

一. 选择题

1. B

→ 对任意一棵二叉树T, 有以下结论 ① $n_0 = n_2 + 1$ ② 总分支数 = 总结点数 - 1

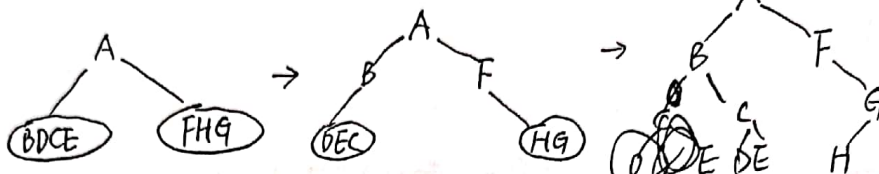
→ $n_2 = n_0 - 1 = 10$, 总分支数 = $n_0 + n_1 + n_2 - 1 = 11 + 2 + 10 - 1 = 22$

2. D

→ 考查了满二叉树的性质 → 深度为k的^满二叉树有 $2^k - 1$ 个结点.

3. A

→ 中序: BDCEAFHG
后序: DECBHGFA



(后序中确定root, 由root在中序中确定左右分支, 用递归看待每个结点)

→ 先序: ABCDEFGH

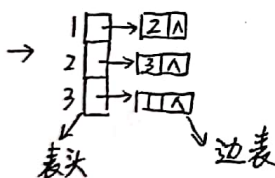
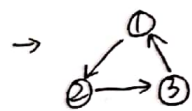
→ 当然, 这题没必要这样做, 只要确认了root=A即可选A项

4. B

→ 这类题区分两种问法

- ① 表头结点的^{链表中}链表元素个数.
- ② 边表结点中顶点 v_i 的出现次数.

, 还要区分是邻接表还是逆邻接表



问法①

问法②

邻接表

表示 v_i 的出度

表示 v_j 的入度

逆邻接表

表示 v_i 的入度

表示 v_j 的出度.

→ 回到本题, 这里的表头结点所指边表结点, 故选B

5. B

→ 见书版P47页.

6. D

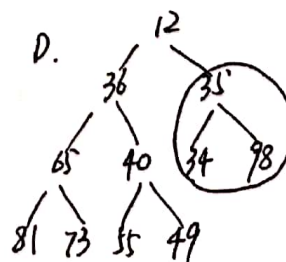
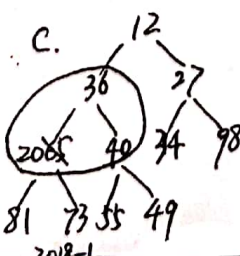
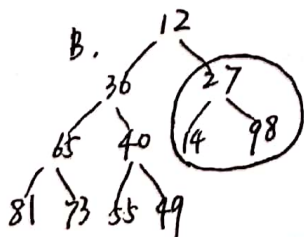
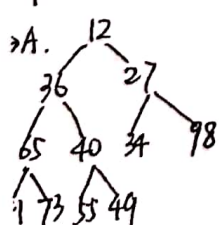
→ 顺序表方便元素存取.

7. C

8. B

→ 第一选择循环链表避免假溢出, 第二选择带尾指针的, 方便插入删除, 带头指针的更方便出入队操作.

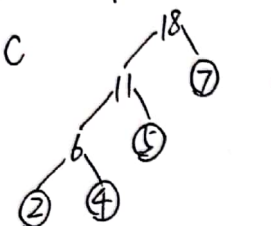
9. A



10. A
 → 从 $a_{1,1}$ 开始, $n \times n$ 矩阵, 公式 $k = \begin{cases} \frac{i(i-1)}{2} + j - 1, & i \geq j \\ \frac{j(j-1)}{2} + i - 1, & i < j \end{cases}$ 代入 $i=8, j=5 \Rightarrow k=32$

11. D

→ 当链表仅有一个元素时, 头指针均要修改

12. C
 →  $\rightarrow wpl = (2+4) \times 3 + 5 \times 2 + 7 \times 1 = 35$

13. C

→ 由常识知, 最佳带图均由树构成, 而且分支数最多为 2 个。

14. C

15. D (同 2011-13)

二. 堆栈

1. 统一每个结点的插入删除操作
2. 队列
3. 顺序存储且关键字有序
4. 该结点右子树中最左边的那个结点
5. 最小 越近
6. m (m 阶 B-树中结点最少 $m-1$ 个关键字, m 个分支)
7. $O(n)$
8. $O(n^2)$ 或 $O(n^2)$ (超邻接表或邻接矩阵存储)
9. 路径长度递增



三、判断题

1. t
2. f
3. t
4. f ($2^5 = 32$)
5. t (B+还可以顺序查找)
6. t
7. t
8. t (见书版 P166-P167)
9. f (所需存储量为了节省空间)
10. t.

四、简答题

1. 解: 所有拓扑序列为

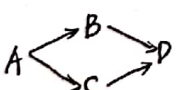
① 1, 2, 3, 4, 5, 6 ② 1, 2, 4, 3, 5, 6

③ 2, 1, 3, 4, 5, 6 ④ 2, 1, 4, 3, 5, 6

还能通过深度优先遍历获得拓扑序列

→ 拓扑排序简单, 说明一下如何由DFS算法得到拓扑序列

→ 深度优先遍历过程中顶点退栈的顺序是逆拓扑序列, 有点绕, 下面举个例子

→  , 从A开始DFS, A入栈S, $S = \{A\}$, B入栈S, $S = \{A, B\}$, D入栈, $S = \{A, B, D\}$

D出栈 $S = \{A, B\}$, B出栈 $S = \{A\}$, C入栈 $S = \{A, C\}$, C出栈, $S = \{A\}$, A出栈 $S = \{\}$

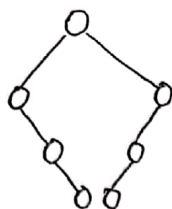
元素出栈序列为 DCBA, 上例的一个拓扑序列为 ACBD, 刚好是出栈序列的逆序列

→ 另一个拓扑序列 ABCD 可以由另一种出栈序列 DCBA 的逆序列得到



2.

解: 满足左子树的前序序列与中序序列相等, 和右子树的中序序列与后序序列相等的二叉树结构如下:



→ 先序与中序相同情况

- ① 空树或仅有 root 结点
- ② 右单支树

→ 中序与后序相同情况

→ 类似问题还有 先序与后序相同, 先序与后序逆序相同等等, 注意发散思维

3. 解:

初始序列: 12 18 4 3 6 13 2 9 19 8

$d=5$, 第一趟: 12 2 4 3 6 13 18 9 19 8

$d=3$, 第2趟: 3 2 4 8 6 13 12 9 19 18

$d=2$, 第3趟: 3 2 4 8 6 9 12 13 19 18

$d=1$, 第4趟: 2 3 4 6 8 9 12 13 18 19

→ 希尔排序亦称缩小增量排序, 比较简单, 如 $d=5$, 12 开始的^的序列 12 13, 2 开始的^的序列为 2 18

又如 $d=2$ 时 3 开始的序列为 3 4 6 12 19, 2 开始的序列为 2 8 13 9 18 要调整为 2 8 9 13 18

(这里就是13与9互调即可)



解: 由散列函数 $H(\text{key}) = (3 * \text{key}) \% 11$ 可得各关键字散列的地址.

key	33	41	20	24	30	13	01	67
Hash(key)	0	2	5	6	2	6	3	3

由线性探测再散列解决冲突 $H_i = (H(\text{key}) + d_i) \% 11$ 得

地址	0	1	2	3	4	5	6	7	8	9	10
关键字	33		41	30	01	20	24	13	67		

各关键字比较次数

key	33	41	20	24	30	13	01	67
次数	1	1	1	1	2	2	2	6

综上所述:

$$ASL_{成功} = \frac{1+1+1+1+2+2+2+6}{8} = 2$$

→ 这类题属常规高频题, 重要掌握, 类似的有2011-6, 其他解决冲突的题有2013-6, 2016-2

5. 解: 采用 Floyd 算法解决 具体过程如下:

设置两个矩阵 A 与 $path$, A 用于记录当前已经求得的所有两个顶点最短路径的长度, $path$ 用来记录

当前两顶点间最短路径要经过的中间结点, 这里规定 -1 为无中间点的情况. a, b, c, d, e 在 $path$ 矩阵中用 1, 2, 3, 4, 5 表示

① 初始情况下

$$A_0 = \begin{matrix} & \begin{matrix} a & b & c & d & e \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix} & \begin{bmatrix} 0 & 1 & \infty & \infty & \infty \\ 6 & 0 & 2 & \infty & \infty \\ \infty & \infty & 0 & 2 & 4 \\ \infty & 1 & 3 & 0 & \infty \\ \infty & \infty & \infty & 5 & 0 \end{bmatrix} \end{matrix} \quad path_0 = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

② 以 a 为中间点, 检测所有点对, 若经过 a 该点对路径更短则更新该值, 置 $path$ 矩阵相应位置为中间点 1.

$$A_1 = \begin{matrix} & \begin{matrix} a & b & c & d & e \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix} & \begin{bmatrix} 0 & 1 & \infty & \infty & \infty \\ 6 & 0 & 2 & \infty & \infty \\ \infty & \infty & 0 & 2 & 4 \\ \infty & 1 & 3 & 0 & \infty \\ \infty & \infty & \infty & 5 & 0 \end{bmatrix} \end{matrix} \quad path_1 = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$



以b为中间点, 检测所有点对

$$A_2 = \begin{matrix} & a & b & c & d & e \\ \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix} & \begin{bmatrix} 0 & 1 & 3 & \infty & \infty \\ 6 & 0 & 2 & \infty & \infty \\ \infty & \infty & 0 & 2 & 4 \\ 7 & 1 & 3 & 0 & \infty \\ \infty & \infty & \infty & 5 & 0 \end{bmatrix} \end{matrix}$$

$$path_2 = \begin{matrix} & a & b & c & d & e \\ \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix} & \begin{bmatrix} - & - & 2 & - & - \\ - & - & - & - & - \\ - & - & - & - & - \\ 2 & - & - & - & - \\ - & - & - & - & - \end{bmatrix} \end{matrix}$$

以c为中间点, 检测所有点对

$$A_3 = \begin{matrix} & a & b & c & d & e \\ \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix} & \begin{bmatrix} 0 & 1 & 3 & 5 & 7 \\ 6 & 0 & 2 & 4 & 6 \\ \infty & \infty & 0 & 2 & 4 \\ 7 & 1 & 3 & 0 & 7 \\ \infty & \infty & \infty & 5 & 0 \end{bmatrix} \end{matrix}$$

$$path_3 = \begin{bmatrix} - & - & 2 & 3 & 3 \\ - & - & - & 3 & 3 \\ - & - & - & - & - \\ 2 & - & - & - & 3 \\ - & - & - & - & - \end{bmatrix}$$

以d为中间点, 检测所有点对

$$A_4 = \begin{matrix} & a & b & c & d & e \\ \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix} & \begin{bmatrix} 0 & 1 & 3 & 5 & 7 \\ 6 & 0 & 2 & 4 & 6 \\ 9 & 3 & 0 & 2 & 4 \\ 7 & 1 & 3 & 0 & 7 \\ 12 & 6 & 8 & 5 & 0 \end{bmatrix} \end{matrix}$$

$$path_4 = \begin{bmatrix} - & - & 2 & 3 & 3 \\ - & - & - & 3 & 3 \\ 4 & 4 & - & - & - \\ 2 & - & - & - & 3 \\ 4 & 4 & 4 & - & - \end{bmatrix}$$

以e为中间点, 检测所有点对

$$A_5 = \begin{matrix} & a & b & c & d & e \\ \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix} & \begin{bmatrix} 0 & 1 & 3 & 5 & 7 \\ 6 & 0 & 2 & 4 & 6 \\ 9 & 3 & 0 & 2 & 4 \\ 7 & 1 & 3 & 0 & 7 \\ 12 & 6 & 8 & 5 & 0 \end{bmatrix} \end{matrix}$$

$$path_5 = path_4 = \begin{bmatrix} - & - & 2 & 3 & 3 \\ - & - & - & 3 & 3 \\ 4 & 4 & - & - & - \\ 2 & - & - & - & 3 \\ 4 & 4 & 4 & - & - \end{bmatrix}$$

由上可知

① 假设以a为学校地址, 总长度为

$$l_{a总} = l(a,b) + l(a,c) + l(a,d) + l(a,e) = 1+3+5+7=16$$

② 假设以b为学校地址, 总长度为

$$l_{b总} = l(b,a) + l(b,c) + l(b,d) + l(b,e) = 6+2+4+6=18$$

③ 假设以c为学校地址, 总长度为

$$l_{c总} = l(c,a) + l(c,b) + l(c,d) + l(c,e) = 9+3+2+4=18$$

④ 假设以d为学校地址, 总长度为

$$l_{d总} = l(d,a) + l(d,b) + l(d,c) + l(d,e) = 7+1+3+7=18$$

⑤ 假设以e为学校地址, 总长度为

$$l_{e总} = l(e,a) + l(e,b) + l(e,c) + l(e,d) = 12+6+8+5=31$$

综上所述

选a村建立学校



扫描全能王 创建

上述画“X”的步骤是计算从学校到各村庄的距离和，而题干是计算从各村庄到学校距离，不过原理一样，从计算A一行元素之和变成计算一列元素之和

① 假设以a为学校地址，从各村庄到学校总长度为

解 $l_{a总} = l\langle b, a \rangle + l\langle c, a \rangle + l\langle d, a \rangle + l\langle e, a \rangle = 6 + 9 + 7 + 12 = 34$

② $l_{b总} = l\langle a, b \rangle + l\langle c, b \rangle + l\langle d, b \rangle + l\langle e, b \rangle = 1 + 3 + 1 + 6 = 11$

③ $l_{c总} = l\langle a, c \rangle + l\langle b, c \rangle + l\langle d, c \rangle + l\langle e, c \rangle = 3 + 2 + 3 + 8 = 16$

④ $l_{d总} = l\langle a, d \rangle + l\langle b, d \rangle + l\langle c, d \rangle + l\langle e, d \rangle = 5 + 4 + 2 + 5 = 16$

⑤ $l_{e总} = l\langle a, e \rangle + l\langle b, e \rangle + l\langle c, e \rangle + l\langle d, e \rangle = 7 + 6 + 4 + 7 = 24$

综上所述：选村庄b作为学校满足题意。

- 此题是目前为止简答题中难度最大的了，不是说思路多难而是计算量大，至少花费半个小时
- 每次测点时都要用最新的路径距离，没更新的话用上一个矩阵矩的数据，更新了用当前矩阵
- 本题还有一个解法，大名鼎鼎的“Dijkstra”算法，以求某顶点的^余其他各顶点最短路径长度，不过要使用5次，感兴趣的小伙伴可以试试！



五. 算法答案

1. ① $j < L.length$

② $L.elem[j]$

③ $j++$

④ $L.length = L.length + 1$

→ 本算法比较简单, i 一直是新顺序表的最后一个位置, 所以④ $L.length = i$, j 指向待插入的元素

2. ⑤ $G.arcs[V_0][V]$

⑥ $W < G.vexnum$

⑦ $final[V_0] = TRUE$

⑧ $D[W] < min$

⑨ $!final[W] \ \&\& \ (min + G.arcs[V][W] < D[W])$

⑩ $min + G.arcs[V][W]$

→ 素材来源于步版《数据结构》P189页, 一模一样!!!

六. 编写算法.

1. $count = 0$
 void preorderNonrecursion (BTNode *bt, int k)

```

{
    if (bt != NULL)
    {
        BTNode *stack[maxsize];
        int top = -1;
        BTNode *p;
        stack[++top] = bt;
        while (top != -1)
        {
            p = stack[top--];
            count++;
            if (count == k)
                printf("%d", p->data);
            if (p->rchild != NULL)
                stack[++top] = p->rchild;
            if (p->lchild != NULL)
                stack[++top] = p->lchild;
        }
    }
}
    
```

本代码来源于先序非递归算法的改编.



2. void resort (int a[], int n)

```
{
    int i=0, j=n-1;
    int temp;
    while (i!=j)
    {
        while (i<j && a[i]<0) i++;
        if (i<j)
            temp=a[i];
        while (i<j && a[j]>=0) j--;
        if (i<j)
            a[i++] = a[j];
            a[j] = temp;
        }
    }
```

本代码改编自快速排序，原理是将左边的第一个大于等于0的数与右边起第一个小于0的数交换，直到*i*>*j*。

3. stack<char> s; // 创建一个存字符的栈

```
if (!s.empty()) {
    s.pop();
}
char str[100];
while (gets(str)) { // 多次输入检验
    for (int i=0; i<strlen(str); i++) {
        if (str[i] == '(')
            s.push(str[i]);
        if (str[i] == ')') {
            if (s.top() == '(') {
                s.pop();
            }
            else {
                break;
            }
        }
    }
}
if (!s.empty()) {
    printf("匹配失败");
}
else {
    printf("匹配成功");
}
```

本代码来自学长机试总结代码的一部分核心代码，已经通过测试，其实编写代码是模仿代码即可，主要写主函数！



扫描全能王 创建