哈工大 2017 年

一．单项选择题（10 分）

片纸鉴心 诚信不败

一、单项选择题：（1-10 小题，每小题 1 分，共 10 分。在每小题给出的四个选项中，请选出一项最符合题目要求的。）

1. 一个算法的执行时间是 $2n^3-3n^2\log_2 n-4n$，其时间复杂度为（ ）。

A. $O(n^3)$　B. $O(n^2\log_2 n)$　C. $O(n\log_2 n)$　D. $O(n^2)$

2. 设一组初始记录关键字序列为(315，205，674，924，627，893)，则用基数排序需要进行（ ）趟的分配和回收才能使得初始关键字序列变成有序序列。

A. 5　B. 4　C. 3　D. 6

1.一个算法的执行时间是 2n^3-3n^2log2(n)-4n，其时间复杂度为（A）

2.设一组初始记录关键字序列为（315,205,674,924,627,893），则用基数排序需要进行（C）趟的分配和回收才能使得初始关键字序列变成有序序列。

设一组初始记录关键字序列为(345，253，674，924，627)，则用基数排序需要进行（）趟的分配和回收才能使得初始关键字序列变成有序序列。

单选题　2018-09-28 12:13:15　0　5:

A.3
B.4
C.5
D.8

正确答案

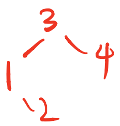
A

最大关键字为多少位，则需进行多少趟的分配与回收。

3. 某二叉树的前序遍历序列为 3，1，2，4，中序遍历序列为 1，2，3，4，则其层序遍历序列为（ ）。

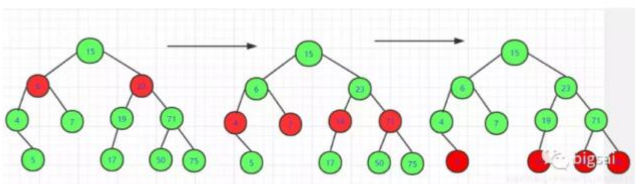
A. 2，1，4，3　B. 3，1，4，2
C. 3，4，2，1　D. 3，2，4，1

3. 某二叉树的前序遍历序列为 3,1,2,4，中序遍历序列为 1,2,3,4，则其层序遍历序列为（B）



3 1 4 2

层序遍历



4. 设无向图 G 中有 n 个顶点 m 条边，则其对应的邻接表中，顶点表结点和边表结点的个数分别为（ ）。

A. m，n　B. n，m　C. n，2m　D. 2m，n

4. 设无向图 G 中有 n 个顶点 m 条边，则其对应的邻接表中，顶点表结点和边表结点的个数分别为（C）（n，2m） n 2m

n 和顶点，m 条边

有向图：邻接表中，顶点表结点和边表结点的个数分别为 n，m

无向图：邻接表中，顶点表结点和边表结点的个数分别为 n，2m

9 4 2 3

二分向下取整

5. 若已排好序的 18 个元素存放在一维数组 A[19]中，第一个元素放 A[1]中，现进行二分（折半）查找，则查找 A[3]的比较序列的下标依次为（ ）。

A. 1，2，3 B. 9，5，2，3 C. 9，5，3 D. 9，4，2，3

5.选 D

6. 设有 n 个关键字具有相同的散列值，则用线性探测法把这 n 个关键字映射到初始为空的散列表中需要做( ) 次线性探测。

A. n^2 B. n（n+1） C. n（n+1）/2 D. n（n-1）/2

6.设有 n 个关键字具有相同的散列值，则用线性探测法把这 n 个关键字映射到初始为空的散列表中需要做（）次线性探测。

$1+2+\cdots+n=\frac{n(n+1)}{2}$

7. 无向图的邻接矩阵对应的二维数组是 A，现将其上三角（即主对角线及以上）元素按行优先顺序压缩存储在一个足够大的一维数组 Sa 中。若 Sa[0]=A[0][0]，Sa[17]=A[2][2]，则矩阵元素 A[5][3]的值存放在一维数组 Sa 的第（ ）个单元中。

A. 25 B. 26 C. 27 D. 33

7. 无向图的邻接矩阵对应的二维数组是 A，现将其上三角（即主对角线及以上）元素按行优先顺序压缩存储在一个足够大的一维数组 Sa 中。若 Sa[0]=A[0][0]，Sa[17]=A[2][2]，则矩阵元素 A[5][3]的值存放在一维数组 Sa 的第（）个单元中。27

A. 树中每个结点至少有 m 棵子树；

B. 根结点至少有 2 棵子树；

C. 除根结点和失败结点外，所有结点至少有 ⌈m/2⌉ 棵子树；

D. 所有的终端结点(失败结点）都位于同一层。

8.

D. 所有的终端结点(失败结点）都位于同一层。

9. 设一组初始记录关键字序列为(50，40，95，20，15，70，60，45)，则以增量 d=4 的一趟希尔（Shell）排序结束后前 4 条记录关键字为（B）。

A. 40，50，20，95 B. 15，40，60，20

C. 15，20，40，45 D. 45，40，15，20

9. 设一维初始记录关键字序列为（50,40,95,20,15,70,60,45），则以增量 d=4 的一趟希尔（Shell）排序结果后前 4 条记录关键字为（）。

15 40 60 20

10、以下四组数据分别顺序存储在一维数组中，堆结构的选项是（ B ）。

1 2 3 4 5

A. 10，15，56，20，30　　　　B. 10，20，56，15，30

C. 15，20，56，10，30　　　　D. 56，20，10，15，30

$k=8$

$k-1=7$

$2^{k-1}=119$

$2^{k-1}+1=120$

$2^k-1=120$

$2^k=120$

$k=7$

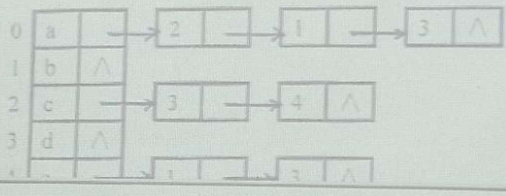
二．填空题（10 分）

二、填空题：（11-15 小题，每空 1 分，共 10 分。）

11. 具有 120 个结点的完全二叉树，其高度的最小值和最大值分别是（ 7 ）和（ 8 ）。

12. 弗洛伊德（Floyd）和迪杰斯特拉（Dijkstra）算法的适用条件分别为（ 无负环 ）和（ 无负权 ）。

13. 已知一个有向图的邻接表存储结构图 1 所示：从顶点 a 出发，深度优先（DFS）搜索和广度优先搜索（BFS）的输出序列分别是（ ）和（ ）。



三．简答题（共 25 分）

三、简答题：（共 25 分）

1.（10 分）关键字的输入顺序为 2，4，6，8，10，12，14，16，18，分别建立初始为空的二叉搜索树（BST）和二叉平衡树（AVL），要求：

（1）画出所建立的 BST，并计算其在等概率情况下搜索成功的平均查找长度 ASLb；

（2）画出所建立的 AVL，并计算其在等概率情况下搜索成功的平均查找长度 ASLa；

（3） 比较（1）和（2），说明数据分布与所建二叉树的结构形态关系，以及对搜索效率的影响。

1.（10 分）

2.（8 分）已知某文件预处理后，得到 5 个初始归并段，归并段中的每个数据占一个磁盘读写单位，初始归并段长度分别为［20，30，10，5，30］。若只有 3 个内存缓冲区，请设计一个读写磁盘次数最少的排序方案。要求：给出设计步骤，并计算磁盘的读写次数。

2. （8 分）

四．算法分析题（共 25 分）

四、算法设计题：（共 25 分）

按以下要求设计算法：

（1）给出算法的基本设计思想。

（2）使用 C 或 C++或 Java 语言，给出相关的数据类型定义。

（3）根据设计思想，采用 C 或 C++或 Java 语言描述算法，关键之处给出注释。

（4）说明你所设计算法的时间复杂度。

1.（8 分）在一个长度为 n 整数序列中，奇数元素和偶数元素各占一半，存放在数组 A[n]中，请设计一个时间和空间尽可能高效的算法 NewSequ(int A[]，int n)，重新排列这些整数，使奇数元素存放在奇数单元，偶数元素存放在偶数单元，说明你所设计算法的时间和空间复杂度。

1.（8 分）

2.（10 分）已知二叉树 BT 采用左右链表示法（亦称二叉链表）作为其存储结构，二叉树的结点结构如下：[lchild][data][rchild]。

请给出二叉树的类型 BTree 定义，并设计一个非递归算法 FirstNode (BTree BT)，直接返回 BT 的后序遍历的第一个被访问的结点，说明你所设计算法的时间复杂度。

“直接”的含义是，不能通过后序遍历得到二叉树的后序序列，然后返回后序序列的第一个结点。

2.（10 分）

3.（7 分）设在 4 地（A、B、C、D）之间架设有 6 座桥，如图 2 所示：

图2 4地6桥图

3.（7 分）

第1章　时间复杂度分析

第2章　线性表

一、单项选择题

1.个算法应该是（ ）。

A.程序　B.问题求解步骤的描述　C.要满足五个基本特性　D. A和C

算法代表了对问题求解步骤的描述。

算法有 5 个重要特性：

1. 输入：
2. 输出：

3. 确定性：
4. 有限性：
5. 可行性

程序与算法不同，它是算法的具体实现，通常采用某种程序设计语言描述。
可以不满足算法的性质 4，即有限性。例如，执行无限循环的程序就不能称其为算法。

选 B（C? --）

2.某算法的时间复杂度为$O(n^2)$，表明该算法的（ ）。

A.问题规模是n^2　B.执行时间等于n^2

C.执行时间与n^2成正比　D.问题规模与n^2成正比

T(n)为算法的执行时间，n 为问题规模，f(n)为 n 的某个函数，若 T(n) = O(f(n))，则称 f(n)为算法的时间复杂性的上界。

时间复杂度为$O(n^2)$，说明算法的执行时间$T(n)<=c * n^2$(c为比例常数)，即$T(n)=O(n^2)$，时间复杂度T(n)是问题规模n的函数，其问题规模仍然是n而不是n^2。

选 C

3.以下算法的时间复杂度为（ ）。

```
void fun(int n) {
    int i=1;
    while(i<=n)
        i=i*2;
}
```

log n

A. O(n)　B. $O(n^2)$　C. $O(n\log_2 n)$　D. $O(\log_2 n)$

3. D

基本运算是i=i*2，设其执行时间为T(n)，则$2^{T(n)}<=n$，即$T(n)<=\log_2 n=O(\log_2 n)$。

假设循环次数为 t，则循环条件满足 2^t < n。
可以得出，执行次数t = log(2)(n)，即 T(n) = log(2)(n)，可见时间复杂度为 O(log(2)(n))，即 O(log n)。

4.【2011年计算机联考真题】
设n是描述问题规模的非负整数，下面程序片段的时间复杂度是（）。

```
x=2;
while(x<n/2)
    x=2*x;
```

4. A

在程序中，执行频率最高的语句为“x=2*x”。设该语句共执行了 t次，则$2^{t+1}=n/2$，故$t=\log_2(n/2)-1=\log_2 n-2$，得$T(n)=O(\log_2 n)$。

5.【2012年计算机联考真题】

求整数n (n>=0)阶乘的算法如下，其时间复杂度是（）。

```
int fact(int n){
    if (n<=1) return 1;
    return n*fact(n-1);
}
```

A. $O(\log_2 n)$　B. O(n)　C. $O(n\log_2 n)$　D. $O(n^2)$

O(n)!

5.　B

本题是求阶乘n!的递归代码，即n*(n-1)*...*1共执行n次乘法操作，故T(n)=O(n)。

6.有以下算法，其时间复杂度为（）。

```
void fun (int n){
    int i=0;
    while(i*i*i<=n)
        i++;
}
```

A. O(n)　B. O(nlogn)　C. $O(\sqrt[3]{n})$　D. $O(\sqrt{n})$

$i<\sqrt[3]{n}$

C

6.　C

算法的基本运算是i++，设其执行时间为T(n)，则有，T(6)*T(n)*T(n)<=n，即$T(n)^3$<=n。故有，$T(n) \leqslant \sqrt[3]{n} = O(\sqrt[3]{n})$。

7.程序段

```
for(i=n-1;i>1;i--)
  for(j=1;j<i;j++)
     if (A[j]>A[j+1])
         A[j]与 A[j+1]对换;
```

其中n为正整数，则最后一行的语句频度在最坏情况下是（）。

A. O(n)　B. O(nlogn)　C. $O(n^3)$　D. $O(n^2)$

7.　D

当所有相邻元素都为逆序时，则最后一行的语句每次都会执行。此时，

$$T(n) = \sum_{i=2}^{n-1}\sum_{j=1}^{i} 1 = \sum_{i=2}^{n-1} i = (n-2)(n+1)/2 = O(n^2) = O(n^2)$$

所以在最坏情况下的该语句频度是$O(n^2)$。

8.以下算法中加下划线语句的执行次数为（）。

```
int m=0, i, j;
for(i=1;i<=n;i++)
   for(j=1;j<=2 * i;j++)
      m++;
```

A. n(n+1)　B. n　C. n+1　D. n^2

[1-4]　[1-6]

2　4　6　...　2n

n(n+1)

等差数列求和公式

$$S_n = \frac{(a_1 + a_n)n}{2} = a_1 n + \frac{n(n-1)d}{2}$$

8. A

m++语句的执行次数为

$$\sum_{i=1}^{n}\sum_{j=1}^{2i} 1 = \sum_{i=1}^{n} 2i = 2\sum_{i=1}^{n} i = n(n+1)$$

具体多少次，不是化为 n^2。

9.下面说法错误的是（ ）。

Ⅰ.算法原地工作的含义是指不需要任何额外的辅助空间

Ⅱ.在相同的规模n下，复杂度O(n)的算法在时间上总是优于复杂度$O(2^n)$的算法

Ⅲ.所谓时间复杂度是指最坏情况下，估算算法执行时间的一个上界

Ⅳ.同一个算法，实现语言的级别越高，执行效率就越低

A. Ⅰ B. Ⅰ、Ⅱ C. Ⅰ、Ⅳ D. Ⅲ

9. A

Ⅰ，算法原地工作是指算法所需的辅助空间是常量。Ⅱ，题中是指算法的时间复杂度，不要想当然认为是程序（该算法的实现）的具体执行时间，而赋予n一个特殊的值。时间复杂度为O(n)的算法，必然总是优于时间复杂度为$O(2^n)$的算法。Ⅲ，时间复杂度总是考虑在最坏情况下的时间复杂度，以保证算法的运行时间不会比它更长。Ⅳ为严蔚敏教材的原话。

二、综合应用题

1.一个算法所需时间由下述递归方程表示，试求出该算法的时间复杂度的级别（或阶）。

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T(n/2) + n & n > 1 \end{cases}$$

式中，n是问题的规模，为简单起见，设n是2的整数幂。

1.解答：

时间复杂度为$O(n\log_2 n)$。

设$n=2^k(k>=0)$，根据题目所给定义，有$T(2^k) = 2T(2^{k-1}) + 2^k = 2^2T(2^{k-2}) + 2 \times 2^k$，由此，可得一般递推公式$T(2^k) = 2^iT(2^{k-i}) + i \times 2^k$，进而，可得$T(2^k) = 2^kT(2^0) + k \times 2^k = (k+1)2^k$，即$T(n) = 2^{\log_2 n} + \log_2 n * n = n(\log_2 n + 1)$，即为$O(n\log_2 n)$。

?为什么--============-

2.分析以下各程序段，求出算法的时间复杂度。

```
// 程序段③
for(i=1;i<=n;i++)
    for(j =1;j <=i;j ++)
        for(k=1;k<=j;k++)
            x++;
```

n^3

```
// 程序段④
for(i=0;i<n;i++)
    for(j=0;j<m;j++)
        a[i] [j]=0;
```

n^2

$n \times m$

③ x++是基本语句, $T(n)=O(\sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=1}^{j}l)=O(\frac{1}{6}n^3)=O(n^3)$ 。

④a[i][j]=0是基本语句，内循环执行m次，外循环执行n次，共执行了 m*n次，所以 T(m, n)=O(m*n)0

$$\sum_{i=1}^{n}\sum_{j=1}^{i} j=\sum_{i=1}^{n}\frac{i(i+1)}{2}$$
$$=\frac{1}{2}\left[\sum_{i=1}^{n}(i^2+i)\right]$$
$$=\frac{1}{2}\left[\sum_{i=1}^{n}i^2+\sum_{i=1}^{n}i\right]$$
$$=\frac{1}{2}\left[\frac{n(n+1)(2n+1)}{6}+\frac{n(n+1)}{2}\right]$$
$$=\frac{n(n+1)(n+2)}{6}$$

第3章 树

3.1 基本术语

3.2 二叉树

https://mp.weixin.qq.com/s?src=11&timestamp=1578192130&ver=2077&signature=yvm1JAiV0wD-i80QJJHU9MuTVJKXQWm*ASrcvtuBomV1PqtapB4Y0aLojpy*8y40FTEWAv1ykHGtQiJ6YwSuZdSd6e7HgmzSyJumyeM0pFxbB11JqLHvRcL9U7a7Sgj2&new=1

3.3 堆

3.4 选择树

3.5 树

3.6 森林和二叉树间的转换

3.7 树的应用

第4章 图

4.1 基本定义

4.2 图的表示

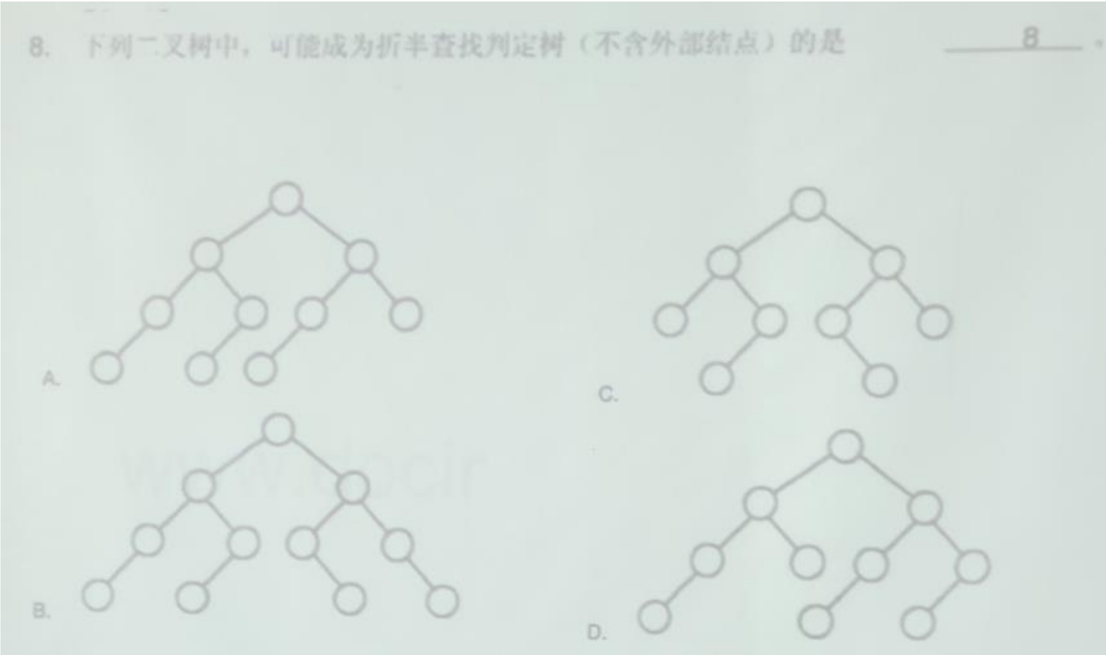
4.3 图的搜索

4.4 图与树的联系

第5章　查找

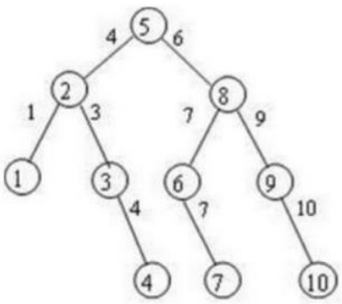
5.1 线性查找

5.2 折半查找（二分查找）



因此可以得知，折半查找的二叉判定树对于所有结点，左子树结点个数<=右子树结点个数。即：

1. 若某序列总长n为奇数，左右子树结点个数相等；
2. 若某序列总长n为偶数，左字数结点个数=右子树结点个数-1.



选？（**4 个都错**===）

5.3 分块查找

第6章 排序

5.4 二叉查找树

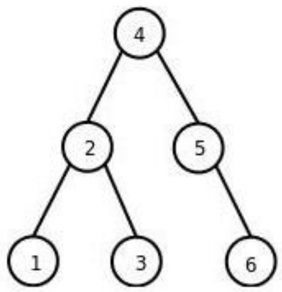
1. 二叉查找树=二叉排序树
2. 二叉判定树

描述折半查找过程的二叉树为判定树。

判定树首先是一个二叉排序树，具有 n 个结点的判定树具有与具有 n 个结点的完全二叉树的深度完全相同，其深度为：

$$\lfloor \log_2 n \rfloor + 1$$

。



在折半查找时，查找成功不成功，和给定值比较的次数最多为

$$\lfloor \log_2 n \rfloor + 1$$

7、下列选项中，不能构成折半查找中关键字比较序列的是

A、500, 200, 450, 180　　B、500, 450, 200, 180

C、180, 500, 200, 450　　D、180, 200, 500, 450

选 A

5.5AVL 树

5.6B – 树与 B+树

5.7Trie 树

5.8 散列法

第6章　排序

6.5 基数排序

第7章　文件与外部排序