2011年830共处合集

一、选择题

1. C

→ 算法分析是指对一个算法的运行时间和占有空间做定量分析，目标是降低时间复杂度与空间复杂度.

2. A

→ $O(1) \le O(\log_2 n) \le O(n) \le O(n\log_2 n) \le O(n^2) \le O(n^3) \le \dots O(n^k) \le O(2^n)$

这题取 $n \to +\infty$，$T1(n) \approx \log_{5000} n$，$T2(n) \approx n^2$，$T3(n) \approx n^3$，$T4(n) \approx 2n\log_2 n$

3. C

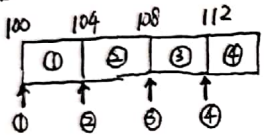
→ 线性表 { 顺序表：便于存取；链表：便于插入删除

4. D.

→ 栈特性：先进后出

→ 分析A项：3出栈→2出栈 { ① → 1出栈；② → 4出栈 }，不可能6出栈，其他错误选项同理.

5. B

→ 

6. D

→ 无论先序还是后序，都是先访问左子树再访问右子树.（递归思想去理解）

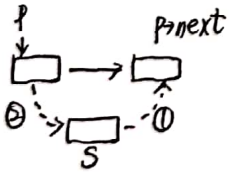
7. C （同2010-10→表示与2010年选择题第10题类似，重大题有很多题会重复考）

→ 考查了满二叉树性质，$h=5$ 的满二叉树有 $2^5-1=31$ 个结点.

8. C.

→ 邻接矩阵适用于表示稠密图（稀疏图的话浪费空间，所以有矩阵压缩这一说）

9. D

→ 

，这里先用 p->next，不然断链后找不到该结点.，即 S->next = p->next

② p->next = S

10. C

→ 折半查找必须采用顺序存储+有序

11. A

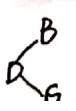
→ 先序序列确定根结点 → 中序序列确定左右分支（用递归思想循环）

先序：ABDGCFK → 根 root = A，中序：DGBAFCK →

→ 再对 DGB 和 FCK 依次上述操作

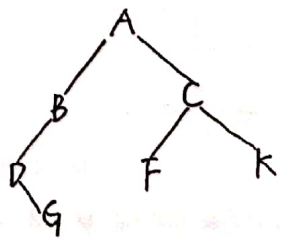
(DGB) 在先序中序列为 BDG，→ root = B

(DGB) 在中序中序列为 DGB →



同理 (FCK) →

→ 后序：GDBFKCA



扫描全能王 创建

12. D

→ 考查了对称矩阵存储问题. 任意下标(i,j),对称矩阵对应公式为

$$k=\begin{cases}\frac{i(i-1)}{2}+j-1, & i\geq j\\ \frac{j(j-1)}{2}+i-1, & i<j\end{cases}$$

,注意,这里存储的一维数组从0开始存储,若从1开始则上述公式加1即可

13. D

→ 希尔排序本质还是插入排序,即插入排序原理是从后往前找插入位置的,明显与初始排列有关排除A,C. B改就很容易排除,若(5,4,3,2,1)与(1,2,3,4,5)要求从小到大排序,比较次数明显前者多,D改原理从待排序列中挑出一个最大或最小放到已排序列中,你只有遍历完全部待排序列元素才可以找到最值.故选D.

14.

→ 这题题目有歧义,"确保"的话必须是5个顶点完全连通(任意两顶点皆有连线),第6个顶点与其中任意一个顶点相连即可满足条件→$C_5^2+1=11$种

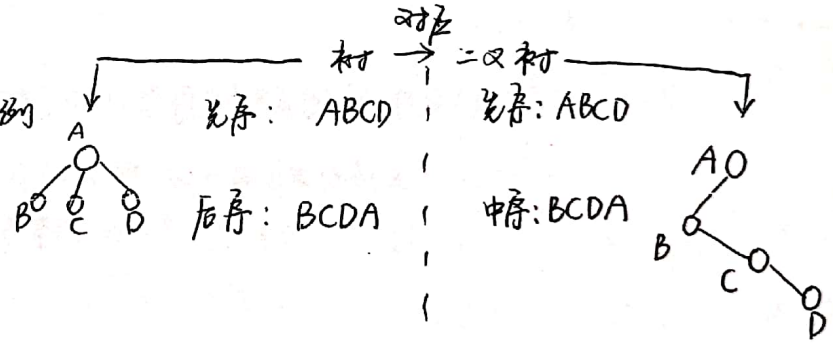
→ 题干本意是考查无向图的极小连通子图(即生成树),应选A(n-1)

15. A

→ 树 二叉树

先序遍历 ↔ 先序遍历

后序遍历 ↔ 中序遍历



二. 填空题

1. 线性结构, 非线性结构

2. 8 9 4 3 6 2 12 13 19 18

3. 完全二叉树 小顶

4. 89 ($n_2=n_0-1=29$, $n_0+n_1+n_2=89$)

5. 元素进栈 栈顶指针增1

6. 只有个元素

7. n-1

8. k_2 (有向图中,度=入度+出度,邻接表中顶点对应的单链表中边节点数是该顶点的出度,逆邻接表的则是入度)

9 (b) (注:顺序表表尾是表,表头为第一个元素,所以答案是(b)不是b)

扫描全能王 创建

10. $\frac{n(n-1)}{2}$ （当序列有序时，比较次数为 $\frac{n(n-1)}{2}$）

三. 判断题.

1. t.（这是队列表表尾定义）
2. f
3. t
4. f（只与顶点数n有关）
5. t
6. t（线性表都可以顺序存取，栈与队列是限上的线性表）
7. t
8. f（B+树根结点范围 $1 \le n \le m$，n是结点内关键字个数，m是阶数）.
9. f
10. f.

四. 简答题

1.

初始序列： 49 38 65 97 75 13 27 51 55 10

第一趟： 13 27 51 55 10 49 38 65 97 75

(d=5)

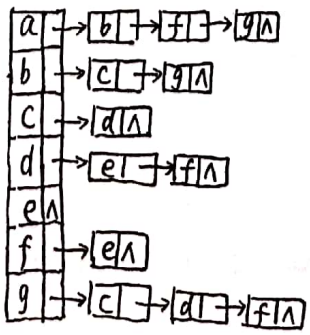
元素移动次数为5次.

2.

(1) 邻接矩阵为

	a	b	c	d	e	f	g
a	0	1	0	0	0	1	1
b	0	0	1	0	0	0	1
c	0	0	0	1	0	0	0
d	0	0	0	0	1	1	0
e	0	0	0	0	0	0	0
f	0	0	0	0	1	0	0
g	0	0	1	1	0	1	0

邻接表为



扫描全能王 创建

10. $\frac{n(n-1)}{2}$ （当序列有序时，比较次数为 $\frac{n(n-1)}{2}$）

三. 判断题.

1. t. （这是队表表尾定义）
2. f
3. t
4. f （只与顶点数n有关）
5. t
6. t （线性表都可以顺序存取，栈与队列是限上的线性表）
7. t
8. f （B+树根结点范围 $1 \le n \le m$，n是结点内关键字个数，m是阶数）.
9. f
10. f.

四. 简答题

1.

初始序列： 49 38 65 97 75 13 27 51 55 10

第一趟： 13 27 51 55 10 49 38 65 97 75

(d=5)

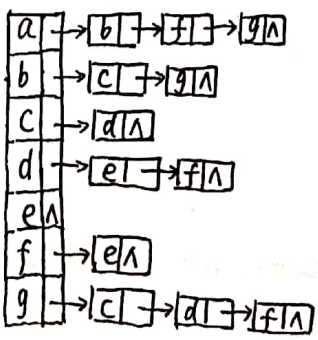
元素移动次数为5次.

2.

(1) 邻接矩阵为

	a	b	c	d	e	f	g
a	0	1	0	0	0	1	1
b	0	0	1	0	0	0	1
c	0	0	0	1	0	0	0
d	0	0	0	0	1	1	0
e	0	0	0	0	0	0	0
f	0	0	0	0	1	0	0
g	0	0	1	1	0	1	0

邻接表为



扫描全能王 创建

(2) 拓扑序列为： a, b, g, c, d, f, e

3.

解：由题意可知

设 ve 表示最早发生时间，vl 表示事件最迟发生时间

事件 事件

	a	b	c	d	e	f	g	h	k
ve(i)	0	6	4	5	7	7	15	14	18
vl(i)	0	6	6	8	7	10	16	14	18

设活动最早发生时间为e，最迟发生时间为l，时间余量为 d=l-e

	a→b	a→c	a→d	b→e	c→e	d→f	e→g	e→h	f→h	g→k	h→k
e(i)	0	0	0	6	4	5	7	7	7	15	14
l(i)	0	2	3	6	6	8	8	7	10	16	14
l(i)-e(i)	0	2	3	0	2	3	1	0	3	1	0

由上表可知，关键活动为 a→b, b→e, e→h, h→k

关键路径为 a→b→e→h→k

→ 解析：上述写法为这类题最标准的写法了

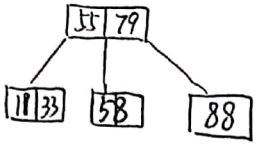
→ 事件 {
ve（最早发生时间）= max{ve(j) + weight(Vj, Vk)}（从前往后）
vl（最迟发生时间）= min{vl(k) - weight(Vj, Vk)}（从后往前）

→ 活动 {
e（最早开始时间）：该活动起点事件最早发生时间.
l（最迟开始时间）：l(i) = vl(j) - weight(Vk, Vj)

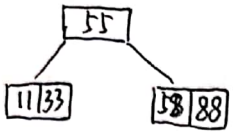
扫描全能王 创建

4.

删除结点44后：



删除79后：

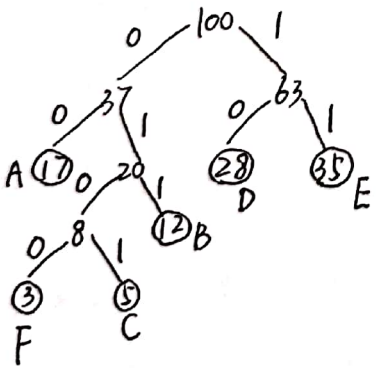


→ 考查了B⁻树的删除（m阶B⁻树）

删除的关键字K
- → 不在终端结点上
  - ① 若小于K的子树中关键字个数 > ⌈m/2⌉−1，则找出K的前驱K'，并且用K'来取代K，再递归删除K'即可
  - ② 若大于K的子树中关键字个数 > ⌈m/2⌉−1，则找出K的后继K'，并且用K'来代替K，再递归删除K'即可
  - ③ 若前后两个子树中关键字均为⌈m/2⌉−1，则直接合并两个子结点，直接删除K即可.
- → 在终端结点上
  - ① K所在结点关键字个数 > ⌈m/2⌉−1 → 直接删除
  - ② K所在结点关键字个数 = ⌈m/2⌉−1
    - a. 相邻左右兄弟结点关键字个数 ≥ ⌈m/2⌉，调整左右兄弟及其双亲结点，以达到平衡
    - b. 兄弟结点关键字个数 = ⌈m/2⌉−1，合并左右兄弟及双亲，再调整

5.

(1) Huffman树如下：



(2) 每个字符的Huffman编码如下：

A：00
B：011
C：0101
D：10
~~F：0100~~
E：11
F：0100

(3)

$$\frac{17\times2+12\times3+5\times4+28\times2+35\times2+3\times4}{8}=28.5$$

(4) 由(2)中，对应字母序列为 BCBAE

→ 解析：第4问要基于一个原则：哈夫曼编码是最短前缀码（任一字符的编码串都不是另一个字符编码串的前缀）.

扫描全能王 创建

6.

解：由 H(key) = key % 11 可得

Hash函数值表为

key	47	7	29	11	16	92	22	8	3
Hash(key)	3	7	7	0	5	4	0	8	3

采用线性探测再散列方法可得各关键字存储地址

地址	0	1	2	3	4	5	6	7	8	9	10
key	11	22		47	92	16	3	7	29	8	

各关键字查找时的比较次数：

key	47	7	29	11	16	92	22	8	3
次数	1	1	2	1	1	1	2	2	4

$$ASL_{成} = \frac{1+1+2+1+1+1+1+2+2+4}{9} = \frac{15}{9}$$

扫描全能王 创建

五. 算法设计

1.

(1) p->next->data <= mink　// 找到第1个大于mink的数

(2) p->next

(3) p->next=q　// 连接断开的链

2.

(4) V < G.vexnum

(5) Visit(V)

(6) ! QueueEmpty(Q)

(7) DeQueue(&Q,u)

(8) EnQueue(&Q,w)

→ 本题素材来自于严蔚敏数据结构P170的算法7.6，这是广度优先的非递归算法，稍微有些不同的 (7)(8) 问严版缺少"&"，而题干的 EnQueue(&Q,u) 有"&"，于是我们"入乡随俗"进队出队那里皆加上"&".

扫描全能王 创建

六. 编写算法（24分）

1. 后序遍历二叉树算法如下：

```
void postorder(BTNode *p)
{
    if(p!=NULL)
    {
        postorder(p->lchild);
        postorder(p->rchild);
        Visit(p);
    }
}
```

→ 答案来自19版<<天勤>>，太简单了，不多做解释。

2. 解：

用邻接矩阵存储的图的kruskal算法如下：

```
typedef struct
{
    int a,b;   // 一条边两个顶点
    int w;     // 权值
}Road;
Road road[maxsize];
int V[maxsize]; // 并查集数组
int getRoot(int a) //从并查集中查找根结点
{
    while(a!=V[a]) a=V[a];
    return a;
}
void kruskal(MGraph g, int &sum, Road road[])
{
    int i;
    int N,E,a,b;
    N=g.n;
    E=g.e;
    sum=0
    for(i=0;i<N;++i) V[i]=i;
    sort(road,E); //对边按权值排序
    for(i=0;i<E;++i){
        a=getRoot(road[i].a)
        b=getRoot(road[i].b)
        if(a!=b){
            V[a]=b;
            sum+=road[i].w
        }
```

答案来自于19版<<天勤>>

解释一下这大段代码

先明确一点，kruskal算法原理是每次挑选权值最小的边，且该边加入后生成树中无环，则将该边加入生成树

答案中sort()用于给边按权值排序

每次挑一条权值最小的边，a!=b这个条件是判断加入该边后生成树否形成环。

若大家先有了并查集，这里不展开说明。这里先把该并查集当作一个集合即可，存储的是一个个子图（树）

getRoot()作用是找到某点的所在子图的根结点，倘若发现一条边的两个顶点所在子树的根结点一样，说明两点在同一个子树中，如果这两个顶点相连，那么该子树会形成环，违背kruskal原理，如果不在同一个子树中，则只需将两个子树连起来即可（V[a]=b）

扫描全能王 创建