# 电子科技大学

# 数据结构与算法

# 习
# 题
# 册

姓名：_____

学号：_____

专业：□ 计算机大类

□ 网络空间安全

□ 物联网+实验班

□ 英才实验班

仅限 2020 年秋季使用
P0800335.06、P2700530.01、P0800335.01、H0816130.01

周益民 博士 教授
2020 年 9 月 7 日

# 查找

## 1. 选择题

(1) 静态查找和动态查找的根本区别在于(　　)。
A．它们的逻辑结构不一样　　B．施加在其上的操作不同
C．所包含的数据元素类型不一样　　D．存储实现不一样

(2) 平均查找长度与查找集合中记录个数 n 无关的查找方法是(　C)　。
A．折半查找　　B．平衡二叉树的查找　　C．散列查找　　D．不存在

(3) 在以下数据结构中，(　　)的查找效率最低。
A．有序顺序表　　B．二叉排序树　　C．堆　　D．散列表

(4) 对线性表进行顺序查找，要求线性表的存储结构为(　　)。
A．散列存储　　B．顺序存储　　C．链接存储　　D．顺序存储或链接存储

(5) 用顺序查找方法在长度为 n 的线性表中进行查找，在等概率情况下，查找成功的平均查找长度为(　　)。
A．$n$　　B．$n/2$　　C．$(n-1)/2$　　D．$(n+1)/2$

(6) 折半查找判定树不属于(　　)。
A．平衡二叉树　　B．二叉排序树　　C．完全二叉树　　D．二叉树

(7) 当 n 足够大时，在有序顺序表中进行折半查找，假设顺序表中每个元素的查找概率相同，则查找成功的平均查找长度为(　　)。
A．$(n+1)/2$　　B．$n/2$　　C．$\log_2(n+1)-1$　　D．$\log_2(n+1)$

(8) 对具有 14 个元素的有序表 $R[14]$ 进行折半查找，查找 $R[3]$ 时比较需要比较(　　)。
A．$R[0]R[1]R[2]R[3]$　　B．$R[6]R[2]R[4]R[3]$
C．$R[0]R[13]R[2]R[3]$　　D．$R[6]R[4]R[2]R[3]$

(9) 对有序表 $A[1..17]$ 进行折半查找，则查找长度为 5 的元素下表依次是(　　)。
A．8，17　　B．5，10，12　　C．9，16　　D．9,17

(10) 用 n 个键值构造一棵二叉排序树，其最低高度为(　　)。
A．$n/2$　　B．$n$　　C．$\lfloor \log_2 n \rfloor$　　D．$\lfloor \log_2 n+1 \rfloor$

(11) 在含有 n 个结点的二叉排序树中查找一个关键码，最多进行(　　)次比较。
A．$n/2$　　B．$\log_2 n$　　C．$\log_2 n+1$　　D．$n$

(12) 二叉排序树中，最小值结点的(　　)。
A．左指针一定为空　　B．右指针一定为空

C．左、右指针均为空　D．左、右指针均为空

(13) 已知 10 个元素(54,28,16,73,62,95,60,26,43)，按照依次插入的方法生成一棵二叉排序树，查找值为 62 的结点所需比较次数为(　　)。
A．2　　B．3　　C．4　　D．5

(14) 已知数据序列为(34,76,45,18,26,54,92,65)，按照依次插入结点的方法生成一棵二叉排序树，则该树的深度为(　　)。
A．4　　B．5　　C．6　　D．7

(15) 按(　　)遍历二叉排序树得到的序列式一个有序序列。
A．前序　　B．中序　　C．后序　　D．层次

(16) 下列二叉排序树中查找效率最高的是(　　)。
A．平衡二叉树　　B．二叉查找树
C．没有左子树的二叉排序树　　D．没有右子树的二叉排序树

(17) 在二叉排序树上查找关键码为 28 的结点(假设存在)，则依次比较的关键码有可能是(　　)。
A．30，36,28　　B．38，48，28
C．48，18，38，28　　D．60，30，50，40，38，36

(18) 不可能生成如图 4-1 所示二叉排序树的关键码序列是(　　)。
A．{4,2,1,3,5}　　B．{4,2,5,3,1}
C．{4,5,2,1,3}　　D．{4,5,1,2,3}

(19) 一棵二叉排序树如图 4-2 所示，该二叉树是(　　)。
A．平衡二叉树　　B．二叉排序树
C．堆　　D．以上都不是

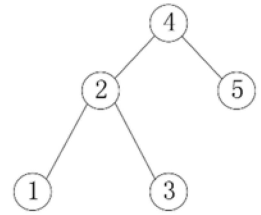

图 4-1　一棵排序二叉树

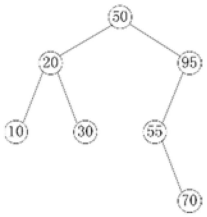

图 4-2　一棵二叉树

(20) 关于二叉排序树，下面说法中正确的是(　　)。
A．二叉排序树是动态树表，在插入新节点时会引起树的重新分裂或组合
B．对二叉排序树进行层次遍历可得到有序序列
C．在构造二叉排序树时，若插入的关键码有序，则二叉排序树的深度最大
D．在二叉排序树中进行查找，关键码的比较次数不超过节点数的一半

(21) 具有 5 层的平衡二叉树至少有(　　)个结点。
A．10　　B．12　　C．15　　D．17

(22) 在平衡二叉树中插入一个结点后造成了不平衡，设最低的不平衡结点为 A，并已知 A 的左孩子的平衡因子为 0 右孩子的平衡因子为 1，应做(　　)型调整以使其平衡。
A．LL　　B．LR　　C．RL　　D．RR

(23) 一棵完全二叉树一定是一棵(　　)。
A．平衡二叉树　　B．二叉排序树　　C．堆　　D．哈夫曼树

(24) 散列技术中的冲突指的是(　　)。
A．两个元素具有相同的序号　　B．两个元素的键值不同，而其他属性相同
C．数据元素过多　　D．不同键值的元素对应于相同的存储地址

(25) 设散列表表长 $m=14$，散列表 $H(k)=k \bmod 11$。表中已有 15、38、61、84 四个元素，如果用线性探测法处理冲突，则元素 49 的存储地址是(　　)。
A．8　　B．3　　C．5　　D．9

(26) 为一组关键码 {87,73,25,55,90,28,31,17,101,22,3,62} 构造散列表，设散列函数为 $H(key)=key \bmod 11$，在用链地址法处理后位于同一链表中的是(　　)。
A．81,90　　B．31,101　　C．3,78　　D．62,73

(27) 在采用线性探测法处理冲突所构成的散列表上进行查找，可能要探测多个位置，在查找成功的情况下，所探测的这些位置的键值(　　)。
A．一定都是同义词　　B．一定都不是同义词
C．不一定都是同义词　　D．$O(n^3)$

(28) 下面关于散列查找的说法正确的是(　　)。
A．散列函数越复杂越好，因为这样随机性好，冲突小
B．除留取余法是所有散列函数中最好的
C．不存在特别好与怀的散列函数，要视情况而定
D．若在散列表中删去一个元素，只要简单地将该元素删去即可

(29) 关于散列查找说法不正确的有几个(　　)。
- 采用链地址法解决冲突，查找一个元素的时间是相同的
- 采用链地址法解决冲突，若插入总是在链首，则插入任一个元素的时间是相同的
- 用链地址法解决冲突易引起聚集现象
- 再散列法不易产生聚集

A．1　　B．2　　C．3　　D．4

(30) 关于散列查找，下面说法正确的是(　　)。
A．再散列法处理冲突不会产生聚集
B．散列表的装填因子越大说明空间利用率越高，因此应使装填因子尽可能大
C．散列函数选择得好可以减少冲突现象
D．对任何关键码结合都无法找不到不产生冲突的散列函数

(31) 散列查找方法一般适用于(　　)情况下的查找。
A．查找表为链表　　B．查找表为有序表

C．关键码集合比地址集合大得多　　D．关键码集合与地址集合存在对应关系

(32) 设哈希表长 $m=15$，哈希函数 $H(key)=key \bmod 13$，关键码集合为{53, 17, 12, 61, 70, 87, 25, 64, 46}，采用二次探测再散列方法处理冲突，则查找成功的平均比较长度为(　　)。

A．1.4　　B．1.6　　C．1.8　　D．2.0

(33) 假设有 10 个关键码，他们具有相同的散列函数值，用线性探测法把这 10 个关键码存入散列表中至少需要做(　　)次探测。

A．110　　B．100　　C．55　　D．45

(34) 采用开放定址法解决冲突的散列查找中，发生聚集的原因主要是(　　)。

A．数据元素过多　　B．装填因子过大

C．散列函数选择不当　　D．解决冲突的算法不好

(35) 对具有 n 个关键码的散列表进行查找，平均查找长度是(　　)。

A．$O(\log_2 n)$　　B．$O(n)$　　C．$O(n\log_2 n)$　　D．与 n 无关

(36) 设某有向图含有 n 个顶点 e 条边，则该有向图采用邻接表表示，则拓扑排序算法时间复杂度为(　　)。

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

## 2. 简答题

(1) 将关键字序列(7、8、1、2、20、28)散列存储到散列列表中，散列表的存储空间是一个下标从 0 开始到 6 的一个一维数组。散列函数为：$H(key)=key\%7$，处理冲突采用线性探测再散列法。(a) 请画出所构造的散列表；(b) 分别计算等概率情况下，查找成功的平均查找长度。

(2) 根据关键字序列(1、3、4、5、7、8、9、11、12)绘制出二分查找树。分别计算给出查找成功和查找失败的平均查找长度。

## 3. 算法设计

(1) 顺序查找算法 SeqSearch。

```
int SeqSearch(int r[],int n,int k)
```

(2) 折半查找非递归算法 BinSearch1。

```
int BinSearch1(int r[],int n,int k)
```

(3) 折半查找递归算法 BinSearch2。

```
int BinSearch2(int r[],int low,int high,int k)
```

# 排序

## 1. 选择题

(1) 排序算法的稳定性是指(　　) 。
A．经过排序之后，能使值相同的数据保持原排序中的相对位置不变
B．经过排序之后，能使值相同的数据保持原排序中的绝对位置不变
C．排序算法的性能与被排序元素的数量关系不大
D．排序算法的性能与被排序元素的数量关系密切

(2) 一个待排序的 n 个记录可分为 n/k 组，每组包含 k 个记录，且任一组内的各记录分别大于前一组内的所有记录切小于后一组内的所有记录，若采用基于比较的排序方法，其时间下限为(　　)。
A．$O(k\log_2 k)$　　B．$O(k\log_2 n)$　　C．$O(n\log_2 k)$　　D．$O(n\log_2 n)$

(3) 关于排序，下列说法正确的是(　　)。
A．稳定的排序方法优于不稳定的排序方法，因为稳定的排序方法效率较高
B．对同一个线性表使用不同的排序方法进行排序，得到的排序结果可能不同
C．排序方法都是在排序表上实现的，在链表上无法实现排序方法
D．在排序表上实现的排序方法在链表上也可以实现

(4) 下列不属于内部排序方法的是(　　)。
A．归并排序　　B．拓扑排序　　C．堆排序　　D．插入排序

(5) 用直接插入排序对下面的四个序列进行由小到大排序，元素比较次数最少的是(　　)。
A．94,32,40,90,80,46,21,69　　B．21,32,46,40,80,69,90,94
C．32,40,21,46,69,94,90,80　　D．90,69,80,46,21,32,94,40

(6) 当待排序序列基本有序或个数较小的情况下，最佳的内部排序方法是(　　)。
A．直接插入排序　　B．起泡排序　　C．简单选择排序　　D．快速排序

(7) 数据序列$\{8,9,10,4,5,6,20,1,2\}$只能是(　　)的两趟排序的结果。
A．选择排序　　B．冒泡排序　　C．插入排序　　D．堆排序

(8) 对数据序列$\{15,9,7,8,20,-1,4\}$进行排序，一趟后数据序列变为$\{9,15,7,8,20,-1,4\}$，则采用的是(　　)。
A．选择排序　　B．冒泡排序　　C．插入排序　　D．堆排序

(9) 下列排序算法中，(　　)可能会出现下列情况:在最后一趟开始之前，所有元素都不在最终位置上。
A．起泡排序　　B．插入排序　　C．快速排序　　D．堆排序

(10) 对有 n 个记录的线性表进行直接插入排序，在最后一趟的最坏情况下需比较(　　)次。

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

(11) 以下排序算法中，(　　)不能保证每趟至少能将一个元素放到其最终位置上。

A．快速排序　　B．希尔排序　　C．起泡排序　　D．堆排序

(12) 对数据序列$\{98,36,-9,0,47,23,1,8,10,7\}$采用希尔排序，下列(　　)是增量为 4 的排序结果。

A．$\{10,7,-9,0,47,23,1,8,98,36\}$

B．$\{-9,0,36,98,1,8,23,47,7,10\}$

C．$\{36,98,-9,0,23,47,1,8,7,10\}$

D．以上都不对

(13) 以下(　　)在数据序列基本有序时效率最好。

A．快速排序　　B．起泡排序　　C．堆排序　　D．希尔排序

(14) 下列序列中，(　　)是执行第一趟快速排序的结果。

A．$[da,ax,eb,de,bb]\ ff[ha,gc]$　　B．$[cd,eb,ax,da]\ ff[ha,gc,bb]$

C．$[gc,ax,eb,cd,bb]\ ff[da,ha]$　　D．$[ax,bb,cd,da]\ ff[eb,gc,ha]$

(15) 快速排序在(　　)情况下最不利于其长处。

A．待排序的数据量太大　　B．待排序的数据中含有多个相同值

C．待排序的数据已基本有序　　D．待排序的数据数量为奇数

(16) 对数列$\{25,84,21,47,15,27,68,35,20\}$进行排序，元素序列的变化情况如下:

①25,84,21,47,15,27,68,35,20　　②20,15,21,25,47,27,68,35,84

③15,20,21,25,35,27,47,68,84　　④15,20,21,25,27,35,47,68,84

则采用的排序方法是(　　)。

A．希尔排序　　B．简单选择排序　　C．快速排序　　D．归并排序

(17) 一组记录的关键码为$\{46,79,56,38,40,84\}$，则利用快速排序的方法，以第一个记录为准得到的一次划分结果为(　　)。

A．$\{40,38,46,56,79,84\}$　　B．$\{40,38,46,79,56,84\}$

C．$\{40,38,46,84,56,79\}$　　D．$\{84,79,56,46,40,38\}$

(18) 就平均时间而言，(　　)最佳。

A．直接插入排序　　B．起泡排序　　C．简单选择排序　　D．快速排序

(19) 快速排序的最大递归深度是(　　)，最小递归深度是(　　)。

A．$O(1)$　　B．$O(\log_2 n)$　　C．$O(n)$　　D．$O(n\log_2 n)$

(20) (　　)方法是从未排序序列中挑选元素，并将其放入已排序序列的一端。

A．归并排序　　B．插入排序　　C．快速排序　　D．选择排序

(21) 对数据序列$\{84,47,25,15,21\}$进行排序，数据序列的变化为:$\{84,47,25,15,21\}$ $\rightarrow\{15,47,25,84,21\}\rightarrow\{15,21,25,84,47\}\rightarrow\{15,21,25,47,84\}$，这采用的排序方法是(　　)。

A．简单选择排序　　B．快速排序

C．起泡排序　　D．直接插入排序

(22) 简单选择排序的比较次数和移动次数分别是(　　)。

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

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

(23) 堆的形状是一棵(　　)。

A．二叉排序树　　B．满二叉树　　C．完全二叉树　　D．判定树

(24) 对数据序列$\{15,9,7,8,20,-1,7,4\}$，用堆排序的筛选法建立的初始堆为(　　)。

A．$\{-1,7,8,9,20,15,7,8\}$　　B．$\{-1,7,15,7,4,8,20,9\}$

C．$\{-1,4,7,8,20,15,7,9\}$　　D．以上都不对

(25) 下列说法中正确的是(　　)。

A．在链表上可以实现简单选择排序，但是难以实现堆排序

B．序列$\{30,40,50,15,25,35,38,10\}$是堆

C．在最坏情况下快速排序的时间性能也好于堆排序的时间性能

D．直接选择排序是一种稳定的排序方法

(26) 下面的序列中，(　　)是堆。

A．$\{1,2,8,4,3,9,10,5\}$　　B．$\{1,5,10,6,7,8,9,2\}$

C．$\{9,8,7,6,4,8,2,1\}$　　D．$\{9,8,7,6,5,4,3,7\}$

(27) 对关键码序列$\{23,17,72,60,25,8,68,71,52\}$进行堆排序，输出两个最小关键码后的剩余堆是(　　)。

A．$\{23,72,60,25,68,71,52\}$　　B．$\{23,25,52,60,71,72,68\}$

B．$\{71,25,23,52,60,72,68\}$　　D．$\{23,25,68,52,60,72,71\}$

(28) 在堆中删除一个关键码的时间复杂度为(　　)。

A．$O(1)$　　B．$O(\log_2 n)$　　C.$O(n)$　　D．$O(n\log_2 n)$

(29) 将两个各有 n 个元素的有序表归并成一个有序表，最少的比较次数是(　　)。

A．$n$　　B．$2n-1$　　C．$2n$　　D．$n-1$

(30) 若需在$O(n\log_2 n)$时间内完成对数组的排序，且要求排序是稳定的，这可选择的排序方法是(　　)。

A．快速排序　　B．堆排序　　C．归并排序　　D．希尔排序

(31) 二路归并的趟数是(　　)。

A．$n$　　B．$\log_2 n$　　C．$n\log_2 n$　　D．$n^2$

(32) (　　)在某趟排序结束后不一定能选出一个元素放到其最终位置上。
A．选择排序　　B．起泡排序　　C．归并排序　　D．堆排序

(33) 下列排序算法中，(　　)在最坏情况下的时间复杂度不高于$O(n\log_2 n)$。
A．起泡排序　　B．归并排序　　C．希尔排序　　D．快速排序

(34) 以下排序方法中，(　　)不需要进行关键码的比较。
A．快速排序　　B．归并排序　　C．基数排序　　D．堆排序

(35) 以下排序方法中，稳定的排序方法是(　　)。
A．快速排序　　B．希尔排序 C．基数排序　　D．堆排序

(36) 已知关键码序列$\{78,19,63,30,89,84,55,69,28,83\}$采用基数排序，第一趟排序后的关键码为(　　)。
A．$\{19,28,30,55,63,69,78,83,84,89\}$　　B．$\{28,78,19,69,89,63,83,30,84,55\}$
C．$\{30,63,83,84,55,78,28,19,89,69\}$　　D．$\{30,63,83,84,55,28,78,19,69,89\}$

(37) 将 1000 个英文单词进行排序，采用(　　)排序方法时间性能最好。
A．插入排序　　B．快速排序　　C．堆排序　　D．基数排序

(38) 假设线性表中每个记录有两个数据项$K1$和$K2$，现对线性按如下规则进行排序:先按数据项$K1$由小到大排序，在$K1$值相同的情况下，$K2$值小的在前面，大的在后面则应该采用的排序方法是(　　)。
A．先按$K1$值进行直接插入排序，再按$K2$的值进行简单选择排序
B．先按$K2$值进行直接插入排序，再按$K1$的值进行简单选择排序
C．先按$K1$值进行简单选择排序，再按$K2$的值进行直接插入排序
D．先按$K2$值进行简单选择排序，再按$K1$的值进行直接插入排序

(40) 下列排序方法中，时间性能与待排序记录的初始状态无关的是(　　)。
A．插入排序和快速排序　　B．归并排序和快速排序
C．选择排序和归并排序　　D．插入排序和归并排序

(41) 设要将序列$\{Q,H,C,Y,P,A,M,S,R,D,F,X\}$中的关键码按升序排列，则(　　)是起泡排序一趟扫描的结果，(　　)是增量为 4 的希尔排序扫描一趟的结果，(　　)是二路归并排序一趟的结果，(　　)是以第一个元素为轴值的快速排序一趟扫描的结果，(　　)是堆排序初始建堆的结果。
A．$\{F,H,C,D,P,A,M,Q,R,S,Y,X\}$　　B．$\{P,A,C,S,Q,D,F,X,R,H,M,Y\}$
C．$\{A,D,C,R,F,Q,M,S,Y,P,H,X\}$　　D．$\{H,C,Q,P,A,M,S,R,D,F,X,Y\}$
E．$\{H,Q,C,Y,A,P,M,S,D,R,F,X\}$

(42) 排序趟数与数据的原始状态有关的排序方法是(　　)。
A．直接插入排序　　B．简单选择排序　　C．快速排序　　D．归并排序

## 2. 算法设计

(1) 直接插入排序算法 InsertSort。

```
void InsertSort(int r[],int n)
```

(2) 折半插入排序算法 BinSort。

```
void BinSort(int r[],int n)
```

(3) 希尔排序算法 ShellSort。

```
void ShellSort(int r[],int n)
```

(4) 冒泡排序算法 BubbleSort。

```
void BubbleSort(int r[],int n)
```

(5) 快速排序一次划分算法 Partition。

```
int Partition(int r[], int first, int end)
```

(6) 快速排序算法 QuickSort。

```
void QuickSort(int r[],int first, int end)
```

(7) 快速排序非递归算法 QuickSort。

```
void QuickSort(int r[],int n)
```

(8) 简单选择排序算法 SelectSort。

```
void SelectSort(int r[],int n)
```

(9) 筛选法调整堆算法 Sift。

```
void Sift(int r[],int k,int m)
```

(10) 堆排序算法 HeapSort。

```
void HeapSort(int r[],int n)
```

(11) 一次归并算法 Merge。

```
void Merge(int r[], int r1[], int s, int m, int n)
```

(12) 归并排序的递归算法 MergeSort。

```
void MergeSort(int r[], int r1[], int s, int t)
```