《数据结构与算法》习题

DSA Spring 2025

第一章 绪论

1）Hailstone问题（又名3n+1问题）中Hailstone(n)的计算程序是

A．对于所有的n都是无穷的

B．对于部分n是无穷的

C．不能确定是否存在n，使程序无法终止

D．对于所有的n都是有穷的

2）判断一个算法是否是一个“好算法”，最重要的一条性质是

A．正确

B．健壮

C．可读

D．效率

3）以下哪项不是图灵机的组成要件?

A．有限长的纸带

B．有限的字母表

C．有限种状态

D．读写头

4）判断正误：RAM模型与图灵机模型的区别在于图灵机的存储空间无限，而RAM的存储空间有限。

A．对

B．错

5）在大O记号的意义下，以下哪一项与$O(n^3)$相等？（m不是常数）

A. $O(3^n)$

B. $O(n^3+2000n^2+1000n)$

C. $O(n^3+m)$

D. $O(2000n^3+n^4)$

6）下列对应关系中错误的是

A. $1^2+2^2+3^2+\ldots+n^2 = O(n^3)$

B. $1+2+4+\ldots+2^n = O(2^n)$

C. $\log1+\log2+\log3+\ldots+\log n = O(n\log n)$

D. $1+1/2+1/3+\ldots+1/n = O(n\log n)$

7）判断：减而治之的思想是：将问题划分为两个平凡的子问题，分别求解子问题，来得到原问题的解。

A．对

B．错

8）用分而治之的思想来解决长度为n的数组的求和问题（n足够大），递归实例的数目会比用减而治之的方法少。

A．对

B．错

9）直接用定义以递归的方式计算fib(n)的时间复杂度是：

A. $O(n^2)$

B. $O(2^n)$

C. $\Theta(2^n)$

D. $O(n)$

10）以现在普通计算机的速度，直接用定义以递归的方式计算fib(100)需要多少时间（不考虑溢出）：

A. 一小时之内

B. 大约一天

C. 十年

D. 这辈子看不到啦

11）用动态规划计算fib(n)的时间、空间复杂度分别为：

A. $\Theta(n^2)$, $\Theta(n^2)$

B. $\Theta(n^2)$, $\Theta(n)$

C. $\Theta(n)$, $\Theta(n)$

D. $\Theta(n)$, $\Theta(1)$

12）下列说法中正确的是：

A．抽象数据类型是C++提供的一种高级的数据类型，用于实现数据结构。

B．抽象数据类型决定了数据的存储方式。

C．同一个抽象数据类型可能用多种数据结构实现。

D．数据结构即抽象数据类型。

13）在一个初始为空的向量上依次执行：insert(0, 2), insert(1, 6), put(0, 1), remove(1), insert(0, 7) 后的结果是：

A．{6, 2, 7}

B．{2, 6, 0, 7}

C．{7, 1}

D．{2, 1, 7}

14）在一个初始最大容量为10的空向量上依次执行：insert(0, 2), insert(1, 6), put(0, 1), remove(1), insert(0, 7) 后的装填因子是：

A. 10%

B. 20%

C. 30%

D. 40%

15）以下代码是向量复制代码的一个变体且语义与其相同，空格处应填入的内容为：

```
void copyFrom(const T *A, Rank lo, Rank hi)
{
    _elem = new T[_capacity = 2 * (hi - lo)];
    _size = hi - lo;
    for (int i = _size - 1; -1 < i; i--)
    {
        _elem[i] = A[ _____ ];
    }
}
```

A. --hi

B. hi--

C. ++lo

16）采用每次追加固定内存空间的扩容策略，规模为n的向量插入元素的分摊时间复杂度为：

A. $\Theta(n\log_2 n)$

B. $\Theta(n)$

C. $\Theta(\log_2 n)$

D. $\Theta(1)$

17）分别采用每次追加固定内存空间和每次内存空间翻倍两种扩容策略，规模为n的向量插入元素的分摊时间复杂度分别为：

A. $\Theta(n)$, $\Theta(1)$

B. $\Theta(n)$, $\Theta(n)$

C. $\Theta(1)$, $\Theta(1)$

D. $\Theta(n)$, $\Theta(\log_2 n)$

18）关于平均复杂度和分摊复杂度，下列说法中错误的是：

A．分摊复杂度所考量的一串操作序列一定是真实可行的

B．平均复杂度依赖于对各操作出现概率的假设，而分摊复杂度则不是如此

C．分摊复杂度得到的结果比平均复杂度低

D．加倍扩容策略中O(1)的结论是指分摊复杂度

19）向量disordered()算法的返回值是：

A．逆序数

B．相邻逆序对个数

C．表示是否有序的bool值

D．表示是否有序的int值

20）为什么有序向量唯一化算法中不需要调用remove()进行元素删除?

A. 本来就没有重复元素

B. 重复元素被直接忽略了

C. 重复元素被移到了向量末尾

D. 重复元素修改成了不重复的元素

21)对于规模为n的向量，低效版uniquify()的最坏时间复杂度为:

A. $\Theta(n^2)$

B. $\Theta(n\log_2 n)$

C. $\Theta(n)$

D. $\Theta(1)$

22）有序向量中的重复元素

A．与无序向量相同

B．大部分紧邻分布，只有极小部分散布在其它位置

C．必定全部紧邻分布

D．全部相间分布

23）对于规模为n的向量，查找失败时find()的返回值是：

A．-1

B．0

C．n

D．NULL

24）在有序向量V中插入元素e并使之保持有序，下列代码正确的是：

A. V.put(V.search(e) , e);

B. V.insert(V.search(e), e);

C. V.put(V.search(e) + 1, e);

D. V.insert(V.search(e) + 1, e);

25）在binsearch(e, lo, hi)版本A中，若V[mi] < e，则下一步的查找范围是：

A. V(mi, hi)

B. V[mi, hi]

C. V(mi, hi]

D. V[lo, hi)

26) Rank mi = (lo + hi) >> 1 等效于下列哪个表达式? (lo和hi非负)

A. Rank mi = (lo + hi) \ 2

B. Rank mi = (lo + hi) % 2

C. Rank mi = (lo + hi) / 2

D. Rank mi = (double)(lo + hi) / 2

27) 有序向量V={2, 3, 5, 7, 11, 13, 17}, 按二分查找算法版本A, V.search(16, 0, 7)需要进行多少次比较?

A. 4

B. 5

C. 6

D. 7

28) 有序向量，V1={2，3，5，7，11，13，17，19，23，29，31，37，41，43，47，53，59，61}
V2={1，2，3，4，5，6，7，8，9，10，11，12，13，14，15，16，17，18}
在V1中查找43的查找长度是x，则在V2中查找14的查找长度为：

A. x

B. x+1

C. 2 * x

D. x / 2

29) 向量fibSearch()算法与binSearch()有什么区别？

A. 二者的返回值不同

B. 前者是递归算法，后者是迭代算法

C. 二者选取轴点mi的方式不同

D. 前者的渐进时间复杂度低于后者，故前者效率更高

30) V={1, 2, 3, 4, 5, 6, 7}, 在V中用Fibonacci查找元素1, 被选取为轴点mi的元素依次是:

A. 4, 3, 2, 1

B. 4, 2, 1

C. 5,2,1

D. 5,3,2,1

31) 如果（有序）向量中元素的分布满足独立均匀分布（排序前），插值查找的平均时间复杂度为:

A. O(n)

B. O(logn)

C. O(loglogn)

D. O(1)

32）在向量V={2，3，5，7，11，13，17，19，23}中用插值查找搜索元素7，猜测的轴点mi__

33）对于规模为n的向量，二分查找版本A和B的最优时间复杂度分别为：

A. $\Theta(n^2)$, $\Theta(n)$

B. $\Theta(n\log_2 n)$, $\Theta(n)$

C. $\Theta(\log_2 n)$, $\Theta(\log_2 n)$

D. $O(1)$, $\Theta(\log_2 n)$

34）对于search()接口，我们约定当向量中存在多个目标元素时返回其中：

A. 秩最大者

B. 秩最小者

C. 秩中间者

D. 随机返回其中一个即可

35）对于二分查找版本C，当e<V[mi]不成立时下一步的查找范围是：

A. V[lo, mi)

B. V[mi, hi)

C. V[mi, hi]

D. V(mi, hi)

36）向量起泡排序中，V={7，2，3，11，17，5，19，13}，对V进行两次扫描交换后V[6] =

A．11

B．13

C．17

D．19

37）经改进的起泡排序在什么情况下会提前结束?

A．完成全部n-1趟扫描交换

B．完成的扫描交换趟数 = 实际发生元素交换的扫描交换趟数 + 1

C．完成的扫描交换趟数 = 实际发生元素交换的扫描交换趟数

D．完成（n - 1）/ 2趟扫描交换

38）试用以下算法对V={19，17，23}排序：
1．先按个位排序
2．在上一步基础上，再按十位排序
这个算法是否正确?

A．一定正确

B．一定不正确

C．若第2步用的排序算法是稳定的，则正确

D 若第1步用的排序算法是稳定的，则正确

39）对{2，5，7}和{3，11，13}进行二路归并，执行的元素比较依次是：

A．2与3、5与3、5与11、7与11

B．2与5、5与3、11与13、5与7

C．2与3、2与11、7与11、13与7

D．2与3、5与11、7与13

40）对于规模为n的向量，归并排序的最优、最坏时间复杂度分别为：

A. $\Theta(n)$, $\Theta(n\log_2 n)$

B. $\Theta(n\log_2 n)$, $\Theta(n\log_2 n)$

C. $\Theta(n\log_2 n)$, $\Theta(n^2)$

D. $\Theta(n)$, $\Theta(n^2)$

41）下列关于列表的秩的说法中不正确的是

A. 列表节点的秩与物理地址有明确的对应关系

B. 列表的秩具有很高的维护成本

C. 在列表中循秩访问的成本较高

D. 在列表中循位置访问会比循秩访问更快

42）下列关于我们定义的接口的描述中，哪一条是错误的?

A. 对于列表中的节点，我们可以通过调用 pred() 和 succ() 接口分别取其前驱和后继。

B. 对于列表接口中的 find(e) 与 search(e)，其中一个重要区别在于 find 普适于所有列表，而 search 适用于有序列表。

C. 如果一个列表的 visible list 部分长度为 n，则头、首、末、尾节点的秩分别为 -1，0，n，n + 1

D. 在构造列表时，我们需要首先构造哨兵节点。

43）若将insertAsPred()改为以下函数，其结果是：

```
template <typename T>
ListNodePosi(T) ListNode<T>::insertAsPred(T const & e) {
    ListNodePosi(T) x = new ListNode(e, pred, this);
    pred = x;    pred->succ = x;
    return x;
}
```

A．能正常插入节点

B．不能插入节点，原列表仍然保持不变

C．不能插入节点，原列表的结构被破坏

D．能否插入节点与当时列表的结构有关

44）我们可以考虑通过如下方式加快循秩访问的速度：如果r>n/2，则我们可以从尾部哨兵开始不断访问pred()，最终从后向前地找到秩为r的节点。关于这种优化，哪种说法是错误的?

A．从期望的角度看，r在[0,n)中是等概率分布的话，那么在循秩访问的过程中，对列表元素的访问次数可以节约一半。

B．原有方法访问最慢的情形大致出现在r≈n时，而改进后的方法访问最慢的情形大致出现在r≈n/2时。

C．当对于列表的访问集中在列表尾部时，这种优化策略的效果最明显。

D.通过这样的优化，我们可以使循秩访问时间复杂度优于O(n)。

45）有序列表唯一化算法的过程是：

A．只保留每个相等元素区间的第一个元素

B．每遇到一个元素，向后查找并删除与之雷同者

C．每遇到一个元素，向前查找并删除与之雷同者

D．检查(n¦2)个不同的元素对，对于每一对元素，若雷同则任意删除其一

46）能否在有序列表中用二分查找使得时间复杂度降为$O(\log_2 n)$？

A．能

B．不能，因为列表扩容的分摊复杂度不是$O(1)$

C．不能，因为列表不能高效地循秩访问

D．能，因为列表删除节点的时间复杂度为$O(1)$

47）V={11，5，7，13，2，3}，对V进行选择排序，被选为未排序子向量中最大的元素依次为：

A. 11，5，7，2，3

B. 13，7，11，2，5

C. 13，11，7，5，3

D. 2，11，13，5，7

48）为了保证selectSort()算法的稳定性，我们采取的措施是：

A. selectMax()中对于多个相等的最大元素，选取其中位置最靠后者

B. selectMax()中对于多个相等的最大元素，选取其中位置最靠前者

C. 先调用deduplicate()删除所有重复元素

D. 无论实现细节如何，该算法本来就是稳定的

49）对于规模为n的向量或列表，选择排序和冒泡排序的最坏时间复杂度为：

A. $\Theta(n\log_2 n)$, $\Theta(n^2)$

B. $\Theta(n\log_2 n)$, $\Theta(n\log_2 n)$

C. $\Theta(n^2)$, $\Theta(n^2)$

D. $\Theta(n^2)$, $\Theta(n\log_2 n)$

50）n个元素的序列所含逆序对的个数最大是：

A. n!

B. n!/2

C. (n(n-1))/2

D. n

51) insertionSort()的平均、最坏时间复杂度分别为:

A. $\Theta(n)$, $\Theta(n^2)$

B. $\Theta(n^2)$, $\Theta(n^2)$

C. $\Theta(n\log_2 n)$, $\Theta(n^2)$

D. $\Theta(n\log_2 n)$, $\Theta(n\log_2 n)$

52) 对于插入过程排序中的已排序子序列（设其长度为k）：

A. 其中的元素是整个序列中最小的k个元素

B. 其中的元素是整个序列中最大的k个元素

C. 其中的元素是原序列中位于前方的k个元素

D. 其中的元素是原序列中位于后方的k个元素

53）在插入排序的某一步后得到如下子序列V={2，7，13，5，3}，此时已排序部分有3个元素。经过又一轮迭代后的结果是：

A. {2，3，7，13，5}

B. {2，7，13，3，5}

C. {2，5，7，13，3}

D. {3，2，7，13，5}

54）下列关于向量和列表的说法错误的是：

A. 向量通常在内存中占据连续的空间，列表则通常不是如此

B. 在有序向量中查找渐进地比在有序列表中查找快

C. 向量归并排序的时间复杂度是$O(n\log_2(n))$，而列表为$\Omega(n^2)$

D. 列表删除单个节点渐进地比向量删除单个元素快

55）为了在列表中插入一个新节点node作为p的直接前驱，有四个相关的语句

①p->pred->succ = node

②node->pred = p->pred

③node->succ = p

④ p->pred = node

上述语句执行顺序正确的是：

A. ③②④①

B. ③②①④

C. ①④③②

D. ④②③①

56）在有序列表中查找一个元素的时间复杂度是：

A. $\Omega(n\log_2(n))$

B. $\Omega(n)$

C. $O(\log_2(n))$

D. $O(1)$

57）对列表{11，5，7，13，2，3}进行选择排序，每一次selectMax()被选为未排序子列表中最大者的元素依次为：

A. 11，5，7，2，3

B. 13，7，11，2，5

C. 13，11，7，5，3

D. 2，11，13，5，7

58）selectionSort()算法的哪种实现是稳定的：

A. 每一趟将最小元素移到前方，对于多个相等的最小元素，选取其中位置最靠前者。

B. 每一趟将最大元素移到后方，对于多个相等的最大元素，选取其中位置最靠前者

C. 每一趟将最小元素移到前方，对于多个相等的最小元素，选取其中位置最靠后者。

D. 以上实现皆稳定。

59）对于插入排序过程中的已排序子序列（设其长度为k）：

A. 其中的元素是整个序列中最小的k个元素

B. 其中的元素是整个序列中最大的k个元素

C. 其中的元素是原序列中位于前方的k个元素

D. 其中的元素是原序列中位于后方的k个元素

60）插入排序中的某一次插入后得到序列{2, 7, 13, 5, 3, 19, 17}，此时已排序部分有3个元素。又经过2趟迭代后的结果是：

A. {2, 3, 5, 7, 13, 17, 19}

B. {2, 3, 5, 7, 13, 19, 17}

C. {2, 5, 7, 3, 13, 19, 17}

D. {2, 3, 5, 13, 7, 17, 19}

61）一个序列的逆序数T定义为该序列中的逆序对总数，规模为n的列表中插入排序进行的元素比较总次数为：

A. $O(n+ T\log_2(T))$

B. $O(n+T)$

C. $O(n^2 + \log_2(T))$

D. $O(T)$

62）长度为n的列表，被等分为n/k段，每段长度为k，不同段之间的元素不存在逆序。对该列表进行插入排序的最坏时间复杂度为：

A. $O(n^2)$

B. $O(nk)$

C. $O(n^2/k)$

D. $O(n^2k)$

63）栈S初始为空，进行以下操作后从栈顶到栈底的元素依次为：

```
S.push(5);
S.push(4);
S.pop();
S.push(2);
S.pop();
S.pop();
S.push(1)
```

A. 5, 4, 2, 1

B. 1, 2, 4, 5

C. 1

D. 5, 4

64）当扫描到一个左括号时：

A. 出栈

B. 进栈

C. 跳过该字符

D. 算法结束

65）{3,1,2,4}是否是{1,2,3,4}的栈混洗?

A．是

B．不是

66）长度为4的序列共有多少个不同的栈混洗?

67）利用栈结构进行中缀表达式求值，什么时候进行实际的运算?

A．每遇到一个新的操作数

B．每遇到一个新的操作符

C．当前的操作符比栈顶的操作符优先级高

D．当前的操作符比栈顶的操作符优先级低

68）利用栈结构进行逆波兰表达式的求值算法中，什么时候进行一次实际的运算?

A．每遇到一个新的操作数

B．每遇到一个新的操作符

C．当前操作符优先级高于栈顶

D．当前操作符优先级低于栈顶

69）思考一下如何将逆波兰表达式还原为中缀表达式呢？试将下列逆波兰表达式还原为中缀表达式：

1 2 + 3 4 ^ *

A. (1-2)*3^4

B. (1+2)*3^4

C. (1+2)^3*4

D. (1+3)*2^4

70）栈初始为空，依次经过以下操作：

```
push(5);
push(8);
pop();
push(5);
top();
push(1);
push(3);
pop();
pop();
push(2);
```

此时从栈顶到栈底依次为：

A. 2, 5, 5

B. 2, 3, 1

C. 5, 5, 2

D. 1, 3, 2

71）阅读下面函数(其中1≤x，y≤16)，指出其功能：

```
char digits[] = { '0', '1', '2', '3', '4', '5', '6', '7'
    , '8', '9', 'a', 'b', 'c', 'd', 'e', 'f'}; //全局变量
void convert(int y, int x){ //x、y都是一定范围内的整数
    if(x != 0){
        convert(y, x / y);
        printf("%c", digits[x % y]);
    }
}
```

A. 打印十进制整数x的y进制表示

B. 打印x进制整数100的y进制表示

C. 打印十进制整数y的x进制表示

D. 打印y进制整数100的x进制表示

72）对序列{2，3，5，7，11}进行栈混洗得到{3，5，2，11，7}的过程中用于中转的栈S进行的操作是：

A. push，pop，pop，push，push，pop，push，push，pop，pop

B. push，push，pop，pop，pop，pop，push，push，push，pop

C. push，push，pop，push，pop，pop，push，pop，pop，pop

D. push，push，pop，push，pop，pop，push，push pop，pop

73) {1,2,3... i...j...k...n}:下列哪一个序列一定不是{1,2,3… i…j…k…n}的栈混洗:

A. {… i…j…k…}

B. {… k…j…i…}

C. {… k…i…j…}

D. {… j…k…i…}

74) 以下几个量中相等的是:
① 不同的n位二进制数个数
② 对小括号所能构成的合法括号匹配个数
③ {1, 2 .. n}的不同栈混洗个数
④ 含n个运算符的中缀表达式求值过程中运算符栈push操作的次数

A. ①②

B. ②③

C. ③④

D. ②④

75）在中缀表达式求值中，某时刻运算数栈从栈顶到栈底依次为：
6，2，1
运算符栈从栈顶到栈底依次为：
x，+，(
剩下的待处理表达式为：
)/(4 × 5 - 7)

在接下来的过程中运算符的入栈顺序以及最终的计算结果分别为：
A. /(x-最终结果为8

B. /(x-) 最终结果为8

C. /(x-最终结果为1

D. /(x-)最终结果为1

76）(1+2×3!)/(4×5-7)的逆波兰表达式为（表达式中的整数都是一位数）

77）树是：

A. 有向连通图

B. 无环平面图

C. 连通无环图

D. 有向无环图

78）在一棵树中，顶点p是顶点v的父亲，则它们的高度的关系是

A. height(v) < height(p)

B. height(v) = height(p) - 1

C. height(v) = height(p) + 1

D. height(p) < height(v)

79）用父节点+孩子节点的方法存储n个节点的树，需要的空间是：

A. O(1)

B. O(n)

C. O(nlogn)

D. $O(n^2)$

80）一棵高度为h，节点数为n的真二叉树的特点是：

A. $h = O(\log_2 n)$

B. 真的是二叉树，而不会是其他种类的树

C. 不存在只有一个父亲的节点

D. 不存在只有一个孩子的节点

81）在长子-兄弟表示法中，树中某节点的长子相当于二叉树中的：

A．幼子

B．左子

C．右子

D.长子

82）设二叉树有n个节点，高度为h.在其中插入一个新的节点，高度发生改变的节点个数为：

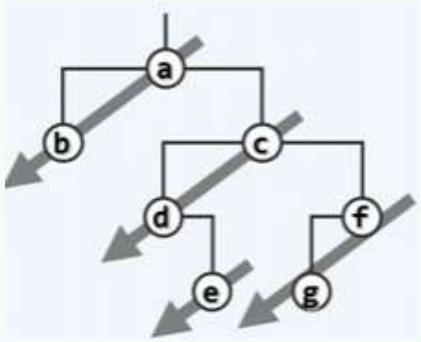
A．O(1)

B．O(n)

C．O(h)

D．$O(h\log_2(n))$

83）对以下二叉树进行先序遍历：



刚访问完节点d时（迭代实现2）栈中的元素从栈顶到栈底依次为：

A. e

B. g,f

C. f,g

D. f

84）二叉树先序遍历的顺序是：

A. 先自下而上访问左侧链上的节点，再自下而上访问它们的右子树

B. 先自上而下访问左侧链上的节点，再自上而下访问它们的右子树

C. 先自上而下访问左侧链上的节点，再自下而上访问它们的右子树

D. 先自下而上访问左侧链上的节点，再自上而下访问它们的右子树

85）中序遍历中第一个被访问的节点是：

A．最左的节点

B．最右的节点

C．根节点

D．左侧分枝的叶节点

86）层次遍历的次序是：

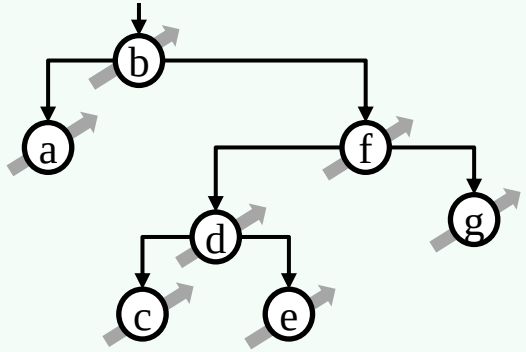
A．先根再左子最后右子

B．先左子再根最后右子

C．自上而下访问各个深度的节点，同样深度的节点中自左向右

D．自下而上访问各个深度的节点，同样深度的节点中自左向右

87）对以下二叉树进行中序遍历：



节点c刚被访问完毕时栈中的元素从栈顶到栈底为：

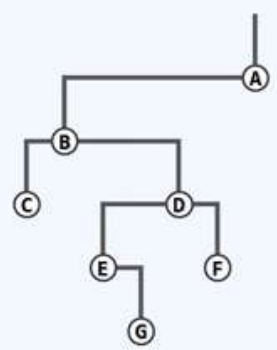
A. d, c, f

B. d, f

C. f, g, d, e

D. d

88）对以下二叉树进行层次遍历：



节点F正欲出队时队列中的元素从队头到队尾为：

A. F

B. F,G

C. E,F

D. E,F,G

89）后序遍历序列中最后一个节点是：

A．根节点

B．最左边的节点

C．最右边的节点

D．深度最大的节点

90）下列关于树的命题中错误的是：

A．顶点数为n的树的边数为n-1。

B．树中任意两顶点之间存在唯一路径。

C．在树中添加任一条边都会破坏树的结构。

D．在树中删除任一条边得到的还是树。

91）从n个节点的二叉树的叶节点u逐个节点地上溯到根节点的过程中，以下说法中错误的是：

A．经过的节点都是u的祖先。

B．最坏时间复杂度为O(n)

C．经过的路径是唯一确定的

D．每上溯一层，当前节点的深度减小1，而高度增加1。

92）对二叉树进行中序遍历，节点v在中序遍历下的后继为（假设v的后继存在）：

A．其右子树中第一个被访问的节点

B．其左子树中第一个被访问的节点

C．其右子树中第一个被访问的节点或v的某个祖先

D．其右子树中第一个被访问的节点或其左子树中的某个节点

93）与先序、中序遍历类似，以左子->右子->根节点的顺序来访问二叉树称为后序遍历。后序遍历中第一个被访问的节点是：

A．左侧链中最深的节点

B．根节点

C．右侧链中最深的节点

D．以上皆不是

94）对二叉树进行先序遍历，u和v是左侧链上两个节点，且u是v的祖先，x、y分别是u和v的右子，试问这四个节点被访问的顺序是：

A．y,v,x,u

B．x,y,v,u

C．u,v,y,x

D．unconfirmed无法确定

95）关于二叉树遍历序列之间关系的说法错误的是：

A．已知先序遍历序列和中序遍历序列可以确定后序遍历序列

B．已知中序遍历序列和后序遍历序列可以确定先序遍历序列

C．已知中序遍历序列和后序遍历序列可以确定层次遍历序列

D．已知先序遍历序列和后序遍历序列可以确定中序遍历序列

96）借助队列对二叉树进行层次遍历时，任意时刻队列中的节点满足：

A．均位于从根节点到当前节点的路径上

B．均是当前节点的后代

C．高度相差不超过1

D．深度相差不超过1

97）二叉搜索树之区别于普通的二叉树在于：

A．任意节点均不大于其右子树中的节点，不小于其左子树中的节点

B．任意节点均不大于其右孩子，不小于其左孩子

C．除了根节点外所有节点均不大于其父亲

D．关键码可以比较

98）二叉搜索树的何种遍历序列是递增的？

A．先序

B．中序

C．后序

D．层次

99）在含n个节点的BST中进行查找的最坏时间复杂度为：

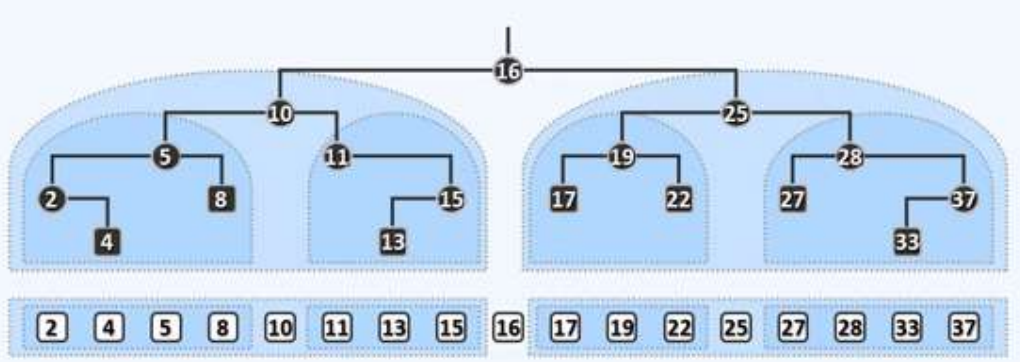
A. O(1)

B. $O(\log_2(n))$

C. O(n)

D. $O(n\log_2(n))$

100）在以下二叉查找树中查找关键码13，经过的节点依次为：



A. 16, 11, 13

B. 16, 10, 5, 8, 13

C. 16, 10, 11, 15, 13

D. 以上二叉树根本不是二叉查找树

101）对BST进行插入操作，对待插入的目标元素e进行查找后，若查找失败，_hot指向的节点为：

A．待插入的节点

B．被插入后的父亲

C．被插入后的左孩子

D．根节点

102）当欲删除的节点v在BST中的度为2时，实际被删除的节点为：

A．v在中序遍历下的直接前驱

B．v在先序遍历下的直接后继

C．v的右子树中左侧分支的最后一个节点

D．v的父亲

103）两个等价的平衡二叉搜索树有相同的：

A. 先序遍历序列

B. 中序遍历序列

C. 后序遍历序列

D. 层次遍历序列

104）在AVL树中刚插入一个节点后失衡节点个数最多为

A. O(1)

B. O(loglogn)

C. O(logn)

D. O(n)

105）在AVL树中刚删除一个节点后失衡节点个数最多为

A. O(1)

B. O(loglogn)

C. O(logn)

D. O(n)

106）AVL树中插入节点引发失衡，经旋转调整后重新平衡，此时包含节点g,p,v的子树高度

A. 减小1

B. 不变

C. 增加1

D. 有可能不变也有可能增加1

107）经过3+4重构后的AVL树_____不变。

A．先序遍历序列

B．中序遍历序列

C．后序遍历序列

D．层次遍历序列

108）AVL树中删除节点引发失衡，经旋转调整后重新平衡，此时包含节点g,p,v的子树高度

A．减小1

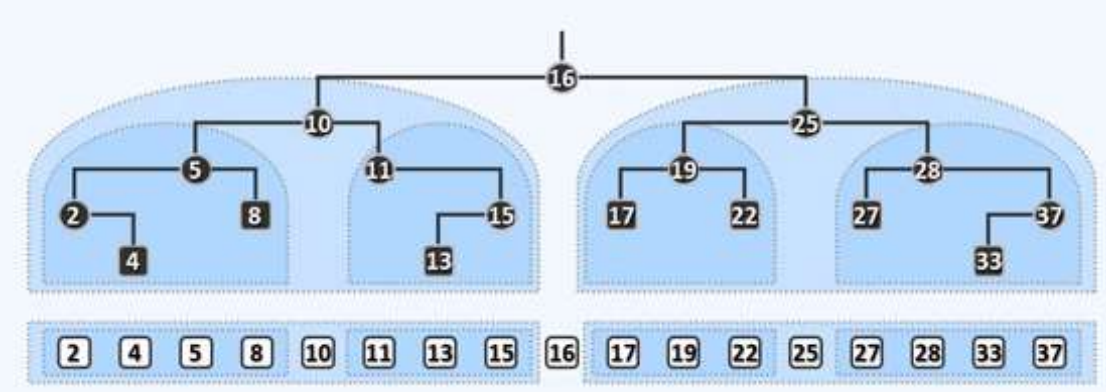
B．不变

C．增加1

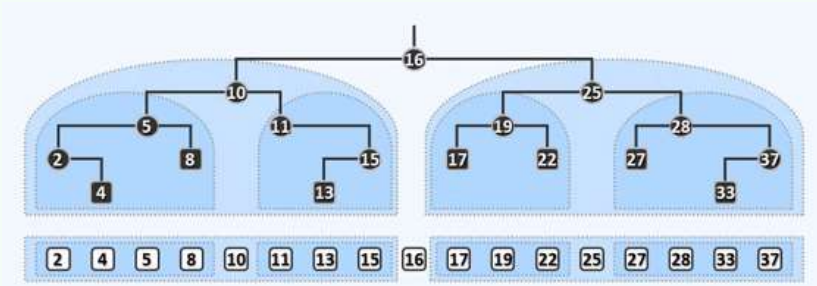
D．有可能不变也有可能减小1

109）包含节点{1,2,3,4}的不同二叉搜索树有多少棵?

110）在以下二叉搜索树中查找元素14，第3个和14发生比较的元素为：



111）欲在以下二叉搜索树中删除节点16，可行的方案是：



A．将16摘除后令25为10的右子，从而10是新的根节点

B．以16为轴进行一次zig操作使之不再是根节点，再直接摘除16

C．将节点16和节点33的关键码互换，再摘除新的节点16

D．将节点16和节点15的关键码互换，摘除新的节点16并令13为11的右子

112）二叉搜索树的高度h和节点个数n满足关系

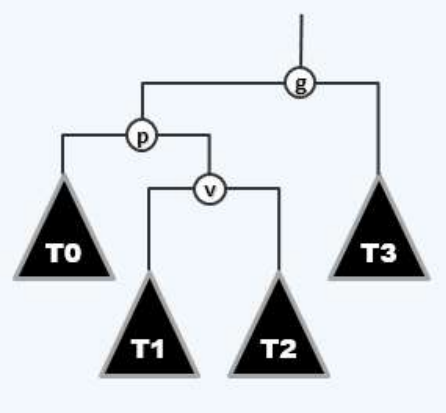
A．h=O(1)

B．h=O(log(n))

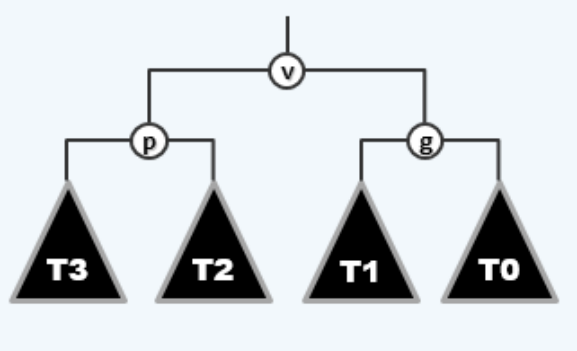
C．h=O(n)

D．h=O(nlog(n))

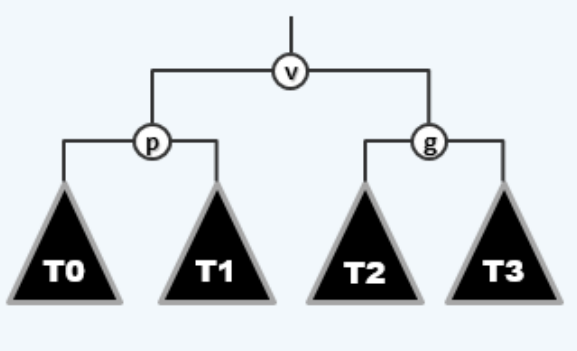
113）对以下子树进行3+4重构，得到的子树为：



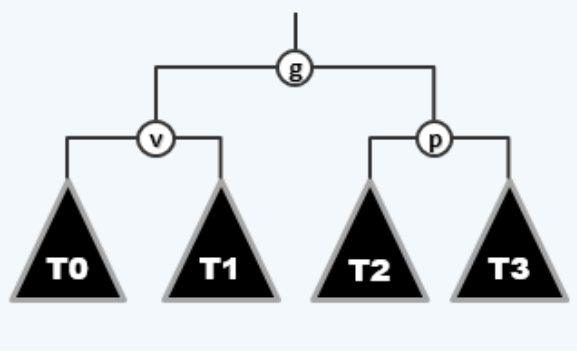
A.



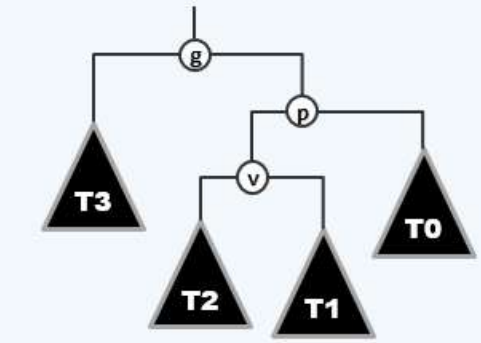
B.



C.



D.



114）伸展树利用了实际问题中数据访问的何种特点：

A．大量性

B．局部性

C．整体性

D.多样性

115）伸展树每次访问过某节点后都会把该节点：

A．删除

B．上移一层

C．移动到根节点

D．再次访问该节点

116）Tarjan提出的伸展算法每几层一起伸展?

A. 1

B. 2

C. 3

D. 4

117）在一棵退化成单链的伸展树中访问其最深的节点，经过伸展后树高大约为原先的：

A. 三分之一

B. 一半

C. 不变

D. 两倍

118）按照Tarjan的伸展算法，单次伸展操作的分摊复杂度为：

A. O(1)

B. O(logn)

C. O(n)

D. O(nlogn)

119）双层伸展策略优于逐层伸展策略的关键在于：

A. zig-zig/zag-zag

B. zig-zag/zag-zig

C. zig-zig/zig-zag

D. 以上均是

120）伸展树采用双层伸展策略，即可避免最坏情况的发生（）

121）所访问的节点及其父亲都是右孩子，则双层伸展要执行的操作是：

A. zig-zag

B. zag-zig

C. zig-zig

D. zag-zag

122）规模为n的伸展树中若所访问的节点只有k个，经过足够长时间的访问序列后，访问的分摊复杂度为：

A. O(lgk)

B. O(klgn)

C. O(nlgk)

D. O(lgn)

123）在任意一颗伸展树中，按节点值的大小以升序依次访问完所有节点，最后树变为一条单链，且仅含左孩子。

124) B-trees are 树结构上的特点是:

A. 矮胖，每个节点至多两个孩子

B. 矮胖，每个节点可有多于两个孩子

C. 瘦高，每个节点至多两个孩子

D. 瘦高，每个节点可有多于两个孩子

125) B树的层数少有助于:

A. 减少I/O次数

B. 减少每次I/O的时间

C. 降低查找的渐进时间复杂度

D. 节省存储空间

126）4阶B树中每个节点的分支数为：

A. 1~4

B. 2~5

C. 2~4

D. 3~5

127）在存储了n个元素的4阶B树中查找，单个节点进行一次查找的时间复杂度为：

A. O(1)

B. O(lgn)

C. O(n)

D. O(nlgn)

128）B树查找算法若最终失败，返回值为：

A. None

B. NULL

C. 指向最后一个所查找节点的指针

D. 指向根节点的指针

129）若B树的阶m=128，则它的高度大致是对应的BBST的：

A. 1/5

B. 1/6

C. 1/7

D. 1/8

130）B树的上溢是指：

A．删除关键码后违反了B树的性质

B．插入新的关键码后违反了B树的性质

C．分裂后违反了B树的性质

D．B树的高度过高

131）B树高度的增加一定伴随着：

A．每个节点所存放的关键码数量增加

B．每个节点所存放的关键码数量减少

C．分裂到根

D．分裂到叶

132）B树的下溢发生于：

A．B树高度减少

B．插入关键码后违反了B树的性质

C．删除关键码后违反了B树的性质

D．在叶节点插入关键码

133）B树高度的减少只会发生于

A．根节点的两个孩子合并

B．根节点被删除

C．根节点发生旋转

D．根节点有多个关键码

134）我们分别学习了B-树的插入和删除，请判断正误：
如果在不发生上溢和下溢的情况下，那么单次删除和插入操作的时间花费大致相同。

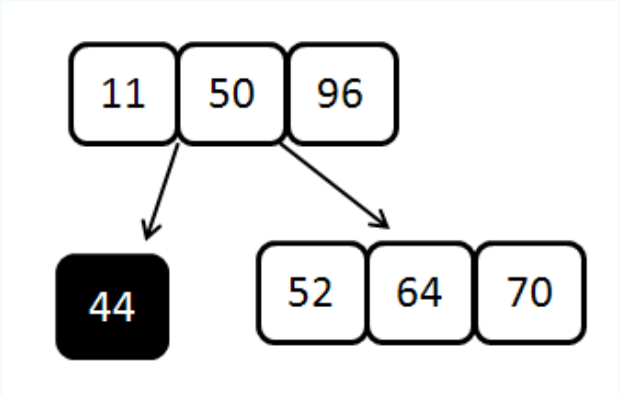
135）伸展树虽然单次操作的最坏时间复杂度比较大，但是可以利用存储器的层次结构降低I/O的次数

136）伸展树相较于AVL树的缺点是它实现起来较为复杂

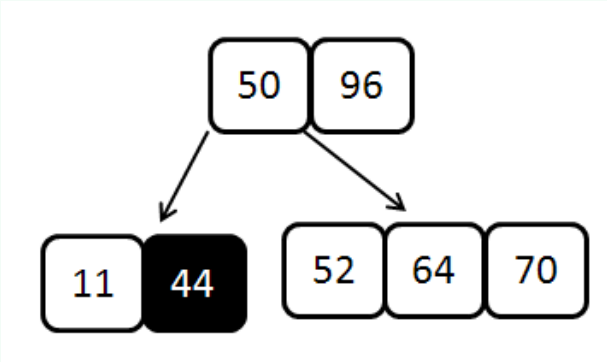
137）伸展树插入操作的分摊复杂度比AVL树大

138）伸展树单次查找操作的最坏时间复杂度比AVL树大

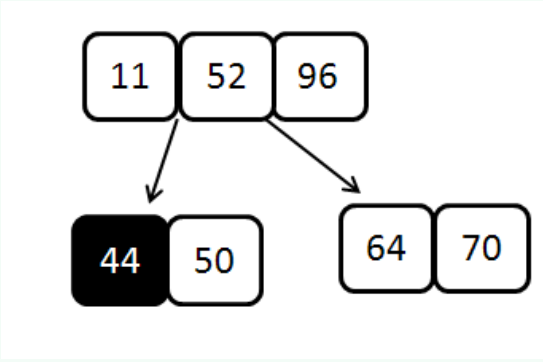
139）下图是(3,6)-树中刚删除某节点后的情形，可以看出发生了下溢，调整后的结果为：



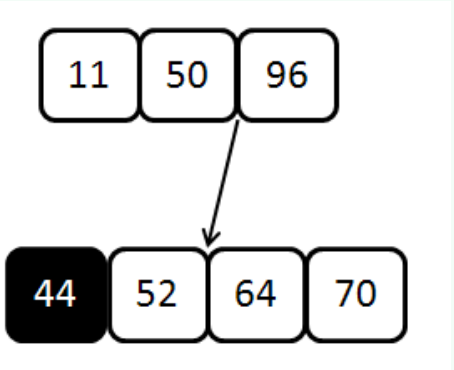
A.



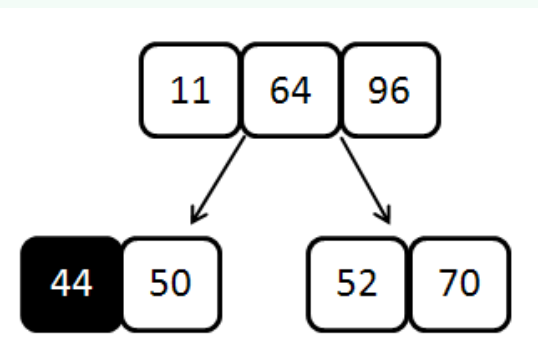
B.



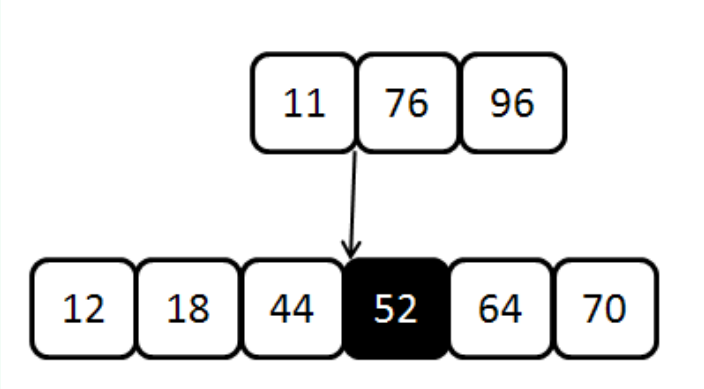
C.



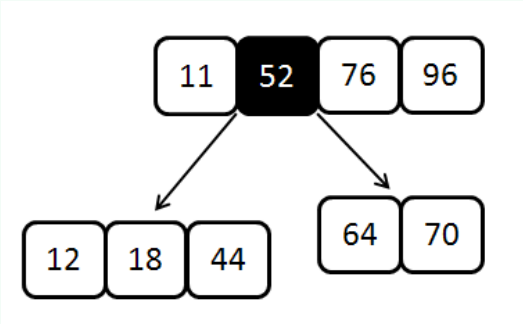
D.



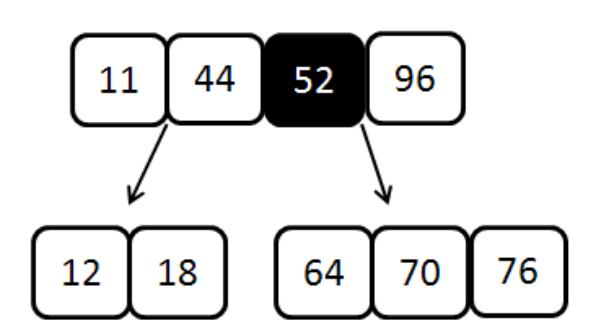
140) 下图是(3,6)-树中刚插入节点52后的情形，可以看出发生了上溢，分裂后的结果为：



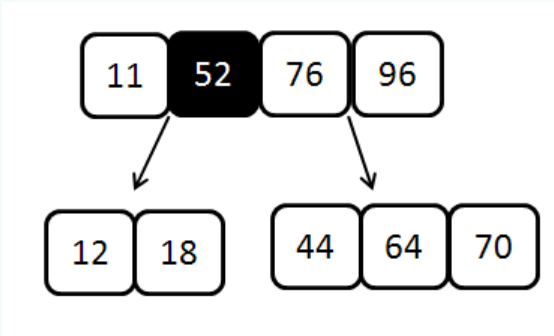
A.



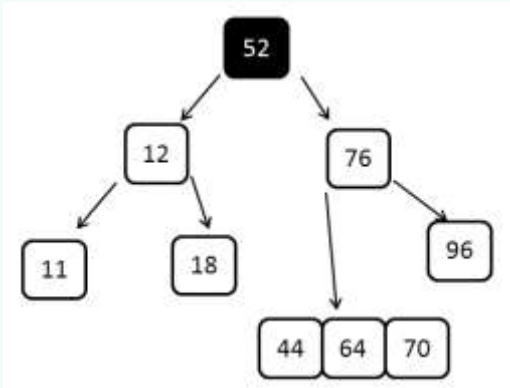
B.



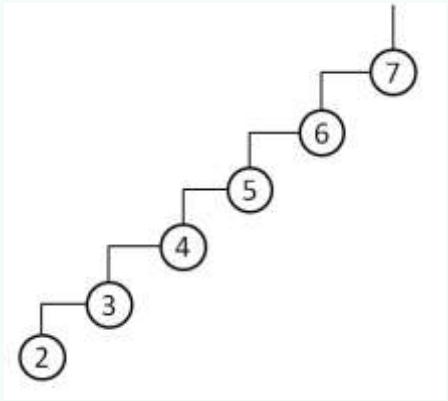
C.



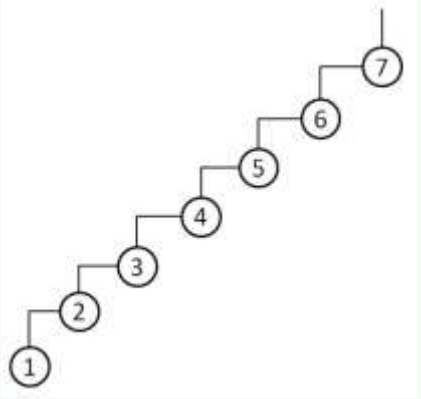
D.



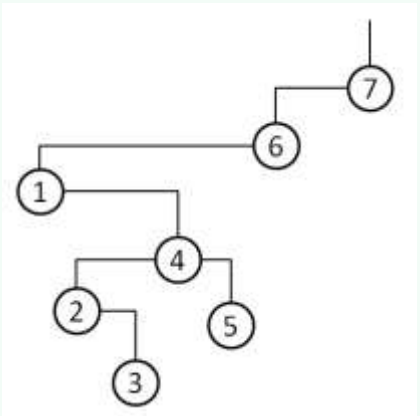
141）在以下伸展树中插入节点1并经过双层伸展后的结果是：



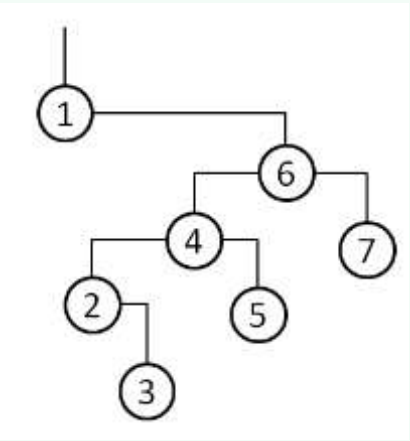
A.



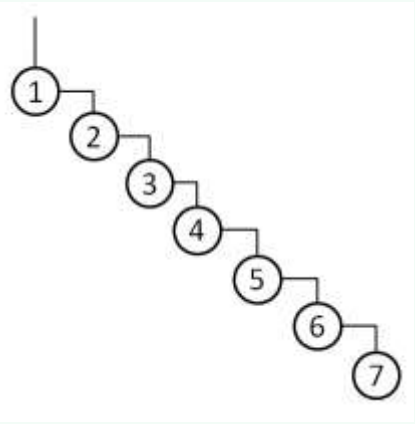
B.



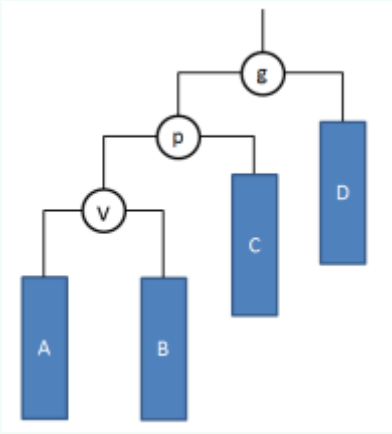
C.



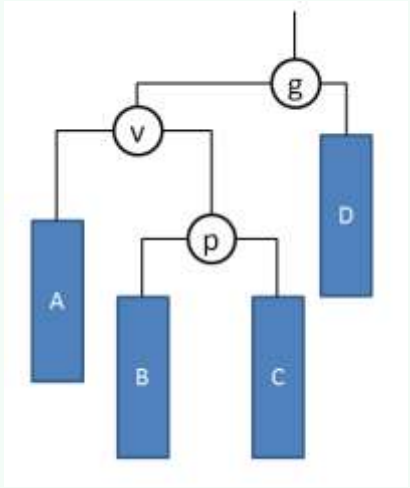
D.



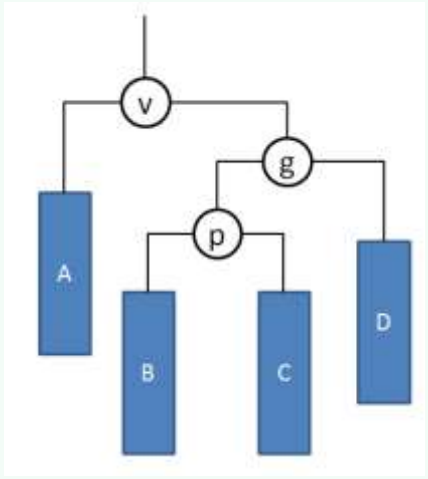
142）下图是伸展树，其中节点v刚被访问过，双层伸展后的结果是：



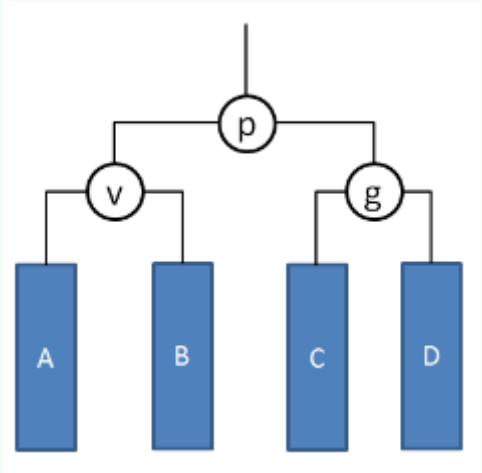
A.



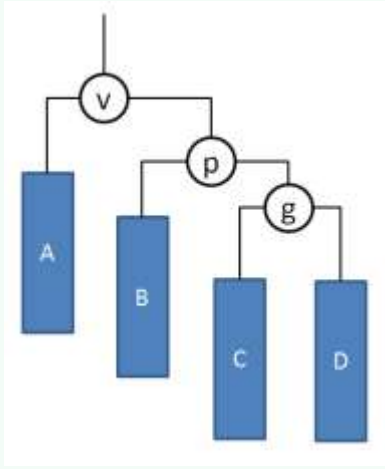
B.



C.



D.



143）散列函数是 h(x) = x % 20，则关键码25，85，15，20中会发生冲突的是：

A．25和85

B．25和15

C．15和85

D．20和15

144）在散列中，关键码个数超过实际使用的空间时，有没有可能不发生冲突：

A．可能
B．不可能

145）S为所有可能词条的空间，A为所有可用地址的空间（|A| < |S|），h是散列函数，则：

A. 从S映射到A，一定是满射

B. 从S映射到A，不可能是单射

C. 从A映射到S，不可能是满射

D. 从A映射到S，一定是单射

146）考虑key的集合S = {0, 8, 16, 24, 32, 40, 48, 56, 64}
用除余法构造的散列函数
h1(key) = key % 12
h2(key) = key % 11
h1将S映射到的值域有几个元素?

____

h2将S映射到的值域有几个元素?

____

147）关于排解冲突的方法，以下说法正确的是：

A. 用独立链法排解冲突，所有词条的实际存放位置均在桶数组内部

B. 用开放定址排解冲突，词条的实际存放位置不一定是对应的散列函数值

C. 用开放定址排解冲突，词条被存放在列表中

D. 只要散列函数设计得当，不一定需要排解冲突的策略

148）规模为11的桶数组当前状态为 A = { *, *, *, *, *, 0, 15, 26, *, 5, 9}，其中*表示空桶
散列函数为h(key) = (3 * key + 5) % 11
用开放定址+线性试探排解冲突
插入词条4，它的实际存放位置是

A. A[4]

B. A[6]

C. A[7]

D. A[8]

149）规模为11的桶数组当前状态为 A = { *, *, *, *, *, 0, 15, 26, *, 5, 9}，其中*表示空桶
散列函数为h(key) = (3 * key + 5) % 11
用开放定址+平方试探排解冲突
插入词条4，它的实际存放位置是

A. A[4]

B. A[6]

C. A[7]

D. A[8]

150）散列表的规模是素数，用开放定址+平方试探法排解冲突，若要保证新的词条能够顺利插入，散列表的装填因子不能超过（请填十进制小数）

151）任何两个顶点间都有一条（无向）边的图称为完全图，包含n个顶点的完全图用K_n表示。下列哪个图一定不是平面图?

A. K_2

B. K_3

C. K_4

D. K_5

152）在包含n个顶点的用邻接矩阵实现的图中，顶点v有m个邻居，遍历所有m个邻居的时间复杂度为:

A. O(1)

B. O(m)

C. O(n)

D. O(mn)

153）图G包含n个顶点(n>0)，用邻接矩阵实现。在其中加入一个新的顶点后邻接矩阵增加了多少项?

A. n+1

B. 2n

C. 2n-1

D. 2n+1

154）图的广度优先搜索访问各顶点的模式类似于二叉树的:

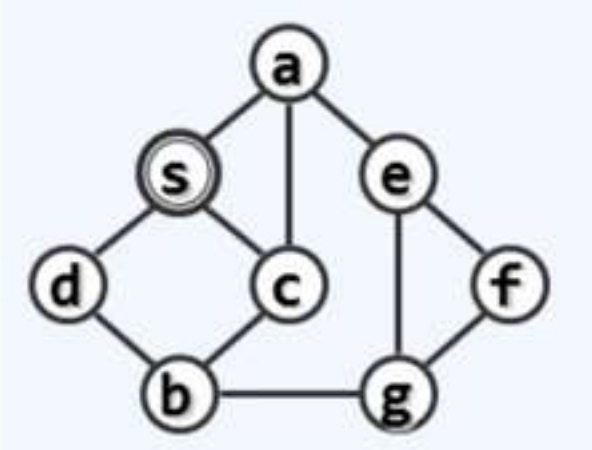
A. 先序遍历

B. 中序遍历

C. 后序遍历

D. 层次遍历

155）以顶点s为起点对以上无向图进行BFS，同一顶点的邻居之间以a~z为顺序，顶点c刚出队时队列中顶点从队头到队尾为：



A. d, e

B. e, d

C. d, a

D. a, d

156）对于用邻接表实现的包含n个顶点e条边的图，BFS的时间复杂度为：

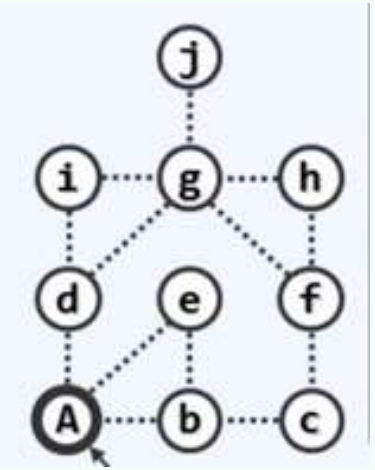
A. O(n)

B. $O(n^2)$

C. O(n+e)

D. $O(n^2+e)$

157）以A为起点对以下无向图进行DFS，同一顶点的邻居以a~z为序，各顶点被访问的顺序为：



A. a, e, b, c, f, g, h, j, i, d

B. a, b, c, f, g, d, i, h, j, e

C. a, b, e, c, f, h, g, i, d, j

D. a, e, b, c, f, h, g, i, d, j

158）u和v为图中两个顶点，对图进行DFS后，dTime(u) < dTime(v) < fTime(v) < fTime(u)，则u和v在DFS森林中的关系是：

A. 属于不同的连通分量

B. u 为 v 的祖先

C. v 为 u 的祖先

D. 互为兄弟

159）对同一个无向图分别运行广度优先算法和深度优先算法，得到的树边数量：

A. 广度优先的树边更

B. 深度优先的树边更多

C. 两种算法得到的树边一样多

D. 数量关系不确定

160）对图进行DFS，以下哪种情况意味着该图包含环路

A. 有TREE边

B. 有BACKWARD边

C. 有FORWARD边

D. 有CROSS边

161）在含20个顶点的简单无向图中，边的数量最多为：_____
此时度最小的顶点的度为：_____

162）某宴会一共有7个人参加，与会者之间进行了亲切的握手。已知他们中的每个人进行握手的次数分别为：
3，1，2，2，3，1，2
请问宴会上总共发生了多少次握手?

_____

163)对于包含n个顶点e条边的简单无向图，以下关于它的邻接矩阵A的说法中错误的是：

A. A有n行e列，其中元素取值于{0，1}

B. A的第k行中1的个数等于顶点k的度

C. $A = A^T$

D. A中位于第u行v列的元素为1当且仅当顶点u和顶点v邻接

164）用邻接矩阵实现含n个顶点e条边的图，其空间复杂度为：

A. O(1)

B. O(n)

C. O(nlogn)

D. $O(n^2)$

165）G是简单无向图，A为G的邻接矩阵，M为G的关联矩阵，D是对角线上第i个元素为顶点i的度的对角矩阵，它们的关系是：

A. A=M

B. $A+D=MM^T$

C. $A+D=M^TM$

D. 没有直接关系

166）G是有向无环图，(u，v)是G中的一条由u指向v的边。对G进行DFS的结果是：

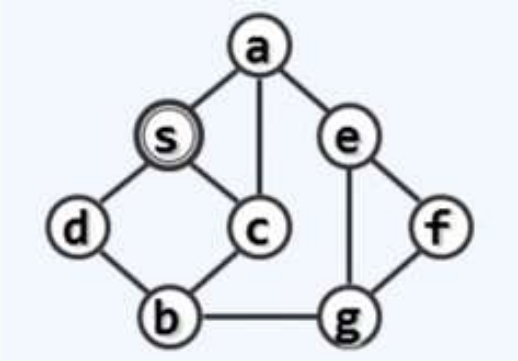
A. dTime(u) > dTime(v)

B. dTime(u) < dTime(v)

C. fTime(u) > fTime(v)

D. fTime(u) < fTime(v)

167）从s开始，对以上无向图进行BFS，同一顶点的邻居之间以a~z为序，求顶点的dTime：



s的dTime = 0

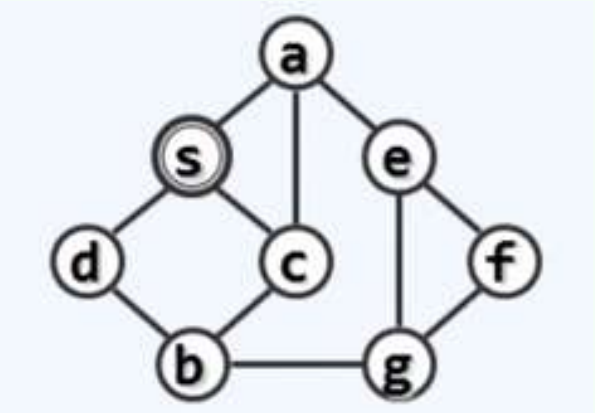
a的dTime =____

b的dTime =____

e的dTime =____

f的dTime =____

168）从s开始，对以上无向图进行DFS，同一顶点的邻居之间以a~z为序，求各顶点的dTime和fTime



s的dTime =0
s的fTime =15

c的dTime =____

c的fTime =____

g的dTime =____

g的fTime =____

169）如果把朋友圈视为一无向图，那么即使A君看不到你给B点的赞，你们仍可能属于同一双连通分量。

170）对于同一无向图，起始于顶点s的DFS尽管可能得到结构不同的DFS树，但s在树中的度数必然固定。

171）已知有向图G=(V, E)，其中V = {v1, v2, v3, v4, v5, v6}, E = {<v1,v2>, <v1,v4>, <v2,v6>, <v3,v1>, <v3,v4>, <v4,v5>, <v5,v2>, <v5,v6>}。G的拓扑序列是:

A. v1, v3, v4, v5, v2, v6

B. v3, v4, v1, v5, v2, v6

C. v1, v4, v3, v5, v2, v6

D. v3, v1, v4, v5, v2, v6

172）在拓扑排序算法中用堆栈和用队列产生的结果会不同吗?

A. 有可能会不同

B. 肯定是相同的

C. 以上全不对

D. 是的肯定不同

173）若将n个顶点e条弧的有向图采用邻接表存储，则拓扑排序算法的时间复杂度是：

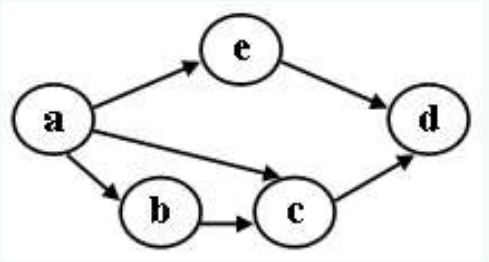
A. O(n×e)

B. O(n2)

C. O(n+e)

D. O(n)

174）对下图进行拓扑排序，可以得到不同的拓扑序列的个数是：



A. 1

B. 2

C. 3

D. 4

175）Prim 算法是通过每步添加一条边及其相连的顶点到一棵树，从而逐步生成最小生成树。

176）数据结构中Dijkstra算法用来解决哪个问题?

A. 字符串匹配

B. 最短路径

C. 拓扑排序

D. 关键路径

177）若从无向图的任意一个顶点出发进行一次深度优先搜索可以访问图中所有的顶点，则该图一定是（ ）图。

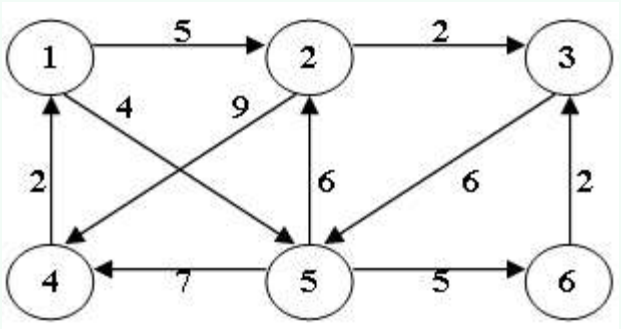
A. 非连通

B. 连通

C. 强连通

D. 有向

178)使用Dijkstra算法求下图中从顶点1到其他各顶点的最短路径，依次得到的各最短路径的目标顶点是：



A. 5, 2, 6, 3, 4

B. 5, 2, 3, 6, 4

C. 5, 2, 4, 3, 6

D. 5, 2, 3, 4, 6

179）Dijkstra算法是按路径长度递增的顺序依次产生从某一固定源点到其他各顶点之间的最短路径。

180）在对有向无环图执行拓扑排序算法之后，入度数组中所有元素的值均为0。

181）完全二叉堆中父节点的优先级

A. 不小于它的孩子

B. 不等于它的孩子

C. 和它的孩子没有必然的大小关系

D. 等于它的孩子

182）在完全二叉堆中插入元素的方法是

A. 插入到底层，上滤

B. 插入到根节点，下滤

C. 直接插入到底层

D. 直接插入到根节点

183）规模为n的完全二叉堆中插入元素的时间复杂度为：

A. O(nlogn)

B. O(n)

C. O(logn)

D. O(1)

184)完全二叉堆中要删除一个元素时，这个元素的位置是：

A. 根节点

B. 叶子节点

C. 可以是任意节点

D. 根节点或叶子节点

185）规模为n的完全二叉堆中删除元素的时间复杂度为：

A. O(nlogn)

B. O(n)

C. O(logn)

D. O(1)

186）使用自上而下的上滤建立规模为n的完全二叉堆，最坏时间复杂度为：

A. O(nlogn)

B. O(n)

C. O(logn)

D. O(1)

187）Floyd建堆算法建立规模为n的完全二叉堆的时间复杂度为：

A. O(nlogn)

B. O(n)

C. O(logn)

D. O(1)

188）如何用堆来实现排序：

A. 建堆后不断调用delMax

B. 建堆后不断调用getMax()

C. 建堆后不断调用insert()

D. 建堆

189）栈作为优先级队列的一种特殊情况，其中元素的优先级：

A. 先入栈者优先级低

B. 先入栈者优先级高

C. 所有元素优先级相同

D. 元素优先级之间没有确定的关系

190）完全二叉树从整体上看的形态是：

A. 完整的三角形

B. 缺右角的三角形

C. 缺左角的三角形

D. 底部呈锯齿状的三角形

191）完全二叉堆在物理上是向量，其所存储的元素次序是：

A. 完全二叉树的先序遍历次序

B. 完全二叉树的中序遍历次序

C. 完全二叉树的后序遍历次序

D. 完全二叉树的层次遍历次序

192）在完全二叉堆中（大顶堆），不正确的是

A. 任何节点的数值不超过其父亲

B. 兄弟节点之间的没有确定的大小关系

C. 节点的数值不超过其任何一个祖先

D. 整个堆中的最大元素在底部的叶子节点

193）当前完全二叉堆物理上作为向量是{10, 5, 8, 3, 2, 7}，插入新元素9后变为：

A. {2, 3, 5, 7, 8, 9, 10}

B. {10, 9, 8, 7, 5, 3, 2}

C. {10, 5, 8, 3, 2, 7, 9}

D. {10, 5, 9, 3, 2, 7, 8}

194）当前完全二叉堆物理上作为向量是{10, 5, 8, 3, 2, 7},调用delMax()后变为：

A. {5, 8, 3, 2, 7}

B. {2, 3, 5, 7, 8}

C. {8, 5, 7, 3, 2}

D. {8, 7, 5, 3, 2}

195）现有n个元素需要组织成一个完全二叉堆，若使用不断插入所有元素的方法，整个过程是：

A. 自上而下的上滤

B. 自上而下的下滤

C. 自下而上的上滤

D. 自下而上的下滤

196）现有n个元素需要组织成一个完全二叉堆，若使用Floyd算法，整个过程是：

A. 自上而下的上滤

B. 自上而下的下滤

C. 自下而上的上滤

D. 自下而上的下滤

197）堆排序在流程上类似于以前学过的哪种排序?

A. 选择排序

B. 插入排序

C. 冒泡排序

D. 归并排序

198）堆排序的时间复杂度为:

A. O(n)

B. O(nlogn)

C. O(nlognlogn)

D. $O(n^2)$

199）堆排序的空间复杂度为：

A. O(1)

B. O(n)

C. O(nlogn)

D. $O(n^2)$

200）文本串的长度为n，模式串的长度为m，蛮力匹配的时间复杂度为

A. O(m)

B. O(n)

C. O(mn)

D. O(mlogn)

201）下面哪位不是KMP算法的发明者：

A. Knuth

B. Morris

C. Pratt

D. Prim

202) KMP算法的过程中，若某次比对在模式串P的第j个位置P[j]处失败，则将对齐位置换为：

A. prev[j]

B. next[j]

C. prev[j] + 1

D. next[j] + 1

203)对于长度为n的文本串和长度为m的模式串，KMP算法的时间复杂度为：

A. $O(n^2)$

B. O(mn)

C. O(mlogn)

D. O(m+n)

204)给定一个进行串匹配的算法，如何衡量它的效率?

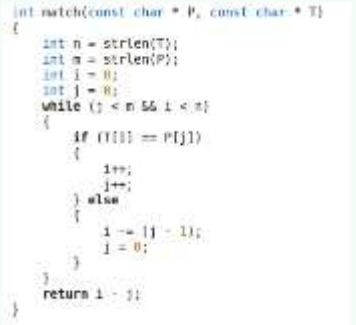
A. 随机生成大量的文本串T和模式串P作为输入，通过实验的方法进行测量

B. 认为所有不同文本串T和模式串P出现的概率是相等的，依此计算时间复杂度的期望

C. 对于成功匹配和失败匹配两种情况分别讨论其时间复杂度

D. 选取固定的文本串T，随机选取模式串P，计算时间复杂度的期望

205) 以下是蛮力串匹配的代码:



```
int match(const char * P, const char * T)
{
    int n = strlen(T);
    int m = strlen(P);
    int i = 0;
    int j = 0;
    while (j < n && i < m)
    {
        if (T[i] == P[j])
        {
            i++;
            j++;
        } else
        {
            i -= j - 1;
            j = 0;
        }
    }
    return i - j;
}
```

当匹配成功/失败时的返回值分别为:

A. P在T中首次出现的位置 / -1

B. P在T中首次出现的位置 / 一个大于n-m的数

C. P在T中最后一次出现的位置 / 一个大于n-m的数

D. P在T中最后一次出现的位置 / -1

206）在文本串YHNMQWERTYFLNYCQWERTYFGIOERNSJTYAFFA中用KMP算法查找模式串QWERTYFLNYCQWERTYO

```
    QWERTYFLNYCQWERTYO
YHNMQWERTYFLNYCNQWERTYFGIOERNSJTYAFFA
```

下一步的对齐位置是

A.
```
    QWERTYFLNYCQWERTYO
YHNMQWERTYFLNYCNQWERTYFGIOERNSJTYAFFA
```

B.
```
      QWERTYFLNYCQWERTYO
YHNMQWERTYFLNYCNQWERTYFGIOERNSJTYAFFA
```

C.
```
                QWERTYFLNYCQWERTYO
YHNMQWERTYFLNYCNQWERTYFGIOERNSJTYAFFA
```

D.
```
      QWERTYFLNYCQWERTYO
YHNMQWERTYFLNYCNQWERTYFGIOERNSJTYAFFA
```

207）KMP算法的查询表为next[]，模式串为P，若P[0,j)与文本串匹配，而在P[j]处失配，则：

A. P[0, next[j]) = P[j - next[j], j)

B. P[0, next[j]+1) = P[j - next[j], j+1)

C. P[0, next[j] - 1) = P[j - next[j], j)

D. P[0, next[j]) = P[j - next[j] - 1, j)

208) 令 A = {t | P[0, t) = P[j - t, j)} ，即A是所有使得P[0,j)的前缀与后缀相等的长度t，如何计算next[j]?

A. next[j] = min A

B. next[j] = max A

C. next[j] = |A| (A中的元素个数)

D. next[j] = max A - |A|

209) 在通过next[j]计算next[j+1]的递推过程中next[j+1] == next[j] + 1当且仅当:

A. j = 0

B. P[j] = P[next[j] - 1]

C. T[j] = P[j]

D. P[j] = P[next[j]]

210）对于模式串CHINCHILLA，计算其next[]

A. -1 0 0 0 0 1 2 3 0 0

B. -1 2 5 4 5 6 2 3 9 1

C. 1 0 0 2 6 7 2 5 7 1

D. 0 0 1 -1 0 0 0 3 2 0

211）迄今为止，我们已经学过许多种排序算法了，请根据描述选择对应的算法(请填入选项大写字母)
A，快速排序 B，堆排序 C，归并排序 D，插入排序 E，冒泡排序
反复比较相邻元素，若为逆序则交换，直至有序：_____
将原序列以轴点为界分为两部分，递归地对它们分别排序_____
将原序列前半部分和后半部分，分别排序，再将这两部分合并：_____
不断从原序列中取出最小元素：_____
抓扑克牌时人们常用的排序方法：_____

212）对于规模为n的向量，快速排序在平均情况下得时间复杂度为

A. $O(n^2)$

B. $O(n\log n)$

C. $O(n)$

D. $O(n+\log n)$

213）对序列A[0，n)用快速排序算法进行排序，u和v是该序列中的两个元素。
在排序过程中，u和v发生过比较，当且仅当（假定所有元素互异）：

A. u < v

B. u在某次被选取为轴点

C. 对于所有介于u和v之间的元素（包括u和v本身），它们之中第一个被选为轴点的是u或者v

D. 所有比u和v都小的元素都始终没有被选为轴点

214)快速排序算法选取轴点时可以采取不同的策略，本题试图用实例说明“三者取中”的策略比随机选取的策略倾向于得到更平衡的轴点
设待排序序列的长度n很大，若轴点的选取使得分割后长/短子序列的长度比大于9:1，则称为不平衡
针对不同的轴点选取策略，估计其发生不平衡的概率(请填十进制小数)：

从n个元素中等概率随机选取一个作为轴点：
____

从n个元素中等概率选取三个元素，以它们的中间元素作为轴点：____

215)快速排序基于的思想是:

A. 减而治之

B. 分而治之

C. 动态规划

D. 递归跟踪

216）对于规模为n的向量，快速排序在平均情况下得时间复杂度为:

A. $O(n^2)$

B. O(nlogn)

C. O(n)

D. O(n+logn)

答案

1)C 2)D 3)A 4)B 5)B 6)D 7)B 8)B 9)B 10)D 11)D 12)C 13)C 14)B 15)A 16)B 17)A 18)C
19)B 20)B 21)A 22)C 23)A 24)D 25)A 26)C 27)B 28)A 29)C 30)D 31)C 32)1 33)D 34)A
35)D 36)C 37)B 38)C 39)A 40)B 41)A 42)C 43)C 44)D 45)A 46)C 47)C 48)A 49)C 50)C
51)B 52)C 53)C 54)C 55)B 56)B 57)C 58)A 59)C 60)B 61)B 62)B 63)C 64)B 65)B 66)14
67)D 68)B 69)B 70)A 71)A 72)D 73)C 74)B 75)C 76) 123!×+45×7-/ 77)C 78)A 79)B
80)D 81)B 82)C 83)D 84)C 85)A 86)C 87)B 88)B 89)A 90)D 91)D 92)C 93)D 94)C 95)D
96)D 97)A 98)B 99)C 100)C 101)B 102)C 103)B 104)C 105)A 106)B 107)B 108)D 109)14
110)11 111) D 112)C 113)B 114)B 115)C 116)B 117)B 118)B 119)A 120)× 121)D 122)A
123)√ 124)B 125)A 126)C 127)A 128)B 129)B 130)B 131)C 132)C 133)A 134)× 135)×
136)× 137)× 138)√ 139)B 140)A 141)C 142)D 143)A 144)B 145)B 146)3,9 147)B 148)D
149)A 150)0.5 151)D 152)C 153)D 154)D 155)A 156)C 157)B 158)B 159)C 160)B
161)190,19 162)7 163)A 164)D 165)B 166)C 167) 1,5,4,6 168) 2,13,6,11 169)√ 170)√
171)D 172)A 173)C 174)C 175)√ 176)B 177)B 178)B 179)√ 180)√ 181)A 182)A 183)C
184)A 185)C 186)A 187)B 188)A 189)A 190)B 191)D 192)D 193)D 194)C 195)A 196)D
197)A 198)B 199)A 200)C 201)D 202)B 203)D 204)C 205)B 206)C 207)A 208)B 209)D
210)A 211)E,A,C,B,D 212)B 213)C 214)0.2,0.056 215)B 216)C