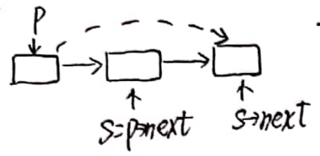
2015年830真题答案.

一、选择题

1.D(同2014-6)

2.C (同2010-9)

3. C.

→ 

→ ① s=p→next
② p→next=s→next
③ free(s)

4. B
→ 非0元素个数表示图中边数.

5. C.(同2014-9)

6. A
→ 判断有环与否 { ① 拓扑排序 ② 深度优先遍历

7.
→ 这题争议在"确保"上，要确保的话要完全无向图，本来的题意是考查无向图的最小生成树，即n个顶点最小生成树边数为n-1

8. B (同2011-10)

9. A
→ 这里考查了树的性质：总分支数 = 总结点数 - 1
→ 二叉树中一个分支占有一个指针域，是n-1个分支.
→ 扩展n个结点的二叉树中空链域个数为 n+1 (2n-(n+1)).

10. C (同2013-3)

11. A
→ 完全二叉树中只有一个孩子的分支结点数为0或1.

12. C
→ 考查了插入排序定义

13. B

14. A

15. A
→ 建图时间复杂度 < 邻接表 O(n+e)；邻接矩阵 O(n^2)

二、填空题.

1. 从任一结点出发可以访问整个链表.

2. 集合　图形式网状结构

3. 5　20　(考查了完全二叉树编号规律，从0开始，i 的孩子为 2i+1、2i+2，从1开始，i 的孩子为 2i、2i+1)

4. $O(N^2)$

→ $O(N*(N-1)) = O(N^2)$

5. $2^h - 1$

6. m $\lceil m/2 \rceil$

7. ① s→next = p→next ② p→next = s

三. 判断题.

1. t (区分数据元素与表指改)
2. f
3. t
4. f (不唯一)
5. t
6. f (根结点可以为1个关键码)
7. t
8. t
9. f
10. t

四. 简答题

1. 答: 中序遍历二叉树, 访问结点并打印结点的数据.

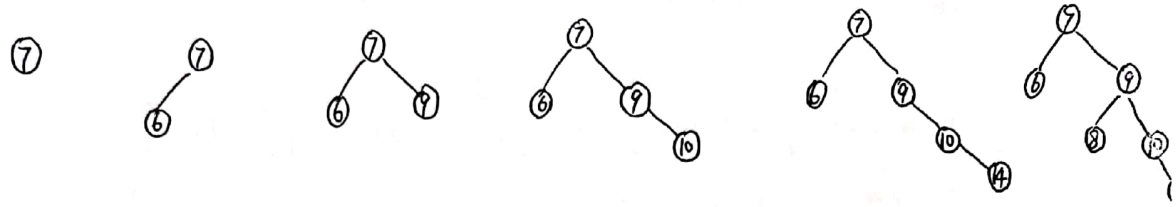
→ 这是中序遍历二叉树的"非递归"算法

```
if(p){
    push(&S, p); p=p->lchild; // 沿根的左子树一直遍历左子树, 途中结点进栈
}
else{
    pop(&S, p); printf(p->data); // 走到左端尽头, 出栈该结点并打印
    p=p->rchild; // 访问其右子树
}
```

→ "递归"与非递归"区别:

其实原理都是栈, 不过一个是系统栈, 可以理解为通用栈; 一个是用户自定义的栈, 可以理解为专用栈, 后者效率更高.

二叉排序树集合 {7} {7,6} {7,6,9} {7,6,9,10} {7,6,9,10,14} {7,6,9,10,14,8,15}



→ 这题比较简单，直接找到插入位置插入就好，插入的新结点一定是叶子结点。

→ 注意，这里并不需要调整树形，平衡二叉树构造时就要考虑调整树形

→ 验证构造的二叉排序树是否正确：看它中序遍历是否为递增序列

3.

原始序列： 63 55 48 37 20 90 84 32

第一趟结果： 32 55 48 37 20 63 84 90

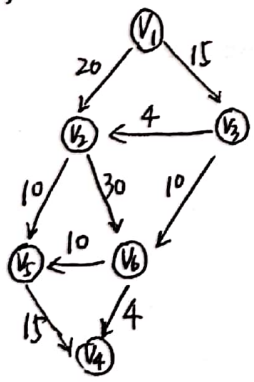
第二趟结果： 20 32 48 37 55 63 84 90

第三趟结果： 20 32 37 48 55 63 84 90

→ 这是快速排序的过程，比较简单，答案中下划线是该趟排序的枢轴，每一趟处理的子序列个数不一样（变化为1,2,4,…），快速排序的趟数与初始序列有关。时间复杂度最好 $O(n\log_2 n)$ 最坏 $O(n^2)$

4. 解：

(1)



(2) 拓扑序列为

$V_1, V_3, V_2, V_6, V_5, V_4$ （唯一）

(3).

根据 Dijkstra 算法，求顶点 V_1 到其余各顶点的最短路径如下表所示

顶点 V_1 先加入集合 S，此时 $S=\{V_1\}$，下表为其余各顶点加入后路径变化

顶点	从顶点 V_1 到各顶点的 dist 值				
	第1趟	第2趟	第3趟	第4趟	第5趟
V_2	20 $V_1\to V_2$	19 $V_1\to V_3\to V_2$			
V_3	15 $V_1\to V_3$				
V_4	∞	∞	∞	29 $V_1\to V_3\to V_6\to V_4$	
V_5	∞	∞	29 $V_1\to V_3\to V_2\to V_5$	29 $V_1\to V_3\to V_2\to V_5$	29 $V_1\to V_3\to V_2\to V_5$
V_6	∞	25 $V_1\to V_3\to V_6$	25 $V_1\to V_3\to V_6$		
集合S	$\{V_1,V_3\}$	$\{V_1,V_3,V_2\}$	$\{V_1,V_3,V_2,V_6\}$	$\{V_1,V_3,V_2,V_6,V_4\}$	$\{V_1,V_3,V_2,V_6,V_4,V_5\}$

由上表可知

V_1 到 V_2 最短路径为 $V_1\to V_3\to V_2$

V_1 到 V_3 最短路径为 $V_1\to V_3$

V_1 到 V_4 最短路径为 $V_1\to V_3\to V_6\to V_4$

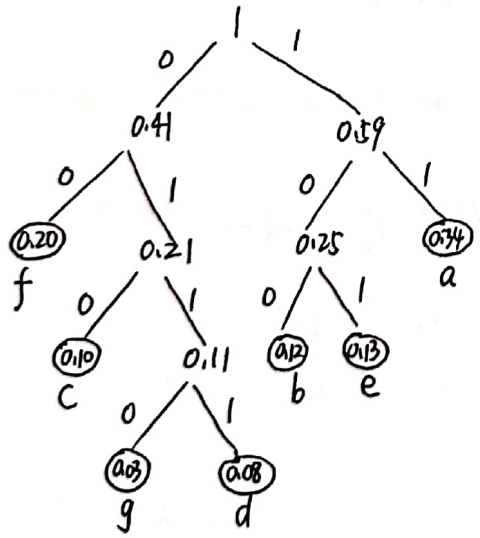
V_1 到 V_5 最短路径为 $V_1\to V_3\to V_2\to V_5$

→ 参考 2013-1

5.

解：

哈夫曼树如下：



各字母前缀编码为：

a：11

b：100

c：010

d：0111

e：101

f：00

g：0110

→ 具体过程见 2010-5，哈夫曼树不唯一，但一般要求左<右.

算法填空

1. ① !P

② e

③ NULL

④ Q.rear->next = P

⑤ P

→素材来自于王版<<数据结构>> P62，一毛一样!!!

2.

⑥ V<G.vexnum

⑦ !visit[V]

⑧ TRUE

⑨ W≥0

⑩ DFSCG,W)

→素材来源于王版<<数据结构>> P169，也是一毛一样!!!

编写算法.(25分)

1.

```
void Delete_List( LNode *head, int X, int Y)
{
    LNode *p, *q;
    p = head->next;
    while( p != NULL)
    {
        if(
        p = head;
        while( p->next != NULL)
        { if( p->next->data < Y && p->next->data > X)
          { q = p->next;
            p->next = q->next //删除q
            free(q)
          }
          p = p->next // 指针后移
        }
    }
```

2.

```
int BFS ( AGraph *G, int v )
{
    ArcNode *p;
    int que [maxsize], front = 0, rear=0;
    int visit [maxsize];
    int i,j ;
    for (i=0 ;i< G->n ;++i)  visit[i]= 0;
    rear =(rear+1)%maxsize ;  // 入队
    que [ rear] = v
    visit [v] = 1 ;
    while ( front != rear)
    {   front =(front+1)%maxsize ;  // 出队
        j = que [front] ;
        p= G->adjlist [j].firstarc ; //依附于j的第一条弧
        while (p != NULL)
        {
            if ( visit [p->adjvex] ==0)
            {   visit [p->adjvex] = 1 ;
                rear =(rear+1) % maxsize;
                que [rear] = p->adjvex ;
            }
            p= p->nextarc ;
        }
    }
    return j ;
}
```

→ 本答案来自19版《天勤》，按天勤说法：广度优先遍历过程中最后一个顶点一定是距离给定顶点最远的顶点（广度优先从V开始遍历），代码改编于广度优先遍历，比较简单。《天勤》最大的优点就是"通俗易懂"，《王道》代码相当精简，但理解难度大，大家依自身情况选择记哪种风格代码！