2014年890真题答案

一、选择题

1. C
→ 数据结构三要素 { 逻辑结构 / 存储结构（物理结构） / 数据运算

2. A

3. A
→ 举例：长度为5的顺序表(1,2,3,4,5)删除第3个元素，需要将4,5依次前移 → n=5, i=3, n-i=2.

4. B（同2010-1）

5. D（同2010-17）

6. D
→ 顺序存储要求存储空间连续，链式存储无要求

7. C（同2012-2）

8. C.
→ 总查找次数 = $1+2+\cdots+n=\frac{n(n+1)}{2}$，每个元素平均查找长度为 $\frac{n+1}{2}$

9. C
A. O(logn)　B. O(1)　C. O(n)　D. O(1)

10.（同2011-14）

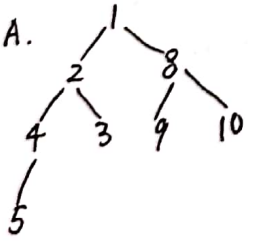
11. A（同2010-4）

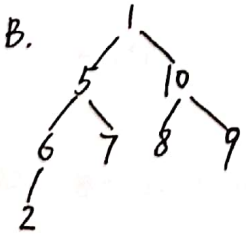
12. D（同2010-19）

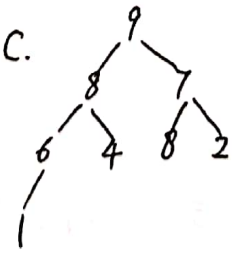
13. B（同2010-13）

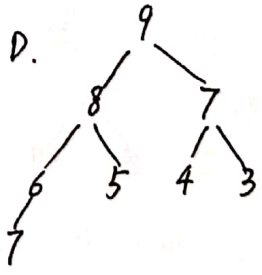
14. A（同2010-20）

15. A
→ 堆只有两种：大顶堆，小顶堆

A.


B.


C.


D.


→ 只有A为小顶堆.

填空题

1. 一对多　多对多

2. 方便统一链表的插入与删除操作（若无头结点，则第一个元素的插入与删除与其他元素有点不一样）

3. 2n-1

4. 将第i行非零元素个数为0 或第i行元素非零元素个数为∞ （前者针对无权图，后者针对有权图）

5. 队尾 队头 先进先出

6. 找出散列地址 解决地址冲突

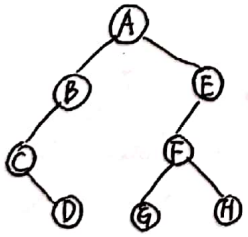
三. 判断题.

1. f （先中、中后皆行）
2. t
3. t （最好O(n)，最差$O(n^2)$）
4. f （快排在序列基本有序时接近$O(n^2)$）
5. t
6. f （比较次数最大为树高）
7. f （由散列函数及Hash冲突处理办法）
8. t （序列做为递归用栈，广度做为用队列）
9. t
10. t（子程序的有：①快排 ②希尔排序 ③堆排序 ④选择排序）

四. 简答题

1. 解：

由中序序列CDBAGFHE与后序序列DCBGHFEA可画出二叉树如下：



→ 解析：本题与2010-3（简答题第3题）类似，做法与它类似，这里就不展开讲解了.

（中序区分左右子树、前/后序中找根结点）

解：

设ve表示事件最早发生时间，vl表示事件最迟发生时间，e表示活动最早开始时间，l表示最迟开始时间

时间余量 d=l-e.

由图可得下表

	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8
ve(i)	0	3	5	9	21	18	28	30
vl(i)	0	7	5	9	21	18	28	30

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
e(i)	0	0	3	5	5	9	21	18	18	28
l(i)	0	4	7	9	5	9	21	18	24	28
d	0	4	4	4	0	0	0	0	6	0

由以上两个表可知：关键活动为 $a_1, a_5, a_6, a_7, a_8, a_{10}$

关键路径为 $V_1 \to V_3 \to V_4 \to V_6 \to V_5 \to V_7 \to V_8$

→ 关键活动性质满足时间余量为0

3.

解：

该算法的功能为删除顺序表中全部值为X的元素

→ while (i<L.length && L.elem[i]!=X) → 找到顺序表中第1个值为X的元素位置i

→
```
for(j=i+1;j<L.length && L.elem[i]!=X)
    if(L.elem[j]!=X){
        L.elem[i]=L.elem[j]
        i++
    }
```

这一段代码意思是遍历i位置以后元素j，若 elem[j]!=X，则将j放在i的位置上覆盖了X，这里就将第1个值为X的元素"删除"了，后面再有值为X的元素在 elem[j]!=X 条件下会被忽略，最后顺序表中X就不存在了，不等于X的元素不断插入.

→ L.length=i → i位置一直在更新，这里的i表示了顺序表的最后一个元素位置.

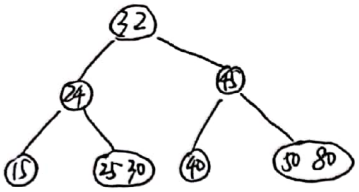
→ 举个例子：

初始序列：1,3,2,4,2,5 $\xrightarrow{X=2}$ 1,3,4,2,5 ⟶ 1,3,4,5

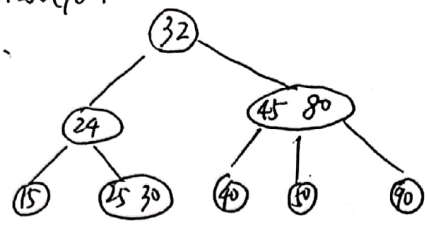
→ 相当于把 4,2,5 插入 1,3,2 中，4先覆盖2，再插入5即可，i一直指向最后一个元素.

4. 解：

① 插入30：



② 插入90：



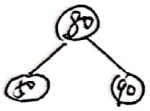
⇒考查3阶B-树的删除、插入操作.

⇒首先明确m阶B-树关键字个数为 $\lceil m/2\rceil-1 \sim m-1$，插入一个元素时先找到插入位置，然后直接插入

插入后检查被插入结点内关键字个数. 若关键字个数大于m-1，则需要进行拆分。进行拆分时，结点内关键字若已经有m个，此时取出第$\lceil m/2\rceil$个关键字并得第$1 \sim \lceil m/2\rceil-1$个关键字和第$\lceil m/2\rceil+1 \sim m$个关键字做成两个结点连接在第$\lceil m/2\rceil$个关键字左右的指针上，并得第$\lceil m/2\rceil$个关键字插入父结点相应位置中；如果在其父结点内出现关键字个数超出规定范围情况，则继续进行拆分操作.

⇒结合题目解题如下：

插入30后结点 (25 30) 满足3阶B-树定义，不用再处理，插入90后 (50 80 90) 内关键字个数 > m-1，进行拆分



，再把80插入父结点位置即可得答案，父结点 (45 80) 满足定义，不用再处理.

5. 解：

原始序列为：12→178→200→530→765→149→52→6

第1趟

分配（按最低位）结果为

0	1	2	3	4	5	6	7	8	9
530	530	52							
200	200	12			765	6		178	149

收集：200→530→12→52→765→6→178→149

第2趟

分配（按中间位）结果为

0	1	2	3	4	5	6	7	8	9
6 200	12		530	149	52	765	178		

收集：200→6→12→530→149→52→765→178

第3趟

分配（按最高位）结果为

0	1	2	3	4	5	6	7	8	9
52 12 6	178 149	200			530		765		

收集：6→12→52→149→178→200→530→765

五、算法设计：

1. ① sizeof(ElemType)

② !S.base

③ S.stacksize

④ e

⑤ top+1

→ 本题素材来自于乎版<<数据结构>> P47 的 Status Push( ), 当然做了些小改编。大家做这种题时不要一味地抄，看题干是否改编了，灵活运用。

→ ④⑤句合成一句 *S.top++=e;

2.

⑥ V<G.vexnum

⑦ DeQueue(Q,u)

⑧ !Visit[w]

⑨ TRUE

⑩ EnQueue(Q,w)

→ 本题素材依旧来源于网上乎版<<数据结构>> P170 的算法7.6. 这题就是书上的原版

→ 算法是广度优先的非递归算法。

六、综合算法（25分）

1.

```
void MergeList_L(LinkList &La, LinkList &Lb, LinkList &Lc){
  // 已知单链表La和Lb的元素为非递减排列
  // 归并得到的Lc也是非递减排列的单链线性表
  pa=La->next; pb=Lb->next;
  Lc=pc=La;
  while(pa && pb){
      if(pa->data <= pb->data){
          pc->next=pa; pc=pa; pa=pa->next;
      }
      else {pc->next=pb; pc=pb; pb=pb->next;
  pc->next=pa?pa:pb;  // 插入剩余段
  free(Lb);
}
```

→ 素材来自于罗版<<数据结构>>的例2.6的算法2.12，这个算法就是逐一找到La与Lb中对应值较小的插入Lc中，最后再把剩余的整个接入Lc的后面

→ 可能有些小伙伴不明白 pc=pa 与 pc=pb，这两句都是为了保留存Lc当前的尾指针，方便下一次插入.

2.

```
void ShortestPath_DIJ(MGraph G, int Vo, PathMatrix &p, ShortPathTable &D){
  for(v=0; v<G.vexnum; ++v){
    final[v]=FALSE; D[v]=G.arcs[Vo][v];
    for(w=0; w<G.vexnum; ++w) P[v][w]=FALSE;
    if(D[v]<INFINITY){P[v][Vo]=TRUE; P[v][v]=TRUE;}
  D[Vo]=0; final[Vo]=TRUE;
  for(i=1; i<G.vexnum; ++i){
    min=INFINITY;
    for(w=0; w<G.vexnum; ++w)
      if(!final[w])
        if(D[w]<min){v=w; min=D[w];}
    final[v]=TRUE;
    for(w=0; w<G.vexnum; ++w)
      if(!final[w] && (min+G.arcs[v][w]<D[w])){
        D[w]=min+G.arcs[v][w];
        P[w]=P[v]; P[w][w]=TRUE;
      }
  }
```

不同于2010-3（同类型题第3题）

笔者这里用于罗版<<数据结构>> P189 算法7.15. 单纯从代码量和言简意赅上，但是改动难度大，下面简单说明：

final[]用于存储顶点是否加入最短路中，D[v]是存V0到V的最短长度

P[]是存储加入最短路径的顶点w的前一个加入的顶点v，即 P[w]=P[v].

→