element    extra[MAX_SIZE]      ;
    while    (length < n) {
        merge_pass(list, extra, n, length);
        length *=2;
        merge_pass (_____);
        length *=2;
}
```

其中 void merge(element list[], elemet sorted[], int i , int m, int n) 是将两个已排好序的 list[i]…..list[m],与 list[m+1]……list[n]归并，并将结果放在 sorted[i]…….sorted[n]中。

七、**考察有限期的作业调度问题：给定 4 个作业和 1 个处理机，作业的处理时间 t、期限 d、收益 p 在下表列出，图为限界剪枝法的状态树。现设剪枝上界 U 为当前结点可能失去的最大收益（例如根结点有 U = 14，其最左边孩子结点有 U = 9），剪枝下界 C 为当前结点肯定要失去的收益（例如根结点有 C = 0，其最右边孩子结点有 C = 10）。请将树中按宽度优先搜索被剪的结点涂黑，写出最大收益，以及对应的作业序列。**（15 分）

i	1	2	3	4
t	2	3	1	2
d	2	5	1	3
p	5	3	2	4