2023 年 833 真题解析

一、选择题

CABBA DBCCD BDDAB AB

1. C

解析：

在一个长度为 n 的顺序表中，删除一个元素时，有 n 个位置可供选择。当删除第 k 个元素时，需要改变从第 k+1 个元素起到第 n 个元素的存储位置，即进行“从第 k+1 到第 n 个元素往前移动一个位置”，共需移动 ni 个元素。

2. A

解析：

基本思想是：采用运算符栈是为了比较运算符的优先级,所有运算符必须进栈。只将大于栈顶元素优先级的运算符直接进栈,否则需要退栈栈顶运算符(先出栈的运算符先计算，同优先级的运算符在栈中的先计算)。表达式 a+b-a*((c+d)/e-f)+g 产生后缀表达式的过程如下表所列：

(以下操作数均直接输出，不作说明)

从左往右扫描表达	运算符栈	后缀表达	说明
a	空	a	
+	+	a	“+”入栈
b	+	ab	
-	-	ab+	“+”和“-”优先级相同，根据出现顺序，
a	-	ab+a	公众号/B 站/知乎：【研梦考研】
*	-*	ab+a	“*”优先级高于“-”，直接入栈
(	-*(	ab+a	“(”直接入栈即可
(	-*((	ab+a	“(”直接入栈即可
c	-*((	ab+ac	
+	-*((+	ab+ac	“+“优先级高于“(”，直接入栈
d	-*((+	ab+acd	
)	-*(	ab+acd+	运算符栈依次出栈，直至“ (”
/	-*(/	ab+acd+	“/”优先级高于“ (”，直接入栈
e	-*(/	ab+acd+e	
-	-*(-	ab+acd+e/	“-“优先级低于”/”，“/出栈输出，“-”
f	-*(-	ab+acd+e/f	
)	-*	ab+acd+e/f-	运算符栈依次出栈，直至“ (”
+	+	ab+acd+e/f-*-	“+”优先级低于”*”和””
g	+	ab+acd+e/f-*-g	
		ab+acd+e/f-*-g+	扫描结束，栈中元素依次出栈输出，直至栈

上表可知运算符栈中最大个数为 5

3. B

解析：

n 阶对称矩阵中的元素满足下述条件：aij=aji,(1<=i，j<=n)。对称矩阵中的每一对数据元素可以共用一个存储空间，因此可以将 n^2 个元素压缩存储到 n(n+1)/2 个元的空间中，即可以一维数组保存。

假设用一维数组 B[n(n+1)/2]作为对称矩阵 A 的存储结构，则 B[k]和矩阵元素 aij 的下标 i、j 的对应关系为：

当 i>j 时，k=i(i-1)/2+i;

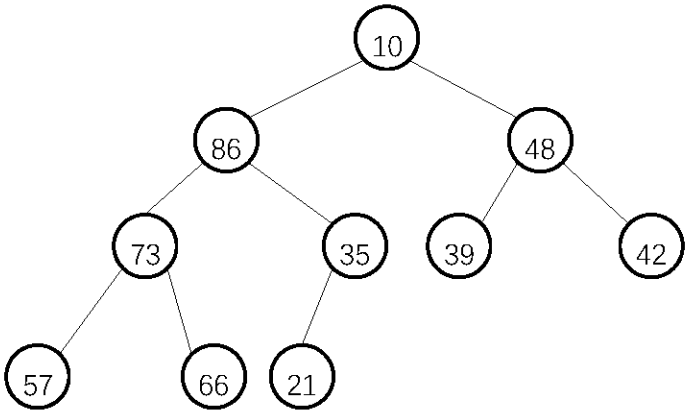
当 i<j 时，k=j(j-1)/2+i;

因为存储下三角元素，所以当 i<j 时，k=j(j-1)/2+i。

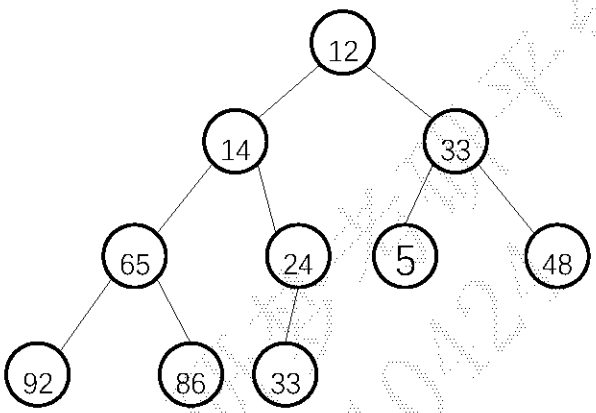
4．B

解析：

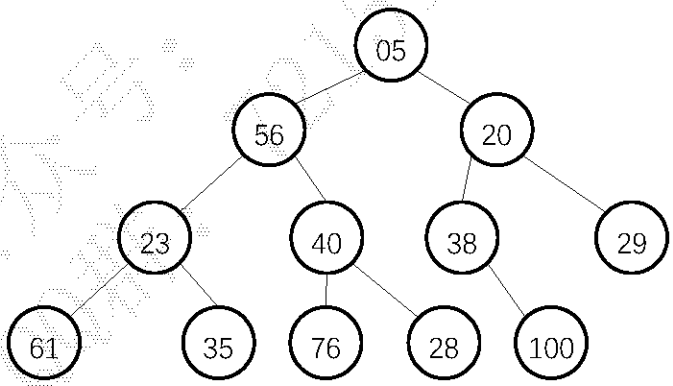
A．属于大顶堆



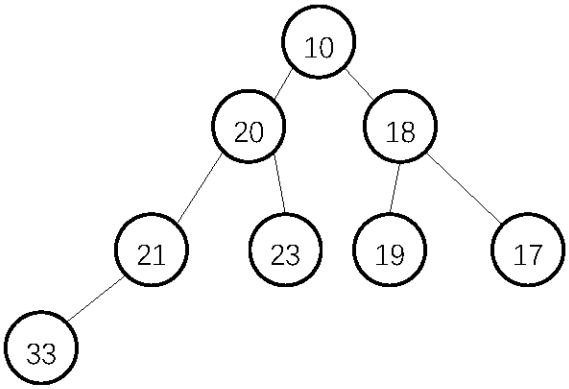
B.符合小顶堆定义，故选 B　　公众号/B 站/知乎：【研梦考研】



C．结点 56 以及结点 40 均不满足小顶堆定义

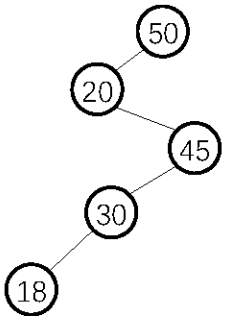


D．结点 18 不满足小顶堆定义　　公众号/B 站/知乎：【研梦考研】



5．A 解析：

折半查找对应判定树满足结点的值大于左子树的值，小于右子树的值：A．180 不该出现在 200 的右子树上



B、C、D 三个选项均满足。

6．D

解析：

A．v2 后的序列应为 **v5** 或 **v6**　　公众号/B 站/知乎：【研梦考研】

B．v5 后的序列应为 **v6** 或 **v7**

C．v2 后的序列应为 **v5** 或 **v6**

D．符合深度优先序列

7、解析：

D7	D6	D5	D4	H3	D3	D2	D1	H2	D0	H1	H0
1	0	1	0	0	1	0	1	0	0	1	0

P0=D6⊕D4⊕D3⊕D1⊕D0⊕H0=0⊕0⊕1⊕1⊕0⊕0=0

P1=D6⊕D5⊕D3⊕D2⊕D0⊕H1=0⊕1⊕1⊕0⊕0⊕1=1

P2=D7⊕D3⊕D2⊕D1⊕H2=1⊕1⊕0⊕1⊕0=1

P3=D7⊕D6⊕D5⊕D4⊕H3=1⊕0⊕1⊕0⊕0=0　　公众号/B 站/知乎：【研梦考研】

P3P2P1P0=0110，D2 出错

发送端信息数据为 D7D6D5D4D3D2D1D0=10101110，选 B

8、解析：

补码右移高位补符号位

[-128]补=1000 0000

右移 2 位后得 1110 0000

无符号数值=128+64+32=224

9、解析：

C0B00000 化为二进制 1 10000001 011 0000 0000 0000 0000 0000

尾数=-1.011B，指数=10000001-01111111B=2

真值=-101.1B=-5.5　　公众号/B 站/知乎：【研梦考研】

10、解析：

地址 512M=2^29，地址 29 位，DRAM 二维译码，地址线 15 位

11、解析：

鼠标键盘输入输出均采用中断方式

12、解析：

8086 总线 D0~D15，芯片总线 D0~D7，故采用的是位扩展

A0=0 时，偶地址有效，A0 连接 A 芯片有效

13、解析：

第九章内容，各节点之间不共享内存

14、参考第四章

15、解析：

变址后得到地址(X) +A

再间址得到有效地址为 ((X) +A)

16、解析：

改变时钟频率只会对每个时钟周期的时长有影响，与每个指令需要的时钟周期数 CPI、无关

17、解析： 公众号/B 站/知乎：【研梦考研】

化为二进制 1000 1111 1010 0000

补码扩展高位添加 16 位符号位，即 1111 1111 1111 1111 1000 1111 1010 0000

化为十六进制得 FFFF 8FA0

二、分析设计题

1、（1） 112，2^(-9)

（2）EX=-2，fX=0.5+0.125=0.625

X=0.625*1/4=0.15625

EY=5，fY=-（0.5+0.25）=-0.75

Y=-0.75*2^5=-24 公众号/B 站/知乎：【研梦考研】

(3)[EX+EY]补=[EX]补+[EY]补

```
   11 110
+  00 101
-------
   00 011
```

阶码 0011

尾数相乘，[fx]补=0.101，[-fx]补=1.011 ，[fY]补=1.010

符号	D	A	A-1	操作说明
00	000	1010	0	
00	000			+0
00	000			
00	000	0101	0	右移
11	011			+[-fx]补
11	011			
11	101	1010	1	右移
00	101			+[-fx]补
00	010			
00	001	0101	0	右移
11	011			+[-fx]补
11	100			
11	110	0010	1	右移

[fX*fY]补=1.100010

尾数左规 1 位得 1.000100，阶码减一得 0010

截尾法得尾数 1.1000，规格化表示为

[X*Y]浮=0010；1.000 公众号/B 站/知乎：【研梦考研】

2、（1）虚拟地址 4GB 得虚拟地址 32 位

主存地址 256MB 得主存地址 ~~26~~ 28 位

页面大小 4KB 得页内地址 12 位

Cache 块大小 64B 得块内地址 6 位

cache 直接相联共 4 块得块号 2 位

虚拟地址与主存地址映射时

虚拟地址划分

虚页号 20 位	页内地址 12 位

主存地址划分

实页号 ~~14~~ 16 位	页内地址 12 位

主存地址与 Cache 地址映射时

主存地址划分

区号 18 位	块号 2 位	块内地址 6 位

Cache 地址划分

块号 2 位	块内地址 6 位

虚页号位数=32-12=20　　公众号/B 站/知乎：【研梦考研】

实页号位数=~~26-12=14~~

物理地址访问 cache 时，物理地址划分为 3 个字段，如下所示

区号 18 位	块号 2 位	块内地址 6 位

（2）

虚页号 20 位	页内地址 12 位
00002H	908H

在主存中，对应物理地址为 236A908H

区号 18 位	块号 2 位	块内地址 6 位
10 0011 0110 1010 1001	00	00 1000

块号为 0，区号为 236A9H，未命中 Cache

（3）虚拟地址划分

标记 18 位	索引 2 位	页内地址 12 位
0000 00000000 0010 00	11	0110 1010 1000

命中 TLB，实页号为 286DH

对应物理地址 286D6A4H

3、（1）相容，互斥，相容，互斥　　公众号/B 站/知乎：【研梦考研】

（2）

节拍	微操作	有效控制信号
T1	AR<-R1	Rlout, ARin
T2	DR<-MM[AR]	ARout, Mread, DRSin
T3	S<-DR	DRlout, Sin
T4	T<-S+RO	ROout, ADD
T5	DR<-T	ToutDRlin
T6	MM[AR]<- DR	DRSout, ARout, Mwrite

（3）

节拍	微操作	有效控制信号
T1	DR<-RO	ROout, DRlin
T2	AR<-SP	SPout, ARin
T3	MM[AR]<-DR, SP<-SP+1	ARout, DRSout, Mwrite, SP+1

4、解答

（1）页大小为 8KB，页内偏移地址为 13 位，因此 A=B=32-13=19；D=13；C=24-13=11 主存块大小为 64B，因此 G=6。2 路组相联，每组数据区容量有 64B×2=128B，共有 64KB/128B=512 组，因此 F=9；E=24-G-F=24-6-9=9。因而 A=19，B=19，　C=11，D=13，E=9，F=9，G=6。TLB 中标记字段 B 的内容是虚页号，表示该 TLB 项对应哪个虚页的页表项

（2）块号 4099=000001000000000011B，因此所映射的 Cache 组号为 000000011B=3 对应的 H 字段内容为 000001000B。

（3）Cache 缺失带来的开销小，而处理缺页的开销大。因为缺页处理需要访问盘，而 Cache 缺失只要访问主存。

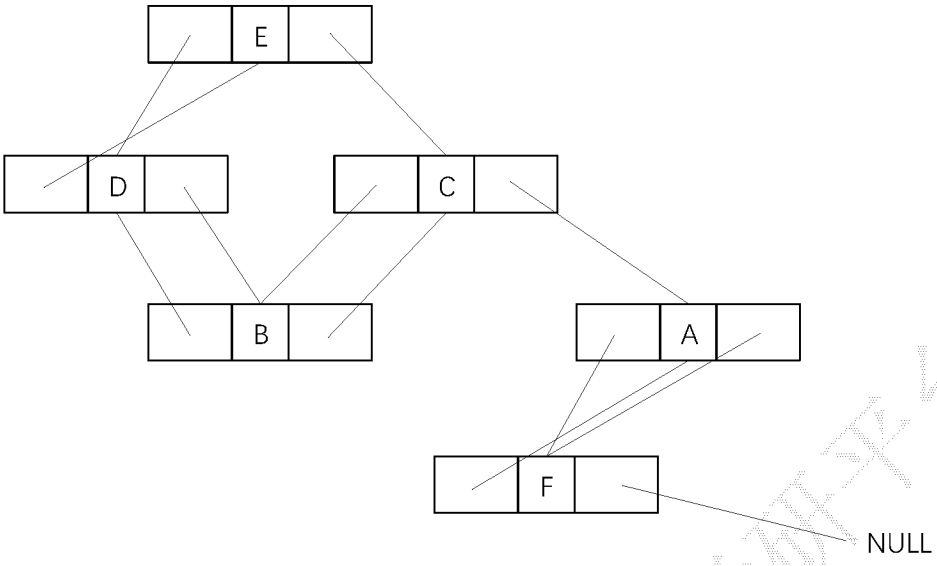
（4）因为采用直写策略时需要同时写快速存储器和慢速存储器，而写磁盘比写主存慢很多，所以在 Cache-主存层次，Cache 可以采用直写策略，而在主存-外存(磁盘)层次，修改页面内容时总是采用回写策略。

三、综合设计题

1.

（1）

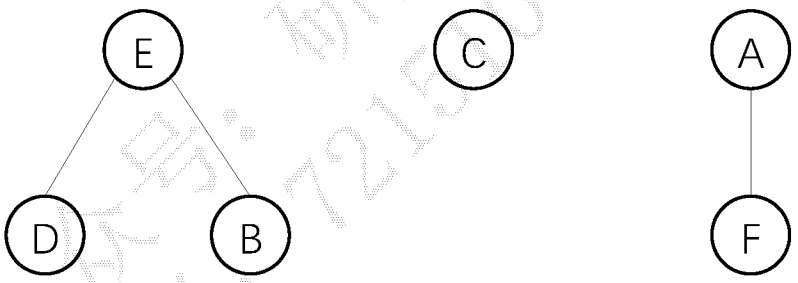
该二叉树的先序序列：EDBCAF，对应的先序线索二叉树如下：　　公众号/B 站/知乎：【研梦考研】



（2）二叉树顺序存储表示，其中-1 表示空结点

1	2	3	4	5	6	7	8	9	10	11	12	13	14
E	D	C	-1	B	-1	B	-1	A	-1	-1	-1	-1	F

（3）



2.（1）

初始：　　31，70，42，12，28，20，76，40

第一趟：31，70，12，42，20，28，40，76

第二趟：12，31，42，70，20，28，40，76

第三趟：12，20，28，31，40，42，70，76

（2）

①　20，28，12，31，42，70，，76，40

②　满足，因为每次划分都可以使左右区间的长度之差小于等于 1，使得快速排序的效率最高

3.

H（31）= 31%11 = 9　　H（70）= 70%11 = 4　　公众号/B 站/知乎：【研梦考研】

H（43）= 43%11 =10　　H（12）= 12%11 =1

H（28）= 28%11= 6　　H（20）= 20%11 = 9

H（76）= 76%11 =10　　H（40）= 40%11 = 7

H（64）= 64%11 = 9

(1)

对于关键字 20，76，64 会发生冲突，，冲突后的哈希表如下所示

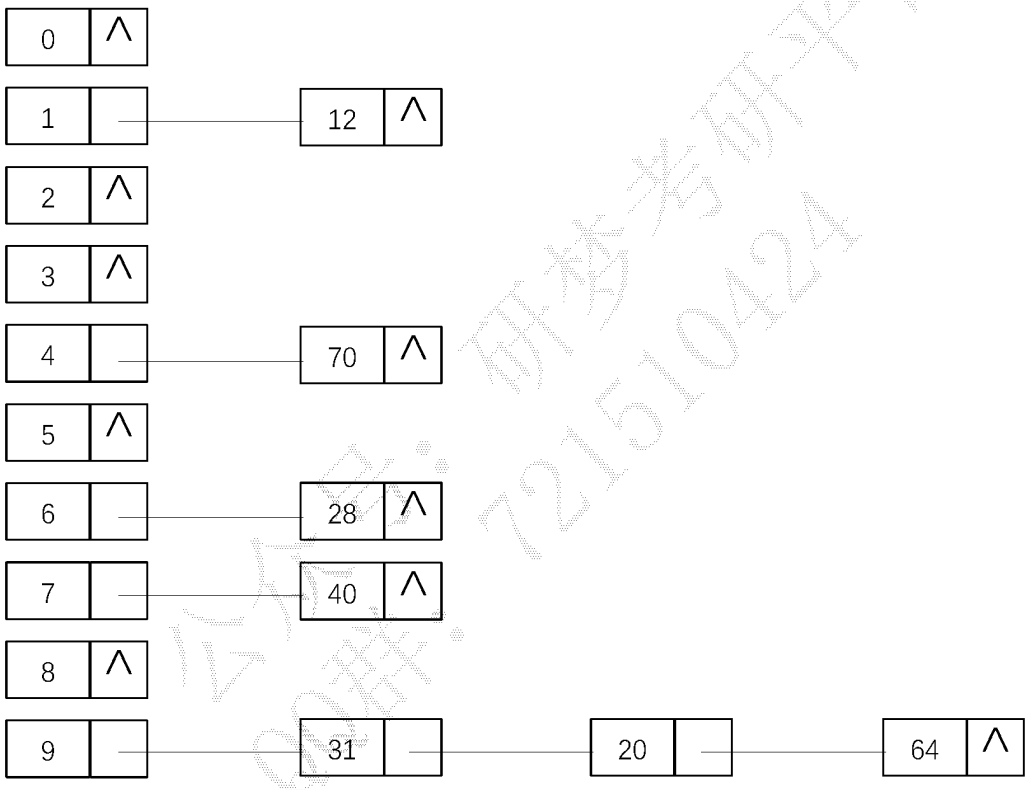
0	1	2	3	4	5	6	7	8	9	10	11	12
64	12			70		28	40		31	43	20	76

公众号/B 站/知乎：【研梦考研】

查找成功的平均查找长度 ASL 成功=∑pi*ci=（1+1+1+1+1+3+3+1+5）/9 =17/9

查找失败的平均查找长度 ASL 失败=∑pi*ci=（3+2+1+1+2+1+3+2+1+7+6）/11 = 29/11

(2)



查找成功的平均查找长度 ASL 成功 =∑pi*ci= (1+1+1+1+1+1+2+2+3) /9=13/9

查找失败的平均查找长度 ASL 失败 =∑pi*ci=(1+1+1+1+2+3) /11=9/11

4.

(1) p->next!=NULL

(2) p = start;

(3)(--len)>=k

(4) p->next　　公众号/B 站/知乎：【研梦考研】

5.

算法思路：

主要考查图的广度优先遍历。通过从顶点 u 开始对图进行广度优先遍历，如果访问到顶点 v，则说明从顶点 u 到顶点 v 存在一条路径。因为在图的遍历过程中，要求每个顶点只能访问一次，所以该路径一定是简单路径。在遍历过程中，将当前访问到的顶点都记录下来，就得到了从顶点 u 到顶点 v 的简单路径。可以利用一个一维数组 parent 记录访问过的顶点，如 parent[uj=w，表示顶点 w 是 u 的前驱顶点。如果 u 到 v 是一条简单路径，则输出该路径。

存储结构：

```
#define MAX_VERTEX_NUM 20
typedef struct ArcNode{
    int adjvex;
    struct ArcNode *nextarc;
}ArcNode;              公众号/B 站/知乎：【研梦考研】
typedef struct VNode{
    VertexType data;
    ArcNode *firstarc;
}VNode,*AdiList[MAX_VERTEX_NUM];
typedef struct{
    AdjList vertices;
    int vexnum,arcnum;          公众号/B 站/知乎：【研梦考研】
}ALGraph;
```

算法代码：

```
void FindPath(ALGraph G, int u, int v) //求图 G 中从顶点 u 到顶点 v 的一条简单路径
{
    int k, i;
    SeqStack S, T;
    ArcNode*p;
    int visited[MAX_VERTEX_NUM];            //判断当前顶点是否访问过
    int parent[MAX_VERTEX_NUM];          //存储已经访问顶点的前驱顶点
    InitStack(&S);
    InitStack(&T);
    for (k = 0; k < G.vexnum; k++)          //访问标志初始化
        visited[k] = 0;
    PushStack(&S, u);                       //开始顶点入栈
    visited[u] = 1;
    while(!StackEmpty(S))                   //BFS 过程
     {
        PopStack(&S,&k);
        p = G.vertex[k].firstarc;          公众号/B 站/知乎：【研梦考研】
        while (p != NULL)
        {
              if (p->adivex == v)        //如果找到顶点 v,从顶点 v 将路径上顶点逆序入栈，再顺序出栈并输出，即可得到从 u 到 v 的路径序列。
              {                                 公众号/B 站/知乎：【研梦考研】
                  parent[p->adjvex]= k;      //顶点 v 的前驱顶点序号是 k
                  printf("顶点%s 到顶点%s 的路径是：",G.vertex[u].data,G.vertex[v].data);
                  i = V;
                  do                            //从顶点 v 开始将路径中的顶点依次入栈
                  {
                      PushStack(&T, i);
                      i = parent[i];          //parent[i]是 i 的前驱顶点
                  }while (i != u);
                  PushStack(&T, u);
                  while (!StackEmpty(T))     //从顶点 u 开始输出 u 到 v 中路径的顶点
                  {
                      PopStack(&T &i);
```

```
                printf("%s ", G.vertex[i].data);
            }
        }                          公众号/B站/知乎：【研梦考研】
        else if(visited[p->adjvex]== 0)    //若未找到顶点v且邻接点未访问过，则继续搜索
        {
            visited[p->adjvex] = 1;
            parent[p->adjvex]= k;
            PushStack(&S,p->adjvex);
        }
        p = p->nextarc;                公众号/B站/知乎：【研梦考研】
      }
    }
    //如果搜索完成 visited[v]=0，说明没有遍历到顶点v，即u到v之间不存在路径
    if(visited[v] == 0)
        printf(“nopath”);
}
```