

2021 数据结构期末模拟试卷

**题型仅供参考，不一定为期末考试类型

**本试卷主要是为了弥补同学们的知识缺陷，难度会稍大于实际难度

一、选择题

1. 对于一个有 n 个顶点的图：若是连通无向图，其边的个数至少为（），若是强连通有向图，其边的个数至少为（）。

- A. $n-1$, n
- B. $n-1$, $n(n-1)$
- C. n , n
- D. n , $n(n-1)$

答案：A

2. 带头节点的双向循环链表 L 为空的条件是（）

- A. $L \rightarrow next = L \rightarrow prior$
- B. $L \rightarrow next = NULL$
- C. $(L \rightarrow next == L) \&\& (L \rightarrow prior == L)$
- D. $(L \rightarrow next == NULL) \&\& (L \rightarrow prior == NULL)$

答案：C

3. 一颗二叉树的前序遍历和后序遍历分别为 ABCD 和 DCBA，则该二叉树的中序遍历不可能是（）

- A. ABCD
- B. BCDA
- C. CBDA
- D. DBCA

答案：C

4. 对有 2500 个记录的索引顺序表(分块表)进行查找，理想块长为（）

- A. 125
- B. 500
- C. 50
- D. $\log_2(2500)$

答案：C

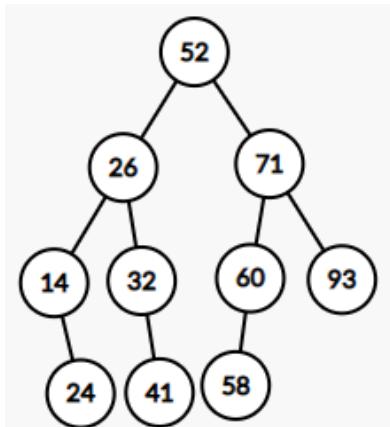
解析：设总块数为 n ，块长为 L ，分块查找时间为 $n/L+L$ ，取极小值的时候 L 为根号 n

5. 从空树开始，依次插入元素 52, 26, 14, 32, 71, 60, 93, 58, 24 和 41 后构成了一颗二叉排序树。该树查找 41 要进行比较的次数为（）

- A. 3
- B. 4
- C. 5
- D. 6

答案：B

解析：最后建成的二叉查找树如下



6. 下列叙述正确的是（）

- A. Floyd 算法可以求带权图的最小生成树
- B. Dijkstra 算法可以求任意带权图的最短路径
- C. 使用普里姆 (Prim) 算法和克鲁斯卡尔 (Kruskal) 算法得到的最小生成树不一定相同
- D. 所有权值最小的边一定会出现在所有的最小生成树中

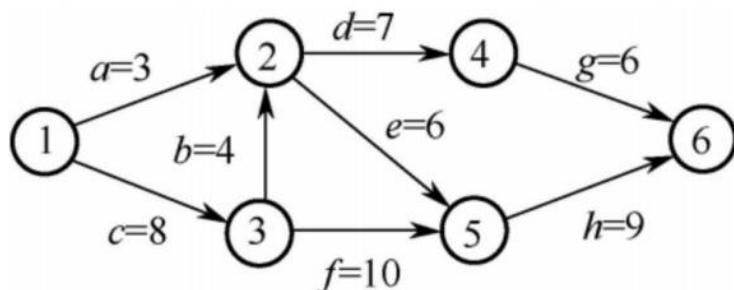
答案：C

7. 下列哪种是哪些判断是正确的（）

- (1) 分块查找是基于关键字比较的查找
 - (2) 置换选择排序产生的归并段长度相等
 - (3) 引入线索二叉树的目的是加快查找结点的前驱或者后继的速度
 - (4) 散列技术是基于关键字存储的查找
- A. (1) (2) (4)
 - B. (2) (3)
 - C. (1) (2) (3) (4)
 - D. (1) (3) (4)

答案：D

8. 下图所示的 AOE 网表示一项包含 8 个活动的工程。活动 d 的最早开始时间和最迟开始时间分别是（）



- A. 3 和 7
- B. 12 和 14
- C. 12 和 12
- D. 3 和 12

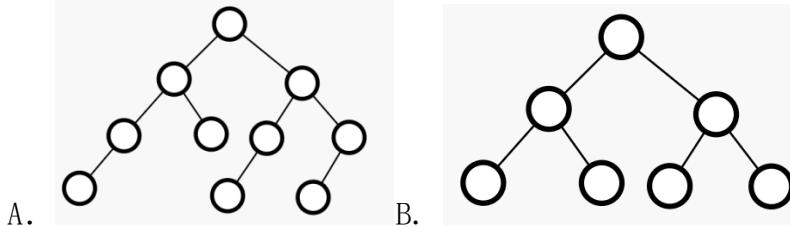
答案：B

解析：由于 $ve(\text{源点}) = 0$, $ve(k) = \max\{ve(j) + \text{weight}(v_j, v_k)\}$,

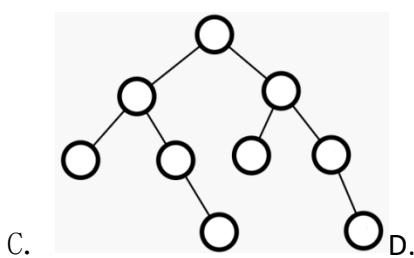
$v1(\text{汇点}) = ve(\text{源点})$, $v1(k) = \min\{v1(j) - weight(vk, vj)\}$,

最终可以算出 $ve(d)$ 是由 $1 \rightarrow 3 \rightarrow 2$ 长度 $8+4=12$ 决定的, $v1(\text{汇点})=27$, 最迟开始时间 $v1(d)$ 由 $6 \rightarrow 4 \rightarrow 2$ 决定为 $27-6-7=14$

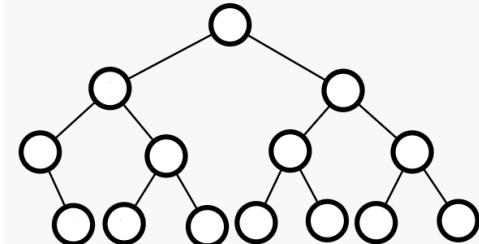
9. 下列二叉树中不可能成为折半查找判定树(不含外部节点)的是()



B.



C.



D.

答案: A

解析: 考察折半查找到判定树的概念, B 是折半时向上取整, C 和 D 是折半时向下取整

10. 已知一颗有 2021 个节点的树, 其叶子结点的个数为 167, 该树对应的二叉树中无右孩子的结点的个数是()

- A. 167 B. 168 C. 1854 D. 1855

答案: D

解析: 由于二叉树的度= $n-1$ =有左孩子的节点数+有右孩子的节点数

无右孩子的节点数= n -二叉树中有右孩子的节点数=二叉树中有节点左孩子数+1=森林中有儿子的节点数+1= n -森林中叶子节点数+1=2021-167+1=1855

二、填空题

1. N 个叶子节点的哈夫曼树总共有_____个度为 2 的节点。

答案: $N-1$

2. 有 4 个节点的互不同构的二叉树有_____种

答案: 14

3. 具有 n 个结点的二叉链表中, 有_____个空指针, 有_____个指向孩子结点的指针。

答案: $n+1, n-1$ 。

4. 一颗完全二叉树有 1005 个节点, 它的叶节点的个数是_____

答案: 503

5. 后缀算术表达式 $1\ 2\ +\ 8\ 2\ 2\ *\ /$ 的中缀表达式为_____, 前缀表达式为_____。

答案: $(1 + 2) + 8 / (2 * 2), + + 1 2 / 8 * 2 2$

6. 若无向图 $G = (V, E)$ 中含有 7 个顶点, 则保证图 G 在任何情况下都是连通的, 则需要的边数最少是_____

答案: 16

7. 若 B-树的阶数 $m = 5$, 高度 $h = 3$, 则关键字总数至少为_____

答案: 17

解析: $N \geq 2 \left\lceil \frac{m}{2} \right\rceil^{h-1} - 1 = 17$, 也可以枚举

8. 有 22 个节点的 AVL 树高度至多为_____

答案: 6

解析 $N_1=1, N_2=2, N(i+2)=N(i+1)+N(i)+1, \dots, N(6)=20, N(7)=33$

三、简答题 (4+4 分)

1.

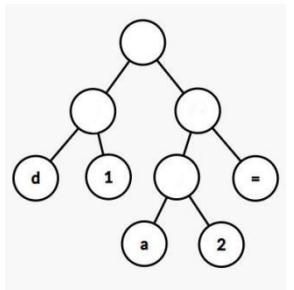
(1) 降序; 每次当前的数字都会插入到有序部分的最前面, 故时间复杂度 $O(n^2)$

(2) 升序 (或降序); 形成一个斜树, 故时间复杂度 $O(n^2)$

2, (6+3 分)

(1) 一种可能的答案见右图:

(2) 16



3. (8 分)

使用一个存放这些有理数的最小堆 (2 分), 先放入 $1/1, 1/2, \dots, 1/n$ 这 n 个数 (2 分)。每次从堆中拿出最小的有理数 (设为 i/j) 并输出, 然后向堆中放入 $(i+1)/j$, 重复执行 n 次 (3 分)。使用堆, 且堆的大小始终为 n , 故时间复杂度 $O(n \log n)$ (1 分)。

(另: 胜者树/败者树的做法也是正确的)

四、程序设计题

1. (分析: 由于为完全二叉树, 且使用顺序存储结构的形式进行编号 $1 \cdots N$, 故

每个节点 $i (>1)$ 的父亲为 $\left\lfloor \frac{i}{2} \right\rfloor$, 且深度取决于其二进制表示下最高位的 1。先考

虑令 x 和 y 中深度大者 ($x < y$, 所以只可能为 y) 不断向上走使得深度相同, 然后两者同时向上直到相遇, 此时即为答案。)

(1) 算法的基本设计思想: (2 分)

首先, 令 y 不断向上 (自除 2) 使得 x 和 y 深度相同;

然后, 令 x 和 y 同时向上直到相遇, 此时即为答案。

(2) 算法描述: (5 分)

算法实现如下:

```
int Lca(int x, int y)
{
    while (y/2>=x) y/=2;
    if (x==y) return x;
    int x1 = x, y1 = y/2;
    while (x>1 && y>1 && x/2!=y/2) x/=2, y/=2;
    while (x1>1 && y1>1 && x1/2!=y1/2) x1/=2, y1/=2;
```

```

        if (x/2==y/2) return x/2;
        else return x1/2;
    }

```

(3) 时间和空间复杂度: (2 分)

由于完全二叉树深度不超过 $\log_2 n$, 故时间复杂度为 $O(\log n)$ 。空间复杂度为 $O(1)$ 。

2. (分析: 由于是无向无边权的联通图, 故需要生成树的树高最小, 等价于每个节点到根节点的距离尽量小, 故可用 BFS 求单源最短路, 所有点的距离最大者就是生成树的最小半径。)

(1) 算法的基本设计思想: (2 分)

以根节点为起点, 对图做广度优先搜索, 记录每个点到根节点的最小距离 $dist$;
所有 $dist$ 取最大值即为该生成树的最小半径。

(2) 算法描述: (5 分)

```

int solve(Graph *G)
{
    int dist[NumVertices+1], ans=0;
    bool visited[NumVertices+1];
    for (int i=1; i<=NumVertices; i++) visited[i]=false;
    EdgeNode *p; QUEUE Q; MAKENULL(Q); // Q 是队列
    visited[1]=true; dist[1]=0;
    ENQUEUE(1, Q); // 节点 1 进队
    while (!Empty(Q)) {
        int i=DEQUEUE(Q); // 队首出队
        p=G->vexlist[i].firstedge;
        while (p) { // 遍历邻接的点
            if (!visited[p->adjvex]) {
                visited[p->adjvex]=true;
                dist[p->adjvex]=dist[i]+1;
                ans=max(ans, dist[p->adjvex]); // 取 dist 最大为答案
                ENQUEUE(p->adjvex, Q); // 入队
            }
            p=p->next;
        }
    }
    return ans;
}

```

(3) 时间和空间复杂度: (2 分)

使用广度优先搜索, 时间复杂度是 $O(n+e)$; 使用队列数据结构, 空间复杂度 $O(n)$ 。

3. (分析: 第 1 小题: 考虑 $next$ 数组, 求的是最长的相同的前缀和后缀的长度。
注意到对于一个字符串, 记 $border$ 为其最长的相同的前缀/后缀字符串, 那么一个字符串的 $border$ 的 $border$ 就是其第二长的相同的前缀和后缀。故先求出 $next$

数组，再用 `next[n]` 不断调用 `next` 数组即可求得最小的。第 2 小题：同第一小题，顺次枚举 i ，每个 i 都能通过不断调用 `next` 得到答案并更新，类似于并查集的路径压缩操作，时间复杂度是 $O(n)$ 。)

第 1 小题：

(1) 算法的基本设计思想：(1 分)

首先求出 `next` 数组，然后从 `next[n]` 开始不断调用 `next` 数组，即可求得最小长度。

(2) 算法描述：(3 分)

```
int ShortestBorder(int n, char S[])
{
    int next[n+1], ans;
    next[0] = next[1] = 0;
    for (int i=1; i<n; i++) {
        int j = next[i];
        while (j && S[i]!=S[j]) j = next[j];
        next[i+1] = S[i]==S[j] ? j+1 : 0;
    }
    ans = next[n];
    while (next[ans]) ans = next[ans];
    return ans;
}
```

(3) 时间和空间复杂度：(1 分)

只需求 `next` 数组和上述 `while` 语句，时间复杂度是 $O(n)$ ，空间复杂度是 $O(n)$ 。

第 2 小题：(2 分)

顺次枚举 i ，每个 i 都能通过不断调用 `next` 得到答案，并更新给当前的 `next[i]`；时间复杂度和空间复杂度是 $O(n)$ 。

对答案的解释：

```
for (int i=1; i<=n; i++) {
    int j = next[i];
    while (next[j]) j = next[j];
    next[i] = j;
}
```