目　录

1

真题讲解

1.1 杭州电子科技大学 2019 年计算机考研真题答案

1.1.1 一、单向选择题 (本大题共 7 小题，每小题 2 分，本大题共 14 分)

1.

B

略

2.

A

栈中数据 1) + 2) - 3) -*((4) -*((+ 5) -*(/

输出数据 1) ab 2) + 3) a 4) cd 5) +e 后面就开始减少了所以最多个数是 5

3.

B

公式结点和分支的关系结点数 - 1 == 分支数

20 + 10 + 1 + 10 + x -1 = 20*4 + 10 *3 + 1 * 2 + 10 * 1

x = 82

4.

C

略

5.

A

front 和 rear 之间空一个元素作为判断满还是空

6.

B

略

7.

A

画出小根堆，然后把 3 放入最后面，然后重建小根堆即可

1.1.2 二、填空题 (本大题共 7 空，每空 2 分，本大题共 14 分)

1. 已知指针 P 指向单链表 L 中的某节点，则删除其后继节点的语句是:if(p->next){q = p->next;p->next=p->next->next;free(q)}.

2. 设广义表 (((a,b), x),((a),(b)),(c,(d,(y)))), 得到 y 的对广义表 A 的操作序列是:Head(Head(Tail(Head(Tail(

3. 已知一棵树有 2011 个节点的树，其叶节点个数为 116，该树对应的二叉树中无右孩子的节点个数是:1896.

4. 有向图 G=(V,E), 其中 V(G)=0,1,2,3,4,5, 用 <a,b,d> 三元组表示弧 <a,b> 及弧上的权 d. E(G)=<0,5,100>, <0,2,10>, <1,2,5>, <0,4,30>, <4,5,60>, <3,5,10>, <2,3,50>, <4,3,20>, 则从源点 0 到顶点 3 的最短路径长度是:50, 经过的中间顶点是:4.

5. 127 阶 B-树种每个节点最多有:126 个关键字; 除根节点外外所有非终端节点至少有 64 棵子树。

1.1.3 三、简答题 (本大题共 4 小题，每小题 10 分，本大题共 40 分)

1.

(1) 请画出所构造的散列表。

解:

$\because \alpha = 0.710$

计算 Hey(key)

7	8	30	11	18	9	14
0	3	6	5	5	6	0

构造的散列表

地址	0	1	2	3	4	5	6	7	8	9
关键字	7	14		8		11	30	18	9	

(2) 分别计算等概率情况下查找成功和查找失败的平均查找长度。

$ASL_{查找成功} = (1+2+1+1+1+3+3)/7 = \frac{12}{7}$

$ASL_{查找失败} = (3+2+1+2+1+5+4)/7 = \frac{18}{7}$

2.

(1) A 中非零元素的行下标与列下标的关系;

解:

i = j 或 i + j = n + 1

(2) 给出 A 中非零元素 $a_{i,j}$ 的下标 (i,j) 与 B 中的下标 R 的关系;

$$R = \begin{cases} 2\times(i-1), i = j \\ 2\times(i-1), i+j = n+1, i < j \\ 2\times(i-1), i+j = n+1, i > j \end{cases} \tag{1.1}$$

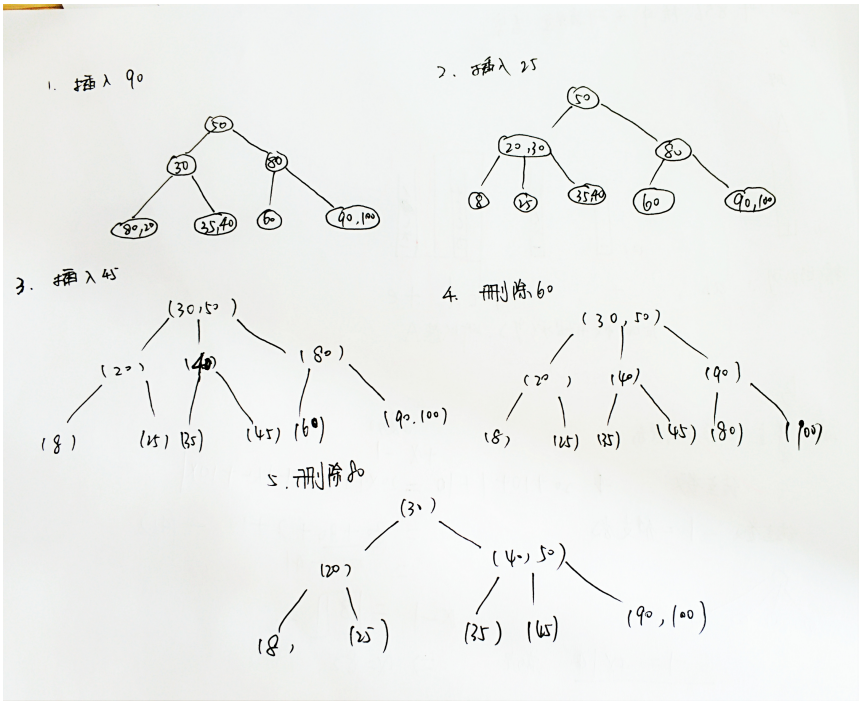
(3) 假定矩阵中每个元素占一个存储单元且 B 的起始地址为 A_0, 给出利用 $a_{i,j}$ 的下标定位在 B 中的位置公式.

$$Loc(a_{i,j}) = \begin{cases} 2\times(i-1)\times size, i = j \\ 2\times(i-1)\times size, i+j = n+1, i < j \\ 2\times(i-1)\times size, i+j = n+1, i > j \end{cases} \tag{1.2}$$

size 为一个存储单元的大小

3. (1) 插入 90 (2) 插入 25 (3) 插入 45 (4) 删除 60 (5) 删除 80

解：



4. 下标给出了某工程各工序之间的优先关系和各工序所需时间.

工序代号	A	B	C	D	E	F	G	H	I	J	K	L	M	N
所需时间	15	10	50	8	15	40	300	15	120	60	15	30	20	40
先驱工作	-	-	A,B	B	C,D	B	E	G,I	E	I	F,I	H,J,K	L	G

(1) 画出相应的 AOE 网;

(2) 列出各时间的最早发生时间，最迟发生时间;

(3) 找出关键路径并指明完成该工程所需最短时间.

解：

错题 =.=

1.1.4 四、程序阅读题 (本大题共 3 小题，每小题 5 分，本大题共 15 分)

1. 算法一:

判断是否是连通图

2. 算法二:

合并两个有序序列, 从小到大

3. 算法三：

最大的排在最后，第二大的排在最前，以此类推，直到 low == high

1.1.5 五、算法设计题 (本大题共 2 小题，每小题 11 分，本大题共 22 分)

1.

算法思想:

首先:

(1) 如果 p1 的指数比 p2 的指数大的话，新生成一个结点赋以 p1 的值。p1 向后移动一个结点

(2) 如果 p1 的指数比 p2 的指数相等的话，判断系数项相加是否等于 0，如果不等于那么将 p1 的系数和 p2 的系数和赋给新的结点,p1 和 p2 都向后移动

(3) 如果 p2 的指数比 p1 的指数大的话，新生成一个结点赋以 p2 的值。p2 向后移动一个结点

直到 p1 或 p2 走到了链的结尾。

然后将剩下的链表赋给新的链表

```
#include <iostream>
using namespace std;
struct PolyNode {
    int coef;
    int expon;
    struct PolyNode *link;
};
typedef struct PolyNode* Polynomial;
Polynomial add(Polynomial a, Polynomial b) {
    Polynomial q = NULL;
    Polynomial head = q;
    while (a && b) {
        if (a->expon > b->expon) {
            Polynomial p = (Polynomial)malloc(sizeof(
```

```
        struct PolyNode));
             p->link = NULL;
             p->coef = a->coef;
             p->expon = a->expon;
             if (q == NULL) {
                 q = p; head = q;
             }
             else {
                 q->link = p;
                 q = p;
             }
             a = a->link;
         }
         else if (a->expon == b->expon) {
             //int expon = a->expon;
             int coef = a->coef + b->coef;
             if (coef != 0) {
                 Polynomial p = (Polynomial)malloc(sizeof(
        struct PolyNode));
                 p->link = NULL;
                 p->coef = coef;
                 p->expon = a->expon;
                 if (q == NULL) {
                     q = p; head = q;
                 }
                 else {
                     q->link = p;
                     q = p;
                 }
             }
             a = a->link;
             b = b->link;
         }
         else {
             Polynomial p = (Polynomial)malloc(sizeof(
        struct PolyNode));
             p->link = NULL;
             p->coef = b->coef;
```

```
            p->expon = b->expon;
            if (q == NULL) {
                q = p; head = q;
            }
            else {
                q->link = p;
                q = p;
            }
            b = b->link;
        }
    }
    while (a) {
        Polynomial p = (Polynomial)malloc(sizeof(struct
       PolyNode));
        p->link = NULL;
        p->coef = a->coef;
        p->expon = a->expon;
        q->link = p;
        q = p;
        a = a->link;
    }
    while (b) {
        Polynomial p = (Polynomial)malloc(sizeof(struct
       PolyNode));
        p->link = NULL;
        p->coef = a->coef;
        p->expon = a->expon;
        q->link = p;
        q = p;
        b = b->link;
    }
    return head;
}
int main() {
    Polynomial p1 = (Polynomial)malloc(sizeof(struct
       PolyNode));
    Polynomial p1_1 = (Polynomial)malloc(sizeof(struct
       PolyNode));
```

```
    Polynomial p1_2 = (Polynomial)malloc(sizeof(struct
       PolyNode));
    Polynomial p2 = (Polynomial)malloc(sizeof(struct
       PolyNode));
    Polynomial p2_1 = (Polynomial)malloc(sizeof(struct
       PolyNode));

    p1->coef = 3; p1->expon = 3; p1->link = p1_1;
    p1_1->coef = 5; p1_1->expon = 1; p1_1->link = p1_2;
    p1_2->coef = 6; p1_2->expon = 0; p1_2->link = NULL;

    p2->coef = 3; p2->expon = 2; p2->link = p2_1;
    p2_1->coef = 4; p2_1->expon = 1; p2_1->link = NULL;

    Polynomial out = add(p1, p2);
    while (out!=NULL) {
        cout << out->coef << ” ” << out->expon << endl;
        out = out->link;
    }

    system(”pause”);
}
```

2.

解:

算法思想：

判断两个结点是否相同

如果 T1 的左孩子 == T2 的左孩子 || T1 的左孩子 == T2 的右孩子，递归判断剩下的结点是否满足同构

如果 T1 的右孩子 == T2 的左孩子 || T1 的右孩子 == T2 的右孩子，递归判断剩下的结点是否满足同构

```
#include <iostream>
#include <set>
using namespace std;
```

```
#define MaxTree 10
#define ElementType char
#define Tree int
#define Null -1
struct TreeNode {
    ElementType Element;
    Tree Left;
    Tree Right;
}T1[MaxTree], T2[MaxTree];

bool check(int t1, int t2) {//t1 t2 乃 下标
    if (t1 == -1 && t2 == -1) {
        return true;
    }
    if (t1 == -1 || t2 == -1) {
        return false;
    }
    if (T1[t1].Element != T2[t2].Element) {
        return false;
    }

    bool checked = true;
    if (T1[t1].Left == T2[t2].Left) {
        checked = check(T1[t1].Left, T2[t2].Left);
    }
    else if (T1[t1].Left == T2[t2].Right) {
        checked = check(T1[t1].Left, T2[t2].Right);
    }
    else {
        return false;
    }

    if (checked == false) {
        return false;
    }

    if (T1[t1].Right == T2[t2].Left) {
```

```
        checked = check(T1[t1].Right, T2[t2].Left);
    }
    else if (T1[t1].Right == T2[t2].Right) {
        checked = check(T1[t1].Right, T2[t2].Right);
    }
    else {
        return false;
    }

    if (checked == false){
        return false;
    }
    return true;
}
int main() {
    for (int i = 0; i < 5; i++) {
        T1[i].Element = 'A' + i;
        T2[i].Element = 'A' + i;
    }
    T1[0].Left = 1;  T1[0].Right = 2;
    T1[1].Left = 3;  T1[1].Right = 4;
    T1[2].Left = -1; T1[2].Right = -1;
    T1[3].Left = -1; T1[3].Right = -1;
    T1[4].Left = -1; T1[4].Right = -1;

    T2[0].Left = 2; T2[0].Right = 1;
    T2[1].Left = 4; T2[1].Right = 3;
    T2[2].Left = -1; T2[2].Right = -1;
    T2[3].Left = -1; T2[3].Right = -1;
    T2[4].Left = -1; T2[4].Right = -1;
    // test
    //T1[1].Element = 'P';
    cout << check(0, 0) << endl;
    system("pause");
}
```

1.1.6 六、计算题 (本大题共 3 小题，本大题共 10 分)

1.

$[x]_{原码} = 0.1101 \times 2^{-2}$

$[x]_{阶移} = 0,11110$

$[x]_{规} = 0,11110;0,110100000$

2.

(教材 81 页) 尾数决定了有效数值的精度

本题，小数点后有 9 位，则精度为 2^{-9}

小知识点 (P82)：对于原码来说，规格化尾数的格式为 x.1xxxx，即小数点后第一位（数值最高位）应为 1；对于补码来说，规格化尾数的格式为 0.1xxxx、1.0xxxxx，即符号位和数值最高位应相反

3.

(浮点数加减，P138)

大端模式：低地址内存存储的是数据的高字节

x=FD50H,y=FAA0H

$[x]_{浮} = 1111110101010000$

$[Ex]_{移} = 1,11111$

$[Mx]_{补} = 0.101010000$

$[y]_{浮} = 1111101010100000$

$[Ey]_{移} = 1,11110$

$[My]_{补} = 1.010100000$

对阶 Ex-Ey=1, 使 $[Ey]_{移} = [Ex]_{移} = 1,11111$，

同时 y 的尾数右移一位，$[My]_{补} = 1.101010000(0)$

尾数相加减 $[Mx - My]_{补} = [Mx]_{补} + [-My]_{补}$

$[-My]_{补} = 0,010110000(0)$

双符号位

[Mx] 补 00,101010000

+[My] 补 00,010110000(0)

= 01,000000000(0)

双符号为为 01，有溢出，需要右规一位，阶码加一，而 $[E]_{移} = 1,11111$，也溢出，所以浮点数 x+y 溢出

1.1.7 七、设计题 (本大题共 2 小题，本大题共 10 分)

1.

解：

2 片 8k*8b 的 ROM 从 2000H 开始，1 片 8k*16b 的 SRAM 从 8000H 开始

地址需要 13 位

A15_____A0

0010 0000 0000 0000

0011 1111 1111 1111 第一片 ROM，高三位 001，对应片选信号为 y1

0100 0000 0000 0000

0101 1111 1111 1111 第二片 ROM，高三位 010，对应片选信号为 y2

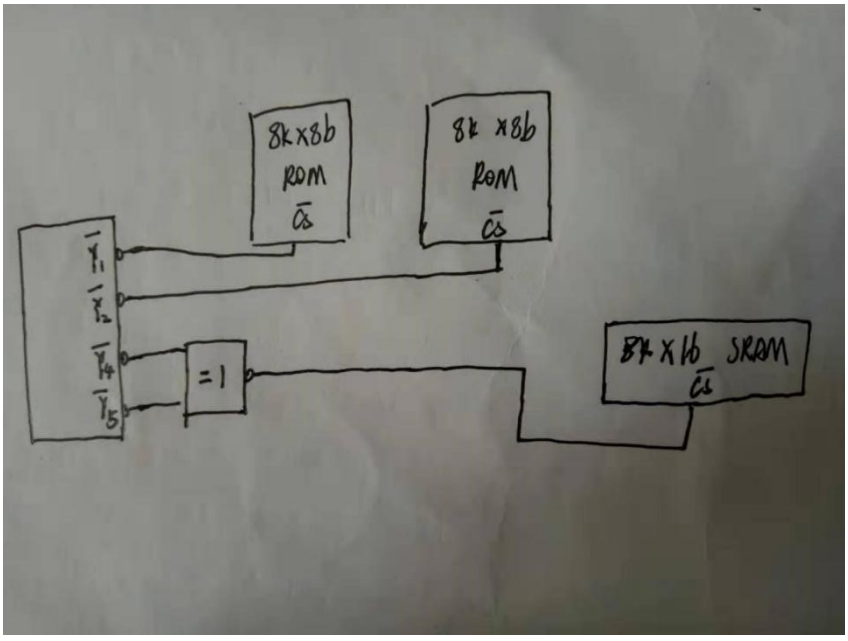
1000 0000 0000 0000

1001 1111 1111 1111

1010 0000 0000 0000

1011 1111 1111 1111 SRAM 对应高三位有两个，100，101，所以对应片选信号有 y4 和 y5, 又低电频有效，所以在 y4、y5 后添加一个与门，只要有一个有效（即为 0），就选中 SRAM

本题错误就在于片选信号



2.

应使用多体交叉存储器的方式来最大限度提高带宽
1M*32b/256kB=4*4
共需要 16 片芯片，分为 4 组，每组 4 片，组内进行位扩展
然后构造 4 个存储体，同一体内地址不连续
（交叉存储器的知识教材 P190）
1M*32b=$2^2$0*32b, 主存中有 $2^2$0 个单元，主存地址为 20 位
低 2 位用于选择存储体，高 18 位是存储单元在存储体内的相对地址

1.1.8 八、(本大题共 10 分)

1.
JNZ LOOP 指令起始地址 0CH
在执行 JNZ LOOP 指令时，PC 中内容为 0EH
偏移 x 14+x=6,x=-8，用补码表示
偏移量 11111000
2.
前一个是寄存器寻址，后一个是变址寻址
3.
8/2=4，该 cache 中一共有四块，每块正好存放一条指令
前三条指令未命中，3*2=6 次
到循环，第一次循环四条指令均未命中，4*2=8 次
一共循环 10 次，剩下 9 次均命中，4*2*9=72 次
最后一条停机指令，未命中 2*1=2 次
命中率：4*2*9/(3*2+4*2+4*2*9+2*1)*100%=81.82%
4.
解:
由上可知，该程序一共未命中 16 次，命中 72 次
t=72*10ns+16*(100ns+10ns)=2480ns

1.1.9 九、(本大题共 15 分)

1.

解：

指令	rd_rt_s	imm_s	rt_imm_s	Alu_mem_s	Write_reg	Mem_Write
ANDi	1	1	1	1	1	0

2.

解：

在这一指令周期内指令不会发生变化

3.

解：

不能，该指令需要对 pc 进行修改即 pc=pc+offset，该图中没有结构可以实现该运算

4.

解:

79FBH 最高位为 0，进行 0 扩展，得到 000079FBH

9F69H 最高位为 1，进行 1 扩展，得到 FFFF9F69H

5.

解：

图 2 的体系结构是哈佛结构，因为数据和指令分开存放

每执行一条指令，pc 值 +4

6.

解：

属于 SISD(教材 p12 Flynn 分类)