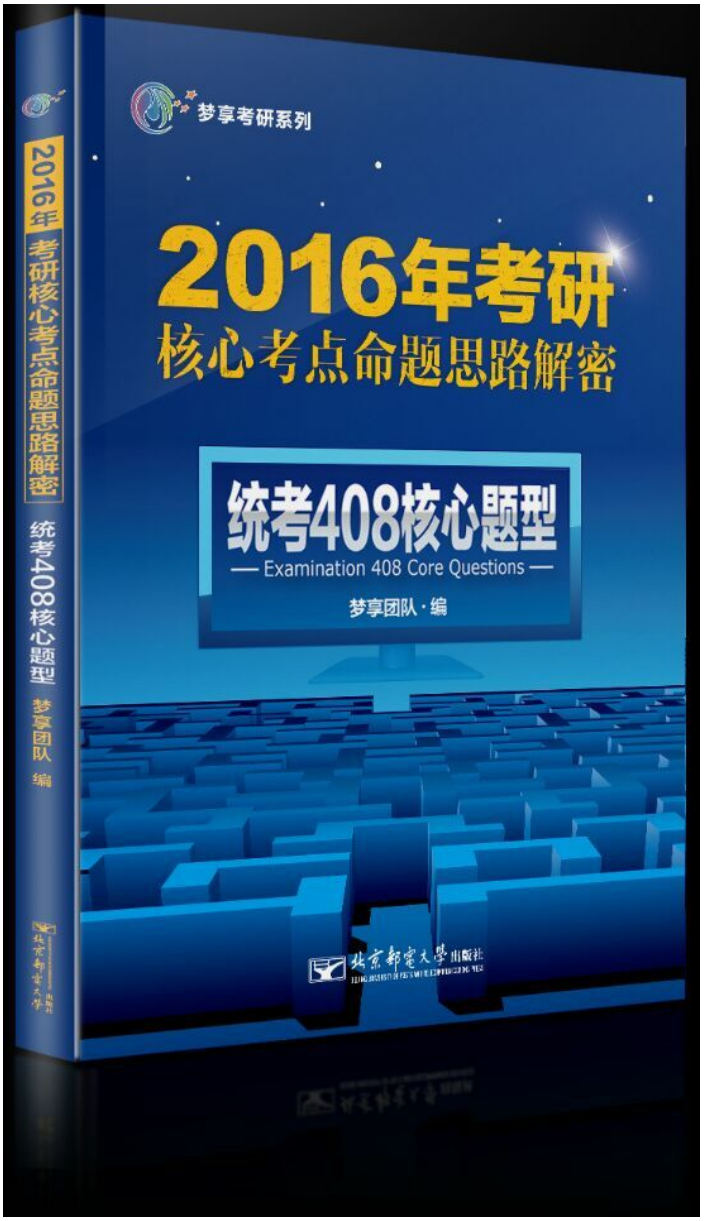
梦享考研系列

2016 年考研终极冲刺 800 题

数 据 结 构

最后 50 天了，再冲刺一次吧，为考研再努力一次!

本资料改编自《2016 年统考 408 核心题型》**数据结构部分和近 3 年高校自主命题的数据结构真题。战友们，让我们再冲锋一次吧！**



2017 年本书拟改写成《2017 年考研终极冲刺 800 题——数据结构》的部分内容。

目录

第 1 章 算法的时间复杂度和空间复杂度

温馨提示：算法的时间复杂度和空间复杂度几乎在每一年的统考中都会命中一个题，基本的要求是考生能够根据算法来分析时间复杂度。

1. 算法分析的主要内容是（　　）。

A. 正确性

B. 可读性和稳定性

C. 简单性

D. 空间复杂性和时间复杂性

【2014—武汉科技大学】

【考查内容】算法分析的主要内容。

【解析】算法是对特定问题求解过程的描述，是指令的有限序列。算法主要包括以下几个重要特性：

(1). 有穷性。 算法的执行步骤必须是有限的，每一步的执行时间也必须是有限的，即算法必须在经过有限的执行时间后完成。

(2). 确定性。 算法中的每一条指令，必须有确切的含义，不能有一个或者多个含义。对于相同的输入，其输出是相同的。

(3). 可行性。 算法描述的操作，可以通过执行已经实现的基本运算执行有限次获得。

(4). 输入和输出。 算法可以有零个或者多个输入，并且至少有一个输出。

值得注意的是，算法的主要性质，并不是算法分析的主要内容。算法分析的主要内容是算法的效率，通常指算法的时间复杂度和空间复杂度两种度量方式。所谓算法的时间复杂度，是指执行算法所需要的计算工作量。 一个算法的空间复杂度，一般是指执行这个算法所需要的内存空间。

【参考答案】D

2. 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;
```

A. $O(\log_2 n)$　　B. $O(n)$

C. $O(n\log_2 n)$　　D. $O(n^2)$

【2011 年统考——第 1 题】

【考查内容】算法的时间复杂度。

【解析】首先，我们要找到程序中执行频率最高的语句，来计算算法的时间复杂度。显然，语句“x=2*x”的执行频率最高。假设该语句总共执行了 k 次，那么第 k+1 次执行时因为 $2^{k+1} \geq n/2$而退出执行。

值得注意的是，在计算算法的时间复杂度时，通常我们都会忽略系数和抓大端。比如某个算法的执行次数与 n 的关系为 $50n^2+100n+10000$，抓大端时我们只考虑阶最高的项，即 $50n^2$，然后再删除系数 50，得到算法的时间复杂度为 $O(n^2)$。

这里一样的道理，有 $k+1 \approx \log_2(n/2)$，程序的执行次数 k 约为 $\log_2(n/2)-1$，算法的时间复杂度为 $O(\log_2 n)$ 。

【参考答案】A

3. 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. $O(n^2)$

【2012 年统考——第 1 题】

【考查内容】算法的时间复杂度。

【解析】时间复杂度是由语句频度分析得来. 递归算法中重复执行的语句主要是调用. 所以递归算法的时间复杂度分析主要是分析递归算法中递归函数调用的次数。

对于本题，咱们给出一个递归的图解过程，如图 1.1 所示。

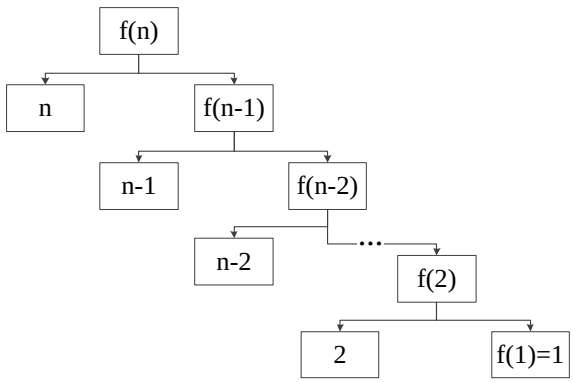


图 1.1

从图解可知，该递归过程是线性的。总共递归执行了 n 次，时间复杂度为 O(n)。

【参考答案】B

4. 已知两个长度分别为 m 和 n 的升序链表，若将它们合并为一个长度为 m+n 的降序链表，则最坏情况下的时间复杂度是（　　）。

A. O(n)　　B. O(m×n)

C. O(min(m,n))　　D. O(max(m,n))

【2013 年统考——第 1 题】

【考查内容】两个升序链表合并成降序链表的时间复杂度。

【解析】我们假设两个升序链表 A 和 B 分别如下图所示。图 1.2 会出现时间复杂度最大的情况（呵呵…请同学自己想想还有没有其他情况？），即 B 表全部插入之后，A 表剩余的结点采用头插法插入到链表 C 中。

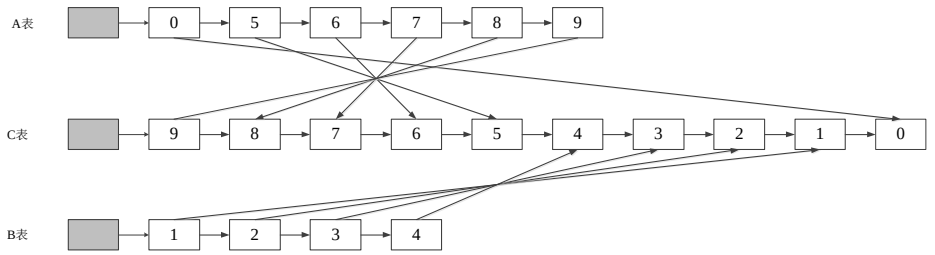


图 1.2

要将图 1.2 的 A 表和 B 表合并成 C 表，显然时间复杂度是 O（m+n）。但是没有这个选项，所以需要咱们再考虑一下哪一个更加接近这个答案。我们再假设一种极端的情况，那就是我们上图的一个延伸。假设 A 表的长度远比 B 表长（即 m 远大于 n），那么时间复杂度 O（m+n）等价于 O（m）。故而，将长度为 m 和 n 的升序链表合并成长度为 m+n 的降序链表，时间复杂度为 O（max(m,n)）。

实际上，$\max(m, n) \leq m + n \leq 2\max(m, n)$，有 $O(m + n)$ 等价于 $O(\max(m, n))$。这样可能便于大家理解一些。

【参考答案】D

5. 下列程序段的时间复杂度是（　　）。

```
count=0;
for(k=1;k<=n;k*=2)
    for(j=1;j<=n;j++)
        count++;
```

A. $O(\log_2 n)$　　B. $O(n)$

C. $O(n\log_2 n)$　　D. $O(n^2)$

【2014 年统考——第 1 题】

【考查内容】算法的时间复杂度。

【解析】从程序可以看出，执行频度最高的语句是“count++”。内层循环的循环变量是 j，与外层层循环的循环变量 i 显然没有半毛钱关系。从内层循环“for(j=1;j<=n;j++)”来看，每一个内层循环都会导致“count++”执行 n 次。

接下来，我们再考虑外层循环“for(k=1;k<=n;k*=2)”。每一次执行外层循环，k 都是以 2 的倍数增加，假设执行了 t 次，则 $2^t = n$,显然有 $t=\log_2 n$。

那么，内外两层循环总共执行了 n*t 次，即 $n\log_2 n$ 次。

【参考答案】C

6. 算法复杂度通常是表达算法在最坏情况下所需要的计算量。一般不用来表达算法复杂度的表达式为（　　）。

A. $O(n^3)$
B. O(100)
C. O(nlogn)
D. $O(1.5^n)$

【2014 年—中山大学】

【考查内容】算法的时间复杂度。

【解析】时间复杂度一般指的是渐进时间复杂度，指当问题规模趋向无穷大时，该算法时间复杂度的数量级。显然，B 答案表示成 O(1)就可以了。

常见的时间复杂度，按数量级递增排列依次为：常数阶 O(1)、对数阶 $O(\log_2 n)$、线性阶 O(n)、线性对数阶 $O(n\log_2 n)$、k 次方阶 $O(n^k)$、指数阶 $O(2^n)$等。

【参考答案】B

7. 下面程序的时间复杂度为（　　）。

```
For（i=0;i<m;i++）
    For (j=0;j<n;j++)
        A[i][j]=i*j;
```

A. $O(m^2)$
B. $O(n^2)$
C. O(m*n)
D. O(m+n)

【2014 年—华东交通大学】

【考查内容】算法的时间复杂度。

【解析】分析算法的时间复杂度时，我们通常先找到执行得最频繁的执行语句，用该语句的执行频度，来度量算法的时间复杂度。显然，语句“A[i][j]=i*j”的执行频度最高。该语句在内层循环需执行 n 次，外层总共循环 m 次。所以总共执行 m*n 次，时间复杂度可以表示成 O(m*n)。

【参考答案】C

第 2 章　线性表

2.1　线性表的定义和基本操作

温馨提示：线性表的定义和基本操作部分，从历年统考和高校自主命题的真题来看，主要考察了两个问题：1、什么是线性表？2、线性表存储方式？3、线性表怎么样插入、删除。请同学们围绕这三个问题，展开复习，其中第 3 个问题是本考点的重点。

1. 在一个单链表中，已知*q 结点是*p 结点的前驱结点，若在*q 和*p 之间插入*s 结点，则执行（　　）。

A. s->next=p->next;p->next=s;　　B. p->next=s->next;s->next=p;

C. q->next=s;s->next=p;　　D. p->next=s;s->next=q;

【2013 年——昆明理工大学】

【考查内容】在单链表中插入结点的正确操作。

【解析】单链表中插入和删除结点操作是常见的操作，希望大家要多练习，掌握这些基本的操作。说透了，其实不同的题目中，这些操作都大同小异的，学会灵活应用，才是硬道理。

已知*q 结点是*p 结点的前驱结点，在*q 结点和*p 结点之间插入*s 结点，如图 2.1 所示。

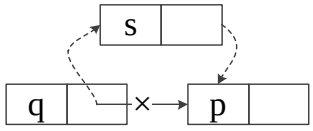


图 2.1

注意插入结点的过程中，要防止“掉链”的情况（即指针调整的过程中，后续指针出现错误）。为了防止“掉链”，需要把 q 的指针先赋给 s 的指针，然后再调整 q 的指针指向结点*s。即“s->next=q->next; q->next=s”。

当然，若 3 个结点的指针都很明确，像本题三个结点的指针 p、q、s 均已知的情况下，可以直接调整指针“q->next=s;s->next=p”。

A 答案应该改成：“s->next=q->next; q->next=s”，B 答案显然不正确，D 答案可改成:“q->next=s;s->next=p”。

【参考答案】C

2. 在长度为 n 的顺序表中删除第 i 个元素（1≤ i ≤n）时，元素移动的次数为（　　）。

A. n-i　　B. n-i+1　　C. i　　D. i+1

【2008 年——太原科技大学】

【考查内容】顺序表中删除元素需要移动元素的次数。

【解析】在长度为 n 的顺序表中删除第 i 个元素（1≤ i ≤n）时，将第 i+1 个位置的元素移动到第 i 个位置，将第 i+2 个位置的元素移动到第 i+1 个位置,…,将第 n-1 个位置的元素移动到第 n-2 个位置，将第 n 个位置的元素移动到第 n-1 个位置。图 2.2 给出元素的移动示意图。

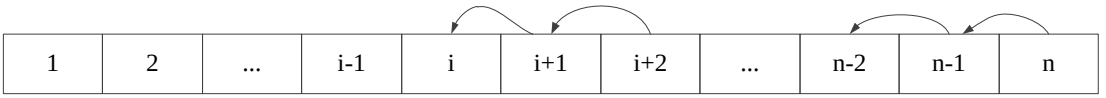


图 2.2

显然，除了从第 i+1 个元素到第 n 个元素总共 n-i 个元素都需要移动。

为什么不是 n-(i+1)呢？比如，从第 3 到第 5 个元素都需要移动，也就是说 i=2,n=5,那么需要移动的元素个数是 n-i=5-2=3 个，而不是 n-(i+1)=5-3=2 个。这些细节的问题，请同学们要注意了，不要弄错了。

【参考答案】A

3. 下列关于线性表的叙述中，错误的是（　　）。

A. 顺序表是使用一维数组实现的线性表

B. 顺序表必须占用一片连续的存储单元

C. 顺序表的空间利用率高于链表

D. 在链表中，每个结点只有一个链域

【2013 年——江苏大学】

【考查内容】线性表的相关概念。

【解析】顺序表一般代表的是使用一维数组实现的线性表，必须占用一片连续的存储空间。线性表结点之间的关系是靠存储位置（数组下标）来表达的，而链表的结点之间的关系是靠指针来表示的。链表需要额外的指针域，所以空间利用率高于链表。

在链表中，每一个结点可能不仅仅只有一个链域，例如双向循环链表的每一个结点既有指向其前驱的指针域，也有指向其后继结点的指针域。

【参考答案】D

4. 若线性表最常用的操作是在表头和表尾进行的，则最节省时间的存储方式是（　　）。

A. 单链表

B. 仅有头指针的单循环链表

C. 双链表

D. 仅有尾指针的单循环链表

【2013 年——黑龙江大学】

【考查内容】线性表的基本操作。

【解析】单链表在表头操作时（如添加一个结点）的时间复杂度为 O(n)，在表尾操作时需要顺序遍历链表，找到表尾结点，时间复杂度为 O(n)。

在单链表中，将终端结点的指针域 NULL 修改并指向表头结点或者开始结点（注意区分表头结点和开始结点），就得到了单链表形式的循环链表，简称为单循环链表。显然，在单循环链表中，在表头和表尾操作的时间复杂度分别为 O(1)和 O(n)。

注意 C 答案说的是**双链表，不是双向循环链表**，在表头和表尾操作的时间复杂度仍然分别为 O(1)和 O(n)。因为要对表尾操作，仍然需要遍历链表。

对于仅有尾指针的单循环链表，在表尾插入一个元素，或者在表头插入一个元素等操作都比较方便。

【参考答案】D

5. 线性表采用链式存储时，其地址（　　）。

A. 必须是连续的

B. 部分地址必须是连续的

C. 一定是连续的

D. 连续与否均可

【2015 年—暨南大学】

【考查内容】线性表的存储特点。

【解析】常见的线性表存储方式有顺序存储和链式存储两种。顺序存储方式下，元素存储在一片连续的地址空间中，便于查找，存储效率高，但是插入和删除元素平均需要移动近表中一半的元素。链式存储下，元素可以存储在地址不连续的存储单元中，查找和删除元素不需要移动表中的其他元素。但是，因为每一个结点都有一个指针域，所以增加了存储空间的开销。

【参考答案】D

6. 已知单链表上的一个结点的指针为 p，则删除该结点后继的正确操作语句是（　　）。

A. s=p->next; p=p->next; free(s);

B. p=p->next; free(p);

C. s=p->next; p->next=s->next; free(s);

D. p=p->next; free(p->next);

【2015 年—暨南大学】

【考查内容】链表的删除结点操作。

【解析】删除 p 结点的后继结点，删除的结果，应该是 p 指针的后继是其原来后继结点的后继结点。我们分析四个选项：

A 答案令一个新的指针 s 指向 p 的后继，再调整 p 指针指向被删除结点，删除 s 指针指向的结点后，p 指针变为 NULL，链表也断开了。

B 答案，令 p 指针指向等待删除的 p 的后继结点，然后释放 p，也会导致链表断开的问题。

C 选项中，s=p->next 令指针 s 指向 p 结点的后继结点。 p->next=s->next 令 p 指针的后继结点调整到原来后继结点的后继结点。最后，通过执行 free(s)释放被删除结点的空间。所以，C 选项满足题意。

D 选项，通过执行 p=p->next，p 指针后移，指向被删除的结点。但是，接下来执行的语句却是 free(p->next)，该语句删除的是希望被删除的结点的后继结点，导致删除了错误的结点。所以，该选项也错误。

【参考答案】C

【总结】

链表的插入和删除，是历年高校数据结构特别喜欢出题的地方。提供的选项会出现以下 3 个问题：

(1). 链表断开。链表断开，是指删除了结点之后，原来的链表断成了好几个子链表，连接不起来了。如本题的选项 A 和 B。

(2). 删除结点错误。删除结点错误，是指本来让删除的是某一个结点，结果删除的却是其他的不想删除的结点。如本题的选项 D。

(3). 插入结点的位置错误。指的是，本来结点应该插入在位置 1，结果插入到了位置 2。

请同学们做链表删除和插入结点的题的时候，从这 3 个方面着手，来解析该类题。

7. 在单链表指针为 p 的结点之后插入指针为 s 的结点，正确的操作是（　　）。

A. p->next=s;s->next=p->next;

B. s->next=p->next;p->next=s;

C. p->next=s;p->next=s->next;

D. p->next=s->next;p->next=s;

【2014—中南大学】

【考查内容】链表结点的插入

【解析】在单链表指针 p 之后插入一个指针 s 指向的结点,其实操作很简单。第一步，把 s 的后继调整，指向它插入后的正确后继，即 p 的后继，即执行 s->next=p->next。第二步，把指针 p 的后继修改，使其指向新插入的结点 s，即 p->next=s。链表的插入过程其实最关键是插入位置正确，指针调整正确，使得链表不断开。

【参考答案】B

8. 有关单链表的正确描述是（　　）。

A. 在 O(1)时间内找到指定的关键字

B. 在插入和删除操作时无需移动链表结点

C. 在 O(1)时间内删除指定的关键字

D. 单向链表的存储效率高于数组的存储效率

【2014—中山大学】

【考查内容】单链表的基本操作和存储效率。

【解析】顺序表（以数组为例）能够直接根据数组下标计算元素的地址，所以能够在O(1)的时间复杂度内访问到元素。但是单链表必须从表头一个一个往下找，直到找到元素为止。所以，A 答案错误。删除结点也必须从表头开始查找，找到元素的前驱，然后令该前驱结点的后继指针指向要删除元素的后继，并释放所删除结点的物理空间。所以，C 答案错误。

与数组不同，单链表除了存储元素之外，还需要至少一个指针域存储指针，所以单链表的存储效率低于数组的存储效率。所以 D 答案错误。

单链表在插入和删除元素时，不需要移动元素。顺序表插入和删除元素平均需要移动一半表长的元素。所以，由顺序表转向链表是典型的空间换时间的方法。

【参考答案】B

2.2 线性表的实现

温馨提示：线性表的实现，考查同学们怎么样利用线性表来实现一个算法。所以，本考点在真题中通常以算法题的形式出现，要求同学们写出算法的基本思想，并用某一种编程语言实现该算法，最后分析算法的时间复杂度，请同学们多留意，本考点特别重要。

1. 线性表若采用链式存储结构时，要求内存中可用存储单元的地址（　　）。

A. 必须是连续的　　B. 部分地址必须是连续的

C. 一定是不连续的　　D. 连续不连续都可以

【2013 年——沈阳工业大学】

【考查内容】链式存储结构的结点地址特点。

【解析】线性表的链接存储结构称为单链表。单链表用一组任意的存储单元存放线性表的数据元素，这组存储单元可以连续，也可以不连续，甚至可以零散分布在内存中的任意位置。

【参考答案】D

2. 给定一个有 n 个元素的线性表。若采用顺序存储结构，则在等概率前提下，向其插入一个元素需要移动的元素个数平均为（　　）。

A. n+1　　B. n/2

C. (n+1)/2　　D. n

【2012 年——华南理工大学】

【考查内容】顺序表中插入元素的平均移动元素个数。

【解析】长度为 n 的顺序表共有 n+1 个插入位置，如图 2.3 所示。

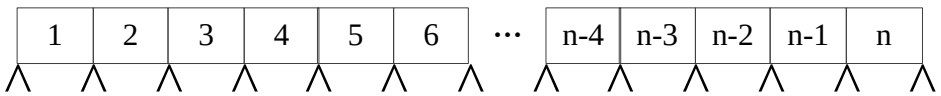


图 2.3

从表尾向表头方向开始计算，在第 n+1 个位置插入元素需要移动 0 个元素，在第 n 个位置插入元素需要移动 1 个元素，…，在第 1 个位置（位置从 1 开始算起）插入元素需要移动 n 个元素。故而，插入一个元素平均需要移动元素为

$$\text{Average} = \frac{(\sum_{i=0}^{n} i)}{n+1} = \frac{n(n+1)}{2} \times \left(\frac{1}{n+1}\right) = \frac{n}{2}$$

【参考答案】B

3. 下面关于线性表的叙述中，错误的是哪一个？（　　）

A. 线性表采用顺序存储，必须占用一片连续的存储单元

B. 线性表采用顺序存储，便于进行插入和删除操作

C. 线性表采用链接存储，不必占用一片连续的存储单元

D. 线性表采用链接存储，便于插入和删除操作

【2010 年——中山大学】

【考查内容】线性表的顺序存储和链式存储的存储空间特点。

【解析】线性表采用顺序存储时，必须占用一片连续的存储空间，插入和删除元素平均需要移动将近一半的表长元素。而采用链式存储时，不必占用一片连续的存储空间，而且插入和删除元素不需要移动元素，比较方便。

【参考答案】B

4. 设单链表中指针 p 指着结点 A，若要删除 A 之后的结点(若存在)，则需要修改指针的操作为(　　)。

A. p->next=p->next->next　　B. p=p->next

C. p=p->next->next　　D. p->next=p

【2013 年——暨南大学】

【考查内容】单链表中删除结点操作。

【解析】我们假设单链表如图 2.4 所示。

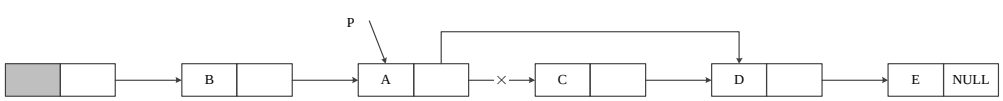


图 2.4

如上图所示，指针 p 指向结点 A，若要删除结点 A 之后的结点 C，只需调整指针，令结点 A 的指针指向结点 D 即可。本题中，即令 p->next 指向 p->next->next，故而选择 A 答案。

【参考答案】A

5. 某线性表中最常用的操作是在最后一个元素之后插入一个元素和删除第一个元素，则采用（　　）储方式最节省运算时间。

A. 单链表

B. 仅有头指针的单循环链表

C. 双链表

D. 仅有尾指针的单循环链表

【2015 年—北京邮电大学】

【考查内容】链表的性质。

【解析】首先，我们来分析一下在最后一个元素之后插入一个元素或者删除第一个元素，四个选项的时间复杂度。

对于单链表而言，删除第一个元素比较快，直接调整指向链表的指针（链表可能没有头指针：指向头结点的指针，头结点一般数据域为空）指向第二个结点就可以了。时间复杂度为 O(1)。在最后一个元素之后插入一个元素，需要遍历链表，时间复杂度为 O(n)。对于仅有头指针的单循环链表和双链表来说，删除第一个元素，和在表尾插入一个新元素，时间复杂度和单链表一样。

与 A、B、C 选项都不同的是，仅有尾指针的单循环链表，在删除第一个元素只需要调整尾指针指向第二个元素即可。在表尾插入一个新的元素，只需要将新插入结点的指针指向尾指针的后继，再调整尾指针指向该插入结点即可。这两种操作的时间复杂度都是 O(1)。

【参考答案】D

6. 已知一个带有表头结点的单链表，结点结构为：

data	link

假设该链表只给出了头指针 list。在不改变链表的前提下，请设计一个尽可能高效的算法，查找链表中倒数第 k 个位置上的结点（k 为正整数）。若查找成功，算法输出该结点的 data 域的值，并返回 1；否则，只返回 0。要求：

(1). 描述算法的基本设计思想。

(2). 描述算法的详细实现步骤。

(3). 根据设计思想和实现步骤，采用程序设计语言描述算法（使用 C、C++或 Java 语言实现），关键之处请给出简要注释。

【2009 年统考——第 42 题】

【解析】本题属于比较基础的算法题。下面，给出详细地解答过程。

(1). 算法的基本思想：定义两个指针 p 和 q，开始时均指向链表头结点的下一个结点位置；令 p 指针沿着链表移动，并且启动计数器 count 开始计数（初始时，count=0）；当 p 指针移动到链表表尾时，q 指针所指向的结点即倒数第 k 个结点。

(2). 算法的详细实现步骤如下：

(A). 令 p 和 q 均指向链表表头结点的下一个结点，令 count=0;

(B). 若 p 为空，即该单链表只有一个表头结点，则转(E);

(C). 若 count=k，则 q 指向下一个结点。否则，count=count+1;

(D). p 指向下一个结点，转(B);

(E). 若 count=k,则查找成功，输出该结点的 data 域的值，并返回 1；否则，查找失败，返回 0。

(3). 采用 C 语言，实现如上的算法，代码如下。

```
  typedef struct LNode{
        int data;
        struct LNode * link;
    } * LinkList;
int SearchN(LinkList list,int k)
    {
          LinkList p,q;
          int count=0;                                  /*计数器赋初值*/
          p=q=list->link;                          /*p 和 q 指向链表表头结点的下一个结点*/
          while(p!=NULL){
               if(count<k) count++;                  /*计数器+1*/
               else q=q->link;                       /*q 移到下一个结点*/
               p=p->link;                            /*p 移到下一个结点*/
          }
          if(count<k)return(0);                    /*如果链表的长度小于 k,查找失败*/
          else
          {
               printf("%d",q->data);                 /*查找成功*/
               return (1);
          }
    }
```

7. 设将 n（n>1）个整数存放到一维数组 R 中。试设计一个在时间和空间两方面都尽可能高效的算法。将 R 中保存的序列循环左移 p（0<p<n）个位置，即将 R 中的数据由（X_0, X_1, ..., X_{n-1}）变换为（X_p, X_{p+1},…, X_{n-1}, X_0, X_1,…, X_{p-1}）。要求：

(1). 给出算法的基本设计思想。

(2). 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。

(3). 说明你所设计算法的时间复杂度和空间复杂度。

【2010 年统考——第 42 题】

【解析】

(1). 本算法的基本思想为：先将 n 个数据 $x_0x_1\cdots x_p\cdots x_{n-1}$ 原地逆置，得到 $x_{n-1}\cdots x_px_{p-1}\cdots x_0$,然后再将前 n-p 个和后 p 个元素分别原地逆置，得到最终结果:$x_px_{p+1}\cdots x_{n-1}x_0x_1\cdots x_{p-1}$。

(2). 算法实现如下。

```
void reverse(int r[],int left,int right)
 {
      int k=left,j=right,temp;              //k 等于左边界 left,j 等于右边界 right
      while(k<j)
      {                                     //交换 r[k]和 r[j]
           temp=r[k];
           r[k]=r[j];
           r[j]=temp;
           k++;                             //k 右移一个位置
           j--;                             //j 左移一个位置
      }
  }
  void shift(int r[], int n, int p)
  {
      if(p>0 && p<n)
       {
            reverse(r,0,n-1);               //将表 r 中的全部元素逆置
            reverse(r,0,n-p-1);             //将表 r 中 0 到 n-p-1 的元素逆置
            reverse(r,n-p,n-1);             //将表 r 中 n-p 到 n-1 的元素逆置
       }
  }
```

(3). 上述算法 shift 中，总共逆置了 3 次表，第一次将表中的元素全部逆置，第二次逆置了表中元素 0 到 n-p-1 的部分，第二次逆置了 n-p 到 n-1 的部分。而 shift 里面的函数 reverse 里面只有一个循环，可见算法的时间复杂度为 O(n)。

该算法借用了一个临时变量 temp，用来暂存当前待交换的元素，所以算法的空间复杂度为 O(1)。

第 3 章　栈、队列和数组

3.1　栈和队列的基本概念

温馨提示：栈和队列部分，考实际算法的题目近几年很少出现，倒是基本的栈和队列基本操作经常出现。希望同学们熟悉栈和队列的基本性质，懂得基本的操作。

1. 设栈 S 和队列 Q 的初始状态均为空，元素 a，b，c，d，e，f，g 依次进入栈 S。若每个元素出栈后立即进入队列 Q，且 7 个元素出队的顺序是 b，d，c，f，e，a，g，则栈 S 的容量至少是（　　）。

A. 1　　B. 2　　C. 3　　D. 4

【2009 年统考——第 2 题】

【考查内容】栈和队列的基本操作。

【解析】元素 a，b，c，d，e，f，g 依次进入栈 S，元素出栈之后立即进入队列 Q。根据队列的基本性质，可知这 7 个元素的出队列顺序与出栈顺序是一样的。我们可以仅简单地考虑这 7 个元素的出栈顺序。

要得到 b，d，c，f，e，a，g 这样的出队顺序，可以对栈 S 进行如表 3.1 操作。

表 3.1

顺序	操作	栈内	栈外	顺序	操作	栈内	栈外
1	a 入栈	a		8	e 入栈	ae	bdc
2	b 入栈	ab		9	f 入栈	aef	bdc
3	b 出栈	a	b	10	f 出栈	ae	bdcf
4	c 入栈	ac	b	11	e 出栈	a	bdcfe
5	d 入栈	acd	b	12	a 出栈		bdcfea
6	d 出栈	ac	bd	13	g 入栈	g	bdcfea
7	c 出栈	a	bdc	14	g 出栈		bdcfeag

根据以上分析，栈中元素最多时，是栈中有元素 a,而 b 和 c 依次入栈的情况，以及栈中只有 a 时元素 e、f 分别入栈的情况。显然，栈最小的容量应该能容纳 3 个元素。

【参考答案】C

2. 若元素 a、b、c、d、e、f 依次进栈，允许进栈、退栈操作交替进行，但不允许连续三次进行退栈操作，则不可能得到的出栈序列是（　　）。

A. d c e b f a　　B. c b d a e f

C. b c a e f d　　D. a f e d c b

【2010 年统考——第 1 题】

【考查内容】栈的基本操作。

【解析】我们来看看每一个答案对应的序列需要进行的操作。

(1). 要得到序列 d c e b f a，需将 a、b、c、d 依次入栈，d、c 依次出栈，e 入栈，e、b 依次出栈，f 入栈，f、a 依次出栈。

(2). 要得到序列 c b d a e f，需将 a、b、c 依次入栈，c、b 依次出栈，d 入栈，d、a 依次出栈，e 入栈，e 出栈，f 入栈，f 出栈。

(3). 要得到序列 b c a e f d，需将 a、b 依次入栈，b 出栈，c 入栈，c、a 依次出栈，e 进栈，e 出栈，f 入栈，f、d 出栈。

(4). 要想得到序列 a f e d c b，可以将 a 入栈之后出栈，然后将 b、c、d、e、f 依次压栈，之后 f、e、d、c、b 依次出栈。但是题目要求，允许进栈、退栈操作交替进行，但不允许连续三次进行退栈操作，所以 D 答案是不可能得到的序列。

【参考答案】D

3. 某队列允许在其两端进行入队操作，但仅允许在一端进行出队操作。若元素 a、b、c、d、e 依次入此队列后再进行出队操作，则不可能得到的出队序列是（　　）。

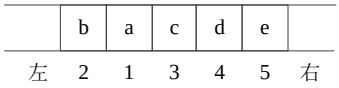
A. b a c d e　　B. d b a c e

C. d b c a e　　D. e c b a d

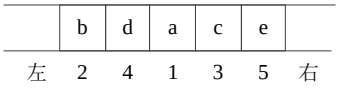
【2010 年统考——第 2 题】

【考查内容】受限双端队列的入队和出队操作。

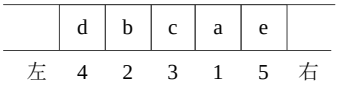
【解析】我们假设队列允许从左边和右边进入，但只允许从左端出队列。则 a、b、c、d、e 依次入队列再出队列，分别如下图 3.1 的(1)、(2)、(3)、(4)所示。



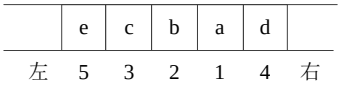
图（1）



图（2）



图（3）



图（4）

图 3.1

下图中的 1~5 分别表示入队列的顺序。显然，A、B、D 答案都能够成立。C 答案中，a 入队列后，b 不管怎么如队列，都会在 a 的左边或者右边，不会存在 bcae 这种情况。

【参考答案】C

4. 元素 a，b，c，d，e 依次进入初始为空的栈中，若元素进栈后可停留、可出栈，直到所有元素都出栈，则在所有可能的出栈序列中，以元素 d 开头的序列个数是（　　）。

A. 3　　B. 4　　C. 5　　D. 6

【2011 年统考——第 2 题】

【考查内容】栈的基本操作。

【解析】元素 a、b、c、d、e 依次进入初始为空的栈中，元素入栈后可停留，可出栈。那么，，必然先将 a、b、c 压入栈中，d 进栈后，出栈，才能在出栈序列中，得到以 d 开头的序列。

d 出栈之后，栈中元素从栈顶到栈底依次为 c、b、a，而且，这三个元素出栈的顺序也为 c、b、a。若 e 进栈后直接出栈，可以得到出栈序列 d、e、c、b、a。若 d,先出栈，e 再入栈，则可以得到出栈序列 d、c、e、b、a。同理，可得到其他的两个出栈序列 d、c、b、e、a 和 d、c、b、a、e。

【参考答案】B

5. 设有 6 个元素按 1、2、3、4、5、6 的顺序进栈，下列不合法的出栈序列是（　　）。

A. 234165

B. 324651

C. 431256

D. 546321

【2014—武汉大学】

【考查内容】栈的基本操作。

【经典总结】不合法的出栈序列，是一种常考的题型。那么，怎么判断哪一个输出序列不合理呢？我们根据入栈和出栈来判断。当非栈顶元素比栈顶元素先出栈时，我们就认为这是一个不合法的序列。

【解析】根据我们的总结，下面我们来分析四个选项是否合法。

A 选项中，首先出栈的元素时 2，说明 1 压入栈底，2 入栈成为新的栈顶之后马上出栈（栈顶是 1）。此时栈中只有 1，之后 3 入栈成为新的栈顶元素，3 出栈（栈顶是 1）；4 入栈代替 1 成为新的栈顶之后，立刻出栈；1 出栈（栈空）；5 入栈（5 是栈顶），6 入栈（6 是栈顶），6 出栈（5 是栈顶），5 出栈（栈空）。由此看，没有比栈顶元素优先出栈的元素，所以出栈序列合法。同理，我们可以推导出 B、D 两个选项也是正确的。

看一下 C 选项。4 先出栈，说明元素 1、2、3、4 依次入栈之后，4 成为新的栈顶，4 接着出栈。此时栈中元素从栈顶到栈底依次为 3、2、1,即栈顶是 3，不是 2。所以出现了如我们总结的，非栈顶元素比栈顶元素首先出栈的不合法操作。所以，选择 C 选项。

【参考答案】C

6. 设栈 S 和队列 Q 的初始状态为空，元素 e1,e2,e3,e4,e5 和 e6 依次入栈 S，一个元素入栈后即进入队列 Q，若 6 个元素出队的序列为 e2,e4,e3,e6,e5,e1。则栈 S 的容量至少应该是（）

A. 6

B. 4

C. 3

D. 2

【2014—云南大学】

【考查内容】栈的基本性质。

【解析】栈的容量其实也是一个常考的内容，不管是统考还是非统考的高校都喜欢出这样的题目。

【经典总结】栈的容量最小值，应该是保证一个入栈序列根据出栈顺序的要求的最小存储空间。元素 e1,e2,e3,e4,e5 和 e6 依次入栈 S，出队的序列为 e2,e4,e3,e6,e5,e1。那么，入栈和出栈的操作序列如下：

元素 e1 入栈，e2 入栈（当前需要栈的容量为 2），e2 出栈，此时栈中只有元素 e1。元素 e3，e4 依次入栈（当前栈的容量为 3）。元素 e4,e3 依次出栈，栈中只有 e1。元素 e5,e6 依次入栈（当前栈的容量为 3），之后 e6,e5,e1 依次出栈。所以，栈的最小容量为 3。

【参考答案】C

3.2 栈和队列的顺序存储结构

温馨提示：栈和队列的顺序存储中，考得比较多的问题是，队列的基本操作、判满、判空、出队列和入队列，以及栈的判满、判空、入栈和出栈的基本操作，请同学们较深入理解这些基本知识。

1. 已知循环队列存储在一维数组 A[0...n-1]中，且队列非空时 front 和 rear 分别指向队头元素和队尾元素。若初始时队列为空，且要求第 1 个进入队列的元素存储在 A[0]处，则初始时 front 和 rear 的值分别是（　　）。

A. 0，0　　B. 0，n-1

C. n-1，0　　D. n-1，n-1

【2011 年统考——第 3 题】

【考查内容】循环队列的基本操作以及队首和队尾指针的值。

【解析】关于这个题目，很多书都有不同的见解，但是都没有把问题讲清楚。我们分两种情况来解析本题。

情况一：队列非空时 front 和 rear 分别指向队头元素和队尾元素

因为 rear 指针始终要指向队尾元素，所以每一个新元素入队列，都需要先将指针加 1，使得队尾指针指到即将用来存放新元素的位置，然后再将新元素放在 rear 指针所指向的该位置。

情况二：队列非空时 front 和 rear 分别指向队头元素和队尾元素的下一个位置

因为 rear 指针始终指向队尾元素的下一个位置，所以当新元素入队列时，先把新元素存放到 rear 指针指向的位置，然后再调整 rear 指针指向新的队尾元素的下一个位置。

在第一种情况下，当初始队列为空时，front 指针指向 A[0]的下标 0，说明第一个入队列的元素要存放在 A[0]的位置，rear 应该指向循环队列的队尾 n-1,。当新元素入队列，rear 先完成+1，此时 rear=0，再将新元素入队列。

在第二种情况下，当初始队列为空时，rear 和 front 都指向 A[0]。当新元素入队列时，先将新元素放在 rear 指定的数组下标 0 位置，然后 rear+1，队尾指针 rear 指向新的队尾元素 A[0]的下一个位置 1。

呵呵...这样讲解，是不是方便同学们理解了呢？

【参考答案】B

2. 顺序存储的循环队列，存储空间大小为 n，队头结点下标为 front，队尾结点下标为 rear。则此循环队列中的元素个数为（　　）。

A. n+front-rear　　B. rear-front+1

C. (rear-front)%n　　D. (n+rear-front+1)%n

【2013 年——华南理工大学】

【考查内容】循环队列的元素个数计算方法。

【解析】不同的学校所使用的教材，可能会对循环队列的队首和队尾指针有着不同的规定。我们在分析华南理工大学的这个真题时，首先我们要记得“%存储空间的大小”，所以排除 A、B 答案。C 答案可能存在 rear 所指向的下标小于 front 所指向的下标的情况，所以排除了 C 答案。

我们再来分析一下 D 答案。rear-front+1+n,则两个指针刚好分别指向表头和表尾的位置，如图中，rear=0,front=3,表中有 0-3+6+1=4 个元素（如图 3.2 所示）。

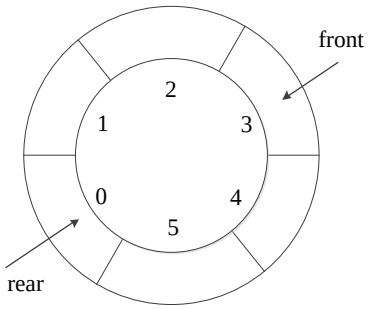


图 3.2

【参考答案】D

【经验心得】本考点常见的 3 种题型：1、计算循环队列中的首尾指针位置；2、计算循环队列中的元素个数；3、循环队列判空和判满。请同学们多注意这三种题型的解法。

3. 若用一个大小为 5 的数组来实现循环队列，且当前 rear 和 front 的值分别为 0 和 2，当从队列中删除 2 个元素,再加入 1 个元素后，rear 和 front 的值分别为多少？（　　）

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

【2010 年——中山大学】

【考查内容】循环队列的基本操作。

【解析】使用数组来实现循环队列，队头元素出队列，队头指针执行 Q.front=(Q.front+1)%MaxSize；队尾有新的元素要入队列，队尾指针执行 Q.rear=(Q.rear+1)%MaxSize。

显然，本题中，MaxSize=5。当从队列中删除 1 个元素，队头指针 Q.front=(2+1)%5=3，再删除一个元素，队头指针 Q.front=(3+1)%5=4。当队尾有一个新的元素入队列，则队尾指针 Q.rear=（0+1）%1=1。

【参考答案】B

4. 循环队列放在一组数组 A[0…M-1]中，End1 指向队头元素，End2 指向队尾元素的后一个位置。假设队列两端均可进行入队和出队操作，队列中最多能容纳 M-1 个元素。初始时为空，下列判断队空和队满的条件中，正确的是（　　）。

A. 队空：End1==End2；　队满：End1==(End2+1) mod M

B. 队空：End1==End2;　队满：End2==(End1+1) mod (M-1)

C. 队空：End2==(End1+1)mod M；　队满：End1==（End2+1）mod M

D. 队空：End1==（End2+1）mod M; 队满：End2==(End1+1)mod (M-1)

【2014 年统考——第 3 题】

【考查内容】循环队列判空和判满的条件。

【解析】根据已知，循环队列中最多能容纳 M-1 个元素，而且 End2 指针指向队尾元素的下一个位置。我们假设某时刻循环队列如图 3.3 所示。

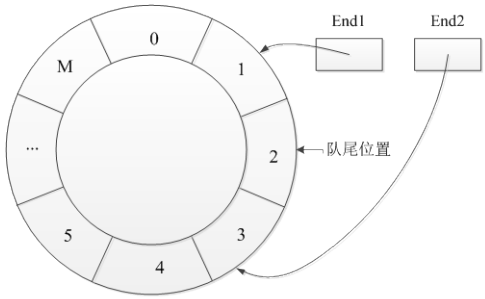


图 3.3

如图 3.3 所示，若某一个时刻，队列为空，则该循环队列中，必然有 End1 和 End2 两个指针指向同一个单元。那么，新的元素入队列时，先将元素放在 End2 的位置，然后再将 End2 的指针后移。因为 End2 指针始终指向队尾的下一个位置，若队满，End1=(End2+1) mod M。

【参考答案】A

5. 循环队列存储在数组 A[0..m]中，则入队时的操作为（　　）。

A. rear=rear+1

B. rear=(rear+1)　mod　(m-1)

C. rear=(rear+1)　mod　m

D. rear=(rear+1)　mod　(m+1)

【2015 年—北京邮电大学】

【考查内容】循环队列的基本操作。

【解析】循环队列存储在数组 A 中，数组 A 的下标从 0~m，总共 m+1 个元素。从题意来看，rear 指针指向了当前队尾的元素，直接让尾指针指向下一个元素即可。但是因为是循环队列，所以需要 mod 存储空间的大小 m+1。所以执行 rear=(rear+1) mod (m+1)。

【参考答案】D

6. 循环队列 Q 的存储空间为 0~m-1，用 front 表示队头，用 rear 表示队尾，采用少用一个单元的方法来区分队列空和满，那么循环队列满的条件是（　　）。

A. Q.rear+1==Q.front

B. (Q.rear+1)%m==Q.front

C. Q.front+1==Q.rear

D. (Q.front+1)%m==Q.rear

【2014 年—河北大学】

【考查内容】循环队列的判满条件。

【解析】我们在《2016 年考研核心考点命题思路解密—数据结构》一书中提到，在保留一个存储单元来区分队空和队满的情况下，循环队列判满的条件为 rear 指针差一个位置就追上 front 指针。而此时的 rear 指针所指向的存储单元，就是这个保留的存储单元。元素

在入循环队列之前需要判断队列是不是满了，即检查(Q.rear+1)%m==Q.front 是否成立。若成立，则队列已满，不能插入新的元素。

【参考答案】B

3.3 栈和队列的链式存储结构

温馨提示：栈和队列的链式存储，考得比较多的是链式存储下，栈和队列的性质、基本操作，请同学们多掌握这方面的知识。本考点近几年很少出现其他的题型，为了给大家减少学习负担，我们仅布置以下题型。

1. 若用单链表来表示队列，则应该选用（　　）。

A. 带头指针的非循环链表　　B. 带头指针的循环链表

C. 带尾指针的非循环链表　　D. 带尾指针的循环链表

【2010 年——湖南大学】

【考查内容】队列的链式存储结构。

【解析】单链表也可用来表示队列。我们都知道，单链表插入和删除操作都比较方便。队列具有先进先出的性质，若用单链表来表示队列，必须考虑什么样的单链表方便新元素入队列和队头元素出队列。

(1). A 答案中，带头指针的非循环链表，若采用开始元素作为队头，则出队列的时间复杂度是 O(1)，入队列的时间复杂度是 O(1)。采用链表尾部作为队头，则入队列的时间复杂度是 O(1)，出队列的时间复杂度为 O(n)。

(2). B 虽然把单链表改成了循环链表，但是没有改变新元素入队和队头元素出队列的时间复杂度。

(3). 相比较带尾指针的非循环链表和带尾指针的循环链表，后者更适合用于表示队列。对于带尾指针的循环链表，可将开始元素作为队头，链尾元素作为队尾。新元素入队列，以及队头元素出队列，时间复杂度都是 O(1)。

【参考答案】D

2. 在下列结论中，正确的是（　　）。

① 因链栈本身没有容量限制，故在用户内存空间的范围内不会出现栈满情况

② 因顺序栈本身没有容量限制，故在用户内存空间的范围内不会出现栈满情况

A. 只有①正确　　B. 只有②正确

C. ①②都正确　　D. ①②都不正确

【2013 年——江苏大学】

【考查内容】栈的链式存储结构和顺序存储结构是否会在用户空间范围内出现栈满的情况。

【解析】链栈本身是没有容量限制的，结点都是动态申请的。所以，在用户空间范围内，不会出现栈满的情况。顺序栈则不同，顺序栈其实就是用一维数组存储的堆栈，数组的大小表示了栈的容量，在用户空间范围内可能出现栈满的情况。

【参考答案】A

3. 在一个带头结点的链队列中，f 和 r 分别为队首尾指针，则进行 s 结点的入队操作时执行（　　）。

A. r->next=s;r=s;　　B. r->next=s;s->next=r->next;

C. s->next=r->next;r=s;　　D. s->next=r->next;r->next=s;

【2013 年——昆明理工大学】

【考查内容】链队列的入队操作。

【解析】在一个带头结点的链队列中，f 和 r 分别表示队首和队尾指针。那么，为了能够让新结点入队列，需将该 s 结点接在尾指针 r 的后面，称为新的尾结点。即“r->next=s;r=s;”。故而，选择 A 答案。

要切记不要出现 B 答案那样的指针错误。B 答案中，先将尾结点的指针域指向结点 s，然后再令结点 s 的指针域指向 r 的指针所指向的结点，即 s->next=s，这显然是不正确的。

【参考答案】A

4. 向一个栈顶指针为 top 的链栈中插入一个 S 所指结点时，则执行（　　）。

A. top->next = S;

B. S->next = top->next;top->next = S;

C. S->next = top; top = S;

D. S->next = top; top = top->next

【2014 年—浙江理工大学】

【考查内容】链栈的入栈操作。

【解析】链栈我们见得比较少，高校命题也比较少。根据题意，我们假设栈顶指针 top 指向当前栈顶，那么向栈中插入一个 S 指针所指向的结点时，只需要让该带插入结点的尾指针指向当前的栈顶（由 top 的后继指针所指向）S->next=top，再调整栈顶指针 top 指向新的栈顶 S 即可（执行 top=S）。

【参考答案】C

3.4 栈和队列的应用

温馨提示，本考点在近些年的真题中，出现了：1、栈在括号匹配中的应用；2、栈在表达式求值中的应用；3、栈在递归中的应用；4、队列在层次遍历中的应用；5、队列在计算机系统中的应用（408 统考）。请同学们多关注这些方面的内容。

1. 为解决计算机主机与打印机之间速度不匹配问题，通常设置一个打印数据缓冲区，主机将要输出的数据依次写入该缓冲区，而打印机则依次从该缓冲区中取出数据。该缓冲区的逻辑结构应该是（　　）。

A. 栈　　B. 队列　　C. 树　　D. 图

【2009 年统考——第 1 题】

【考查内容】队列的应用。

【解析】在解决计算机主机与打印机之间速度不匹配问题时，通常设置一个打印数据缓冲区，主机将要输出的数据依次写入该缓冲区，而打印机则从该缓冲区中取出数据打印。该缓冲区是内存的一块区域，文件输入/输出和实现打印功能时，用于存储传输的数据，它的实现要用队列来组织。

【参考答案】B

2. 判断一个表达式中左右括号是否匹配，采用（　　）实现较为方便。

A. 线性表的顺序存储　　B. 队列

C. 线性表的链式存储　　D. 栈

【2012 年——华南理工大学】

【考查内容】栈可以用于括号匹配的特点。

【解析】判断一个表达式中左右括号是否匹配，采用栈来实现比较方便。在该方法下，每一次压栈的都是左括号，故而无需判断每次出栈时栈顶元素是不是左括号。

我们可以利用以下的算法思想来解决本题：输入一个算术表达式后，依次逐个扫描表达式字符。若

(1). 扫描遇到左括号，将其存入链栈中；

(2). 扫描遇到右括号，栈顶左括号出栈，并继续扫描；

(3). 扫描遇到其他字符，不做任何操作，继续下个字符扫描。

当扫描结束后，若栈为空，即表达式圆括号配对正确。反之，若栈不为空，则表达式的圆括号匹配不正确。

【参考答案】D

3. 假设栈初始为空，将中缀表达式 a/b+(c*d-e*f)/g 转换为等价后缀表达式的过程中，当扫描到 f 时，栈中的元素依次是（　　）。

A. +(*-　　B. +(-*　　C. /+(*-*　　D. /+-*

【2014 年统考——第 2 题】

【考查内容】利用堆栈来实现表达式的转换。

【解析】很多书对这类题目的解析很复杂，我们来一个简单的记忆：不能立即参与运算的操作符号压栈后不能立即出栈。具体而言，我们以本题为例。本题出栈和入栈过程如下：

(1). a/b 能够运算，“/”压栈之后遇到 d，再出栈；

(2). “+”后面的表达式不知道，所以该符号压栈；

(3). “（”入栈，只有扫描到右括号才能出栈；

(4). “c*d”中的“*”在扫描到“d”时出栈，“-”因为后面的表达式不明确，只有等“e*f”运算完返回之后才能出栈；

(5). “e*f”中，运算符“*”先入栈，遇到 f 之后再出栈。当扫描到 f 时，*仍然在栈中。

经过以上分析，当扫描到 f 时，栈中的元素依次是“+(-*”，故而选择 B 答案。

【参考答案】B

4. 下列关于栈和队列的说法中，正确的是（　　）。

A. 消除递归不一定需要使用栈

B. 对同一输入序列进行两组不同的合法入栈和出栈操作，所得到的输出序列也一定相同

C. 通常使用队列来处理函数和过程处理

D. 队列和栈是运算首先的线性表，只允许在表的两端进行运算

【2014—中国传媒大学】

【考查内容】栈和队列的应用。

【解析】不是所有的递归都需要使用栈，比如计算 n 的阶乘这种单向递归，直接用循环来代替递归就可以了。

如 3.1 节的第二小题，对于同样的输入序列，出栈的顺序不同，得到了多组不同的合法的输出序列。所以，B 选项不正确。

过程处理一般使用堆栈结构，而不是队列。所以，C 选项不正确。

D 选项中，栈和队列虽然都是操作受限的线性表，但是栈只允许在栈顶进行操作，而不是两端都可以进行运算。所以，D 选项也错误。

【参考答案】A

3.5 特殊矩阵的压缩存储

温馨提示：特殊矩阵的压缩存储，考查内容经典总结如下：1、什么是特殊矩阵？特殊矩阵包括哪几种？（对称矩阵、三角矩阵和三对角矩阵等）2、稀疏矩阵有什么特点？3、特殊矩阵怎么存储？4、怎么计算矩阵中某一个元素转换成一维数组之后的下标？请同学们务必掌握这几种题型。

1. 稀疏矩阵一般的压缩存储方法有两种，即（　　）。

A. 二维数组和三维数组　　B. 三元组和散列

C. 三元组和十字链表　　D. 散列和十字链表

【2013 年——沈阳农业大学】

【考查内容】常见的两种稀疏矩阵压缩存储方式。

【解析】一般来说，稀疏矩阵非 0 元素的分布是没有规律的。因此，在存储非 0 元素的同时，还需要存储适当的辅助信息。稀疏矩阵的常见压缩存储方式有三元组表和十字链表两种方式，这两种方式在本考点接下来的考题中我们会详细讲解，这里先不解释。

【参考答案】C

2. 若将 n 阶三对角矩阵 A 按照行序为主序方式将所有非零元素依次存放在一个一维数组 B 中,则该三对角矩阵在 B 中至少占用了（　　）个数组元素。

A. n^2　　B. 3n+2　　C. 3n-2　　D. 3n

【2010 年——中山大学】

【考查内容】三对角矩阵转存成一维数组后的所占数组元素个数。

【解析】三对角矩阵除了第一行和最后一行，每一行都只有 3 个元素，此性质需要我们牢记。显然，对于 n 阶三对角矩阵，除了第一行和最后一行分别有 2 个元素之外，其他 n-2 行每行都有三个元素。

设行下标为 i，列下标为 j，三对角矩阵除满足条件 i=1，j=1、2，或$i = n, j = n - 1$、n或$2 < i < n - 1$，$j = i - 1$、i、$i + 1$的元素$a_{i,j}$之外，其余元素为 0。如下的矩阵便是三角矩阵

$$\begin{bmatrix} 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \cdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

对于这种矩阵，若按行优先方式来存储，除第 1 行和第 n 行有两个元素之外，其他每一行都有 3 个非零元素，因此共需存储元素个数 2+2+3*(n-2)=3n-2 个。

【参考答案】C

3. 假设以行序为主序存储二维数组 A=array[1.100,1…100]，设每个数据元素占 2 个存储单元，基地址为 10，则 LOC[5,5]=（　　）。

A. 808　　B. 818　　C. 1010　　D. 1020

【2011 年——太原科技大学】

【考查内容】二维数组的地址计算方法。

【解析】我们仍然复杂的问题简单地分析，数组 A[100][100]以 10 为基地址，A[5][5]元素所在位置的前面有 4 行，左边有 4 列。因为每行有 100 个元素，所以元素 A[5][5]的前面总共有元素$4 \times 100 + 4 = 404$个。因为每一个数据元素占 2 个存储单元，所以

$$LOC[5,5] = 404 \times 2 + 10 = 818$$

【参考答案】B

4. 对稀疏矩阵进行压缩存储的目的是（　　）。

A. 便于进行矩阵运算　　B. 便于输入和输出

C. 节省存储空间　　D. 降低运算的时间复杂度

【2013 年——江苏大学】

【考查内容】对矩阵压缩存储的目的。

【解析】**特殊矩阵**是指具有许多相同矩阵元素或零元素，并且这些相同矩阵元素或零元素的分布有一定规律性的矩阵。最常见的特殊矩阵有对称矩阵、上(下)三角矩阵、对角矩阵等。 而稀疏矩阵则指矩阵元素个数 s 相对于矩阵中非零元素个数 t 来说非常大，即 s>>t 的矩阵。至于稀疏到什么程度才能算稀疏矩阵，有的书把稀疏因为<0.05 的矩阵称为稀疏矩阵。稀疏矩阵要是按照一般存储方式来存储十分浪费存储空间，稀疏矩阵的压缩存储，正是为了节省存储空间。

【参考答案】C

5. 设有二维数组 A[1..12,1..10]，其每个元素占 4 个字节，数据按行优先顺序存储，第一个元素的存储地址为 100，那么元素 A[5,5]的存储地址为（　　）。

A. 76

B. 176

C. 276

D. 376

【2014 年—武汉科技大学】

【考查内容】二维数组转一维数组。

【解析】二维数组 A 有 12 行，10 列。按照行序优先顺序存储，元素 A[5,5]的前面有 4 行，每行 10 个元素，共 40 个元素。在第 5 行，该元素之前还有 4 个元素。所以元素 A[5,5]的前面总共有 44 个元素，需占用 44*4=176 个字节的存储空间。而数组的首地址为 100，所以 A[5,5]的存储地址为 100+176=276。

【参考答案】C

6. 已知有一维数组 A[0…m*n-1]，若要对应 m 行、n 列的矩阵，将元素 A[k](0≤ k ≤ m * n)表示成矩阵的第 i 行、第 j 列的元素（0≤ i < m,0 ≤ j < n）,则下面的对应关系是（　　）。

A. i=k/n, j=k%m

B. i=k/m, j=k%m

C. i=k/n, j=k%n

D. i=k/m, j=k%n

【2014—中国传媒大学】

【考查内容】一维数组转二维数组。

【解析】一维数组转换成二维数组，通常以行序为先。即，先第一行，再第二行,…,最后到第 m 行。应为每一行有 n 个元素，所以 k 所在的行应该是 i=k/n。对应的 j=k%n。

举一个简单的例子，m=10,n=5,k=11。一维数组的元素 A[0]、A[1]、A[2]、A[3]、A[4] 转入二维数组的第一行，A[5]~ A[9]转入二维数组的第二行。所以 A[11]应该是第三行的第二个元素 A[2][1]。

k%m=1,k/m=1,k%n=1,k/n=2。显然，C 答案成立。

【参考答案】C

7. 设有一个 10 阶的对称矩阵 A，采用压缩存储方式，以行序为主存储，$a_{1,1}$ 为第一个元素，其存储地址为 1，每个元素占一个地址空间，则 $a_{8,5}$ 的地址是（　　）。

A. 13

B. 33

C. 18

D. 40

【2014—中国传媒大学】

【考查内容】对称矩阵的压缩存储。

【解析】对称矩阵只需要存储上半部分或者下半部分（加上对角线元素）就可以了。假设我们存储上半部分加上对角元素，A[8][5]=A[5][8]，则第一行到第 4 行分别需要存储元素 10、9、8、7 个。在第 5 行，A[5][8]前面还有 7 个元素。所以，A[8][5]的数组下标为 10+9+8+7+7=40。

【参考答案】D

第 4 章　树与二叉树

4.1　树的概念

温馨提示：本考点主要考查树的定义、树的性质。请同学们了解树每一层的结点分布情况，以及树的叶子结点数目计算方法。

1. 在一棵度为 4 的树 T 中，若有 20 个度为 4 的结点，10 个度为 3 的结点，1 个度为 2 的结点，10 个度为 1 的结点，则树 T 的叶结点个数是（　　）。

A. 41　　B. 82　　C. 113　　D. 122

【2010 年统考——第 5 题】

【考查内容】树的基本概念，树的结点计算方法。

【解析】本题画图或者计算，都不太容易。我们可以这样来推理：这棵度为 4 的树 T 中，若有 20 个度为 4 的结点，10 个度为 3 的结点，1 个度为 2 的结点，10 个度为 1 的结点。所以，树的度数为$20 \times 4 + 10 \times 3 + 1 \times 2 + 10 \times 1 = 122$。因为根结点没有父结点，所以，所以，树中一共有结点 122+1=123 个。除去这 20+10+1+10=41 个结点，其他的 82 个结点是叶子结点。

注意，这里不用再出去父结点了。因为父结点的度肯定不为零，即不是叶子结点。也就是说，父结点的度数一定是 1,2,3,4 当中的一个，已经在题目中所述的结点范围之内了。

【参考答案】B

2. 一棵完全二叉树上有 1000 个结点，其中叶子结点的个数是（　　）。

A. 489　　B. 500　　C. 254　　D. 512

【2010 年——中山大学】

【考查内容】二叉树的基本概念及其应用。

【解析】完全二叉树总共有 1000 个结点，在第 i-1 层为满的情况下，第 i 层才可能有结点。第 1、2、3、4、5、6、7、8、9 层各有结点 1、2、4、8、16、32、128、256 个结点，总共 1+2+4+8+16+32+64+128+256=511 个。在第 10 层，还有 1000-511=489 个叶子结点，这些结点需要 245 个第 9 层的结点作为父结点。故而，第 9 层还有 256-245=11 个结点是叶子结点。

故而，总共有叶子结点 11+489=500 个。

【参考答案】B

3. 若一棵高度为 6 的完全二叉树的第 6 层有 3 个叶子结点，则该二叉树一共有（　　）个叶子结点。

A. 17　　B. 18　　C. 19　　D. 20

【2013 年——黑龙江大学】

【考查内容】二叉树中叶子结点的计算方法。

【解析】和第二题的推导一样，该二叉树第 1、2、3、4、5 层分别有结点 1、2、4、8、16 共计 31 个结点。在第 6 层有 3 个叶子结点，需要第 5 层的 2 个结点作为父结点，故而第 5 层有叶子结点 16-2=14 个。

综上，该二叉树总共有叶子结点 14+3=17 个。

【参考答案】A

4.2　二叉树

温馨提示：本考点是考研的重点内容，常以选择题、作图题的形式出现，请同学们多练习。本考点涉及到二叉树的遍历、线索化二叉树、完全二叉树、二叉排序树、平衡二叉树（其中二叉排序树和平衡二叉树我们在考点 4.4 中讲解）等，并把二叉树的顺序存储结构和链式存储结构也归纳到这部分。所以，内容较多，多花点时间，争取多学习一些。

1. 给定二叉树如图 4.1 所示。设 N 代表二叉树的根，L 代表根结点的左子树，R 代表根结点的右子树。若遍历后的结点序列是 3，1，7，5，6，2，4，则其遍历方式是（　　）。

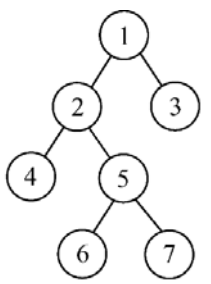


图 4.1

A. LRN　　B. NRL　　C. RLN　　D. RNL

【2009 年统考——第 3 题】

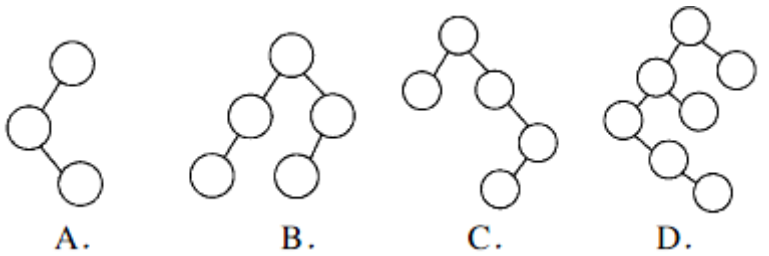
【考查内容】二叉树的 RNL 遍历方式。

【解析】我们常见的遍历算法有前序遍历、中序遍历和后序遍历，这里的“前”、“中”、“后”表示的都是二叉树中根的遍历顺序，左子树和右子树的顺序在遍历过程中不发生变化。

本题不同于我们常见的这三种遍历。首先，本题的遍历也是一个递归的算法，但是遍历的顺序都是右子树、根节点、左子树。所以其遍历方式应该是 RNL。

【参考答案】D

2. 下列二叉排序树中，满足平衡二叉树定义的是（　　）。



【2009 年统考——第 4 题】

【考查内容】平衡二叉树的应用。

【解析】平衡二叉树或为空树，或为如下性质的二叉排序树：

(1). 左右子树深度之差的绝对值不超过 1；

(2). 左右子树仍然为平衡二叉树。

我们定义平衡因子 BF=左子树深度－右子树深度，平衡二叉树每个结点的平衡因子只能是 1，0，-1。若结点平衡因子的绝对值超过 1，则该二叉排序树就是不平衡的。

显然，A 答案的二叉树根节点的左子树深度为 2，右子树深度为 0，不是平衡二叉树。C 答案对应的二叉树根节点的左子树深度为 1，右子树深度为 3，显然也不是平衡二叉树。D 答案对应的二叉树根节点的左子树深度为 4，右子树深度为 1，显然也不是平衡二叉树。B 答案每一个节点都满足平衡二叉树的定义，所以是平衡二叉树。

【参考答案】B

3. 已知一棵完全二叉树的第 6 层（设根为第 1 层）有 8 个叶结点，则该完全二叉树的结点个数最多是（　　）。

A. 39　　B. 52　　C. 111　　D. 119

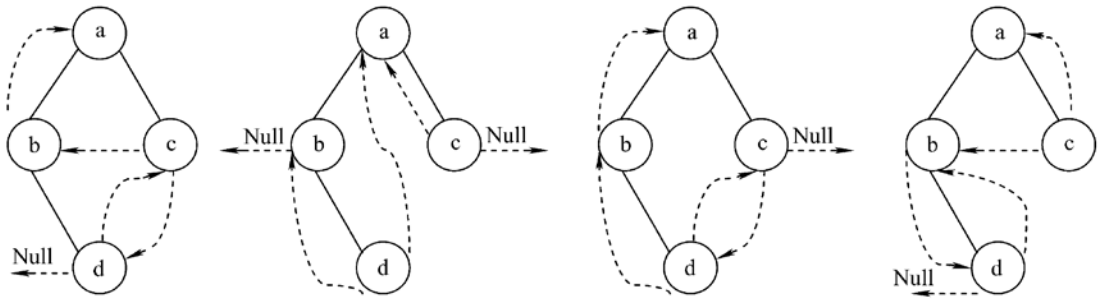
【2009 年统考——第 5 题】

【考查内容】完全二叉树的结点个数计算方法。

【解析】对于完全二叉树而言，若有 7 层，则第一层到第 6 层的结点应该全部都是满的。第一层 1 个结点、第 2 层 2 个结点、第三层 4 个结点、第 4 层 8 个结点、第五层 16 个结点、第 6 层 32 个结点。因为第 6 层的 32 个结点中有 8 个是叶子结点，所以有 24 个非叶子节点。该完全二叉树结点最多的情况是，这 24 个结点都分别有左右孩子时，整棵完全二叉树的节点数目最多，即总共有 1+2+4+8+16+32+48=111。

【参考答案】C

4. 下图 4.2 线索二叉树中（用虚线表示线索），符合后序线索树定义的是（　　）。



A　　B　　C　　D

图 4.2

[2010 年统考——第 3 题]

【考查内容】二叉树的后序线索化。

【解析】对于某一种遍历顺序对应的线索化，只需写出对应的遍历序列，然后修改空指针域分别指向该遍历序列的前驱和后继即可。例如，本题中的二叉树的后序遍历可得到序列 d、b、c、a。那么，d 是第一个元素，没有前驱，所以其左指针域原来为空，线索化时亦为空；a 是最后一个元素，但是其左右孩子都不为空，所以不需要考虑该结点的线索化；其余的 b 和 c 结点都各有一个前驱结点和后继结点。

那么，将 d 右指针域（初始为空）调整并指向其后继结点 b。将 b 结点的左指针域调整指向其前驱结点 d，因为 b 的右指针域不为空，所以线索化过程中不需要调整。c 的左右指针域都为空，令其左指针域指向其前驱结点 b，右指针域指向其后继结点 a。

【参考答案】D

5. 在图 4.3 所示的平衡二叉树中，插入关键字 48 后得到一棵新平衡二叉树。在新平衡二叉树中，关键字 37 所在结点的左、右子结点中保存的关键字分别是（　　）。

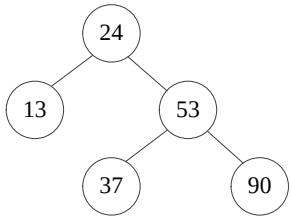


图 4.3

A. 13，48　　B. 24，48

C. 24，53　　D. 24，90

【2010 年统考——第 4 题】

【考查内容】平衡二叉树的调整。

【解析】我们曾在《2016 年考研核心考点命题思路解密》一书中，用我们独到的理解，总结了平衡二叉树树的旋转规则，并总结了失去平衡后的结点旋转校验规则：旋转后的平

衡二叉树中序遍历应该是一个递增的序列（因为平衡二叉树是二叉排序树的一种），若不满足该条件，则旋转可能出现错误。

就本题而言，我们在原来的平衡二叉树中插入一个新的结点 48，如图 4.4 所示。

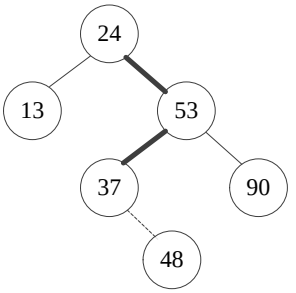


图 4.4

插入新的结点 48 之后，我们沿着叶子结点 48 往根节点的路径上，查找第一个不平衡的结点。显然，24 是第一个不平衡的结点，其左子树的高度为 1，右子树的高度为 3。图中的粗线部分，即需要旋转的部分。

那么，造成了什么类型的不平衡呢？我们这样判断：53 是 24 右孩子（R），37 是 24 的左孩子（L），所以是 RL 型不平衡。于是，先把结点 24 的右孩子 53 的左孩子 37 向右上方旋转到原来 53 的位置，再将 37 向左上旋转到 24 的位置，24 被旋转下来。成了如图 4.5 所示的样子。

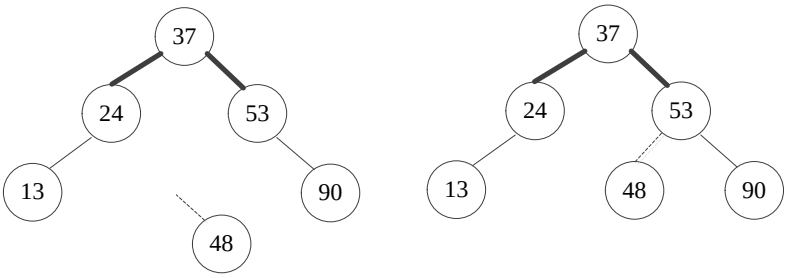


图 4.5

如图 4.5 所示，刚好还有一个结点 48，需要挂成某一个结点的左/右子树。到底挂哪儿呢？按照平衡二叉树中序遍历得到递增序列的原则，我们将结点 48 挂在结点 53 左子树上。

【参考答案】C

6. 若一棵二叉树的前序遍历序列和后序遍历序列分别为 1，2，3，4 和 4，3，2，1，则该二叉树的中序遍历序列不会是（　　）。

A. 1，2，3，4　　B. 2，3，4，1

C. 3，2，4，1　　D. 4，3，2，1

【2011 年统考——第 5 题】

【考查内容】二叉树的遍历。

【解析】已知二叉树的前序遍历序列和后序遍历序列，不能唯一确定一棵二叉树。只有在已知前序遍历序列或者后序遍历序列的情况下，又知道中序遍历序列，才能唯一确定一棵二叉树。

遍历一棵二叉树，要使得前序遍历序列和后序遍历序列刚好相反，那么必须保证每一个结点都只有一个孩子结点。故而，二叉树的高度为 4。那么，在前序遍历序列为 1、2、3、4，后序遍历序列为 4、3、2、1 的情况下，该二叉树第 1、2、3、4 层的结点依次为 1、2、3、4。显然，A、D 答案可对应图 4.6 所示的二叉树。

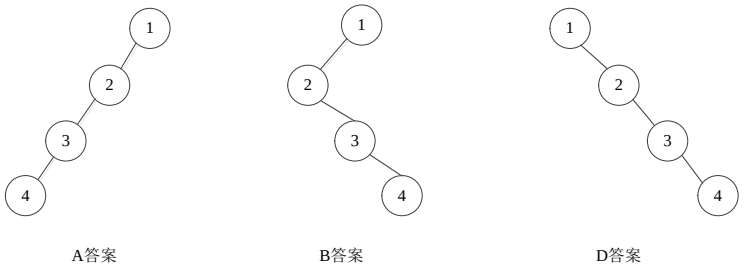


图 4.6

【参考答案】C

7. 下面关于二叉树的叙述，正确的是（　　）。

A. 完全二叉树的高度 h 与其结点数 n 之间存在确定的关系

B. 二叉树的顺序存储和链式存储结构中，完全二叉树更适合采用链式存储结构

C. 完全二叉树中一定不存在度为 1 的结点

D. 完全二叉树中必定有偶数个叶子结点

【2010 年——中山大学】

【考查内容】完全二叉树的概念、存储。

【解析】我们先分析比较简单的选项，再分析比较难的选项。完全 2 叉树中也可以存在度为 1 的结点，而且叶子结点不一定是偶数个（图所示的二叉树就存在度为 1 的结点，而且叶子结点的个数是奇数个）。如图 4.7 所示。

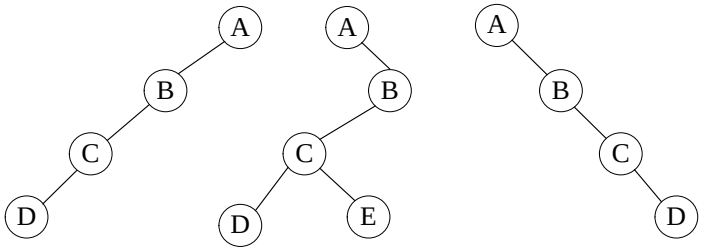


图 4.7

完全二叉树不同于普通的二叉树。完全二叉树可以采用顺序存储方法来存储，从左到右、从上到下依次存储第 1 层到最后一层的结点。完全二叉树采用顺序存储有很多好处，比如节省存储空间、查找方便（如第 i 个结点若存在左孩子，其左孩子的数组下标为 2i，若存在右孩子，则右孩子结点的编号为 2i+1）等。

完全二叉树中，有第 i+1 层的前提是第 1~i 层的结点都是满的。其高度 h 与其结点数 n 之间的关系满足：$2^{h-1}<=n<2^{h}$。

【参考答案】A

8. 设高度为 h 的二叉树上，只有度为 0 和度为 2 的结点，则这一类二叉树中所包含的结点个数至少是（　　）。

A. h-1　　B. h　　C. 2h-1　　D. 2h

【2011 年——南京邮电大学】

【考查内容】只有度为 0 和 2 的结点的二叉树的结点个数计算方法。

【解析】设叶子结点的个数为N_0，度数为 2 的结点个数为N_2，那么

$$N_0 = 1 + N_2$$

Balabala...

这么分析是不是又太复杂了？还是复杂的问题简单分析，因为二叉树中只有度为 0 和 2 的结点，当二叉树中结点个数最少时，我们可以先来一个左分支，在第 1 层到底 h-1 层的结点都有左右孩子，如图 4.8 所示。

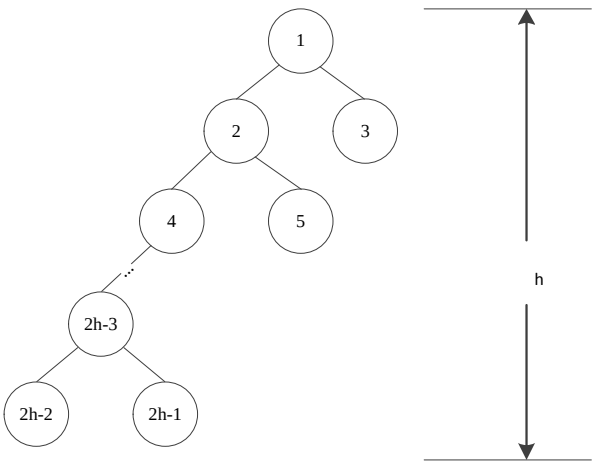


图 4.8

如图 4.8 所示的二叉树中，第 1 层只有一个结点，在第 2~h 层每层有两个结点。所以，满足题目已知条件的二叉树，结点最少时应该是 2（h-1）+1=2h-1 个。

【参考答案】C

9. n 个结点的线索二叉树上含有的线索数为（　　）。

A. 2n　　B. n-1　　C. n+1　　D. n

【2012 年——青岛理工大学】

【考查内容】线索二叉树的线索数目。

【解析】对于 n 个结点的二叉树，总共有指针域 2n 个，每一个结点都分别有一个左指针域和右指针域。这 n 个结点，除了根节点之外，每一个结点都有且仅有父节点的一个指针指向它。那么，n-1 个结点总共需要 n-1 个指针。其余的 n+1 个指针域为空，在线索化二叉树时用来存放指向其某种遍历顺序下的前驱结点或者后继结点。

【参考答案】C

10. 若使用二叉链表作为树的存储结构，在有 n 个结点的二叉链表中非空的链域的个数为（　　）。

A. n-1
B. 2n-1
C. n+1
D. 2n+1

【2015 年—暨南大学】

【考查内容】树的二叉链表存储结构。

【解析】树的二叉链表存储结构中，某一个结点的左（右）链域非空表示该结点有一个左（右）孩子结点。显然，有 n 个结点的二叉树除了根结点之外，其他结点都有一个父结点的指针指向它。所以，二叉树中共有非空的链域共 n-1 个，空的链域 n+1 个。

【参考答案】A

11. 一棵非空二叉树的先序遍历和后序遍历序列正好相反，则该二叉树一定（　　）。

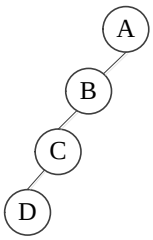
A. 所有的结点均无左孩子
B. 所有的结点均无右孩子
C. 只有一个叶子结点
D. 是任意一棵二叉树

【2014 年—河北大学】

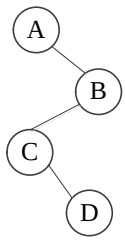
【考查内容】二叉树的遍历。

【解析】从**二叉树的遍历性质可知，二叉树的叶子结点在遍历的过程中，相对位置是不变的**。这个结论相当的重要。这也说明了，如果存在两个叶子结点，那么很难保证先序遍历和后序遍历得到的序列正好相反。所以，D 答案显然是不正确的。但是，C 答案说的只有一个叶子结点是前提，是不是还需要其他的条件呢？

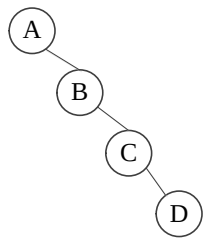
我们画了下面几种二叉树，来对比一下剩下的 A、B、C 三个选项。



图（1）



图（2）



图（3）

图 4.9

如图 4.9 所示，图（1）和（3）分别对应 A、B 两个选项。显然，对图（1）~（3）的二叉树进行前序遍历都能够得到序列 ABCD,对这三棵二叉树进行后序遍历都能够得到序列 DCBA，所以 A 和 B 选项是错误的。实际上，对于只有一个叶子结点的二叉树，前序和后序遍历所得到的序列是一样的。

12. 已知一棵二叉树的先序序列为 ABDGCFK，中序序列为 DGBAFCK，则后序序列为（ ）。

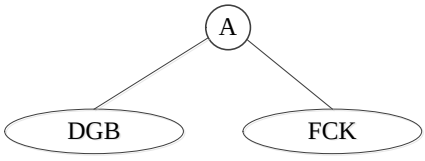
A. ACFKDBG
B. GDBFKCA
C. KCFAGDB
D. ABCDFKG

【2014 年—武汉科技大学】

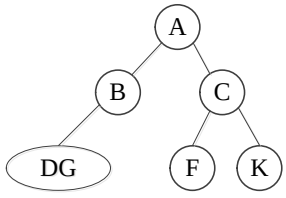
【考查内容】二叉树的遍历。

【解析】依题意，我们来重建二叉树。重建的过程如下：

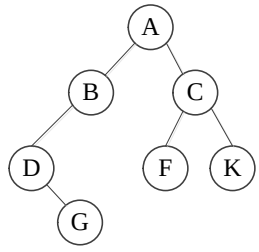
(1). 二叉树的先序序列为 ABDGCFK，所以 A 是二叉树的根结点。因为该二叉树的中序遍历序列为 DGBAFCK，所以左子树上的结点为 BDG，右子树上的结点为 FCK（如图 4.10（1）所示）。



图（1）



图（2）



图（3）

图 4.10

(2). 左子树前序遍历为 BDG，中序遍历为 DGB，显然，B 是左子树的树根，DG 两个结点都在 B 的左子树上。而 A 的右子树的先序遍历为 CFK，中序为 FCK，所以 C 是右子树的树根，F 是 C 的左孩子结点，K 是 C 的右孩子结点（如图 4.10（2）所示）。

(3). B 的左子树上的结点 D 和 G 的中序遍历是 DG，前序遍历也是 DG，说明 D 是 B 的左子树的树根，G 在 D 的右子树上（如图 4.10（3）所示）。

通过以上分析，得到了如图 4.10（3）所示的二叉树。对该二叉树进行后序遍历，可以得到序列 GDBFKCA。所以，选择 B 答案。

【参考答案】B

13. 在二叉树结点的先序，中序和后序序列中，所有叶子结点的先后顺序（　）。

A. 都不相同

B. 完全相同

C. 先序和中序相同，而与后序不同

D. 中序和后序相同，而与先序不同

【2014 年—武汉科技大学】

【考查内容】二叉树的遍历。

【解析】关于二叉树为什么在先序和后序遍历中，孩子结点的顺序不变，我们在《2016 年考研核心考点命题思路解密—数据结构》一书中已有了多次的解释。其实不管是前序、中序还是后序，这里的“前”、“中”、“后”描述的都是树根在遍历中相对于左右子树的遍历顺序，子树的先后顺序并没有发生变化。到了叶子结点，就体现在叶子结点的先后顺序没有发生改变。

【参考答案】B

14. 若从二叉树的任一结点出发到根的路径上所经过的结点序列按其关键字有序，则该二叉树是（　　）。

A. 二叉排序树

B. 平衡二叉树

C. 完全二叉树

D. 堆

【2014 年—河北大学】

【考查内容】堆的特点。

【解析】显然，题目描述的堆，满足堆的定义。大（小）顶堆从任意结点出发到根结点（堆顶）的路径上的结点递增（减）有序。二叉排序树的特点是中序遍历递增，每一个结点的关键字大于其左子树上所有结点的关键字，小于其右子树上所有结点的关键字。所以，不满足题意。

平衡二叉树是一种特殊的二叉排序树，显然也不满足题意。完全二叉树只能保证一棵二叉树中，一个结点有右孩子，必然有左孩子，不能保证有序。

【参考答案】D

15. 假设二叉树 T=<T_L,root, T_R>中结点数的定义如下：

$$\text{Nodes(T)} = \begin{cases} 0 & T\text{是空树} \\ 1 + Nodes(T_L) + Nodes(T_R) & \text{其他} \end{cases}$$

已知二叉树的结点定义如下：

```
typedef struct _BinNode{
    int key;
    struct _BinNode *LChild;*RChild;
}BinNode;
```

(1). 函数 Nodes(root)是求以结点 root 为根的二叉树的结点数。

```
int Nodes(BinNode *root)
{
if(_____) return 0;
return (_____);
}
```

(2). 函数 InOder(root)是中序遍历以结点 root 为根的二叉树。

```
void InOrder(BinNode *root)
{
if (_____){
    _____;
    printf("Key:%d\n",root->key);
    _____;
    }
}
```

【2013 年—中山大学】

【考查内容】本题考查十分基本的二叉树操作。

【解析】

(1). 这一问考查了递归求二叉树的结点个数的方法。其实递归的算法题目已经给出了，我们直接写出来就可以了。

① 第一个空，根据题目给出的数学表达式，首先判断是不是空的二叉树，所以填写 root==NULL。若二叉树为空，则返回 0，表示结点的个数为 0。

② 接下来，递归遍历二叉树的左右子树，统计左右子树的结点个数。这些结点个数需要加上根结点，所以结点个数总共是 1+Nodes(root->LChild)+Nodes(root->RChild)。

(2). 这一问考查了对二叉树的中序遍历。考查基本的遍历，请同学们要会写才行。中序其实和前序、后序的算法相差不大的（但是结果相差很大）。

① 第一空，当然是看看是不是空的二叉树了。所以填 root!=NULL。

② 如果不是空的二叉树，则先遍历左子树，输出左子树上的结点，所以填 InOrder(root->LChild)，输出了左子树的结点，然后再输出根结点。

③ 最后，输出右子树上的结点，所以填 InOrder(root->RChild)。

4.3 树、森林

温馨提示：本考点主要考查：1、树的存储结构，包括双亲表示法、孩子表示法和孩子兄弟表示法；2、树、森林与二叉树之间的相互转换；3、树和森林的遍历；4、树的应用，并查集（408 统考）。解题方法相对固定，请同学们熟悉解题方法。

1. 将森林转换为对应的二叉树，若在二叉树中，结点 u 是结点 v 的父结点的父结点，则在原来的森林中，u 和 v 可能具有的关系是（　　）。

Ⅰ．父子关系

Ⅱ．兄弟关系

Ⅲ．u 的父结点与 v 的父结点是兄弟关系

A. 只有Ⅱ　　B. Ⅰ和Ⅱ

C. Ⅰ和Ⅲ　　D. Ⅰ、Ⅱ和Ⅲ

【2009 年统考——第 6 题】

【考查内容】森林转换成二叉树。

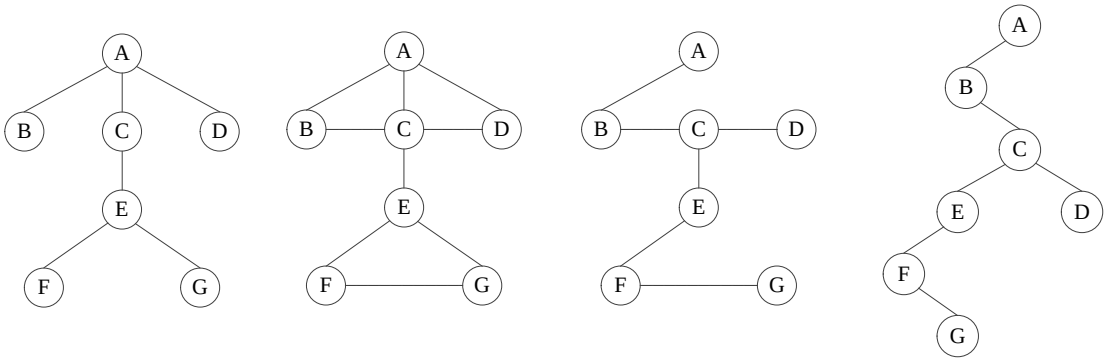
【解析】将一般树转化为二叉树的思路，主要根据树的孩子-兄弟存储方式而来，步骤是：

(1). 加线：即在各兄弟结点之间用虚线相连，可理解为每个结点的兄弟指针指向它的一个兄弟。

(2). 抹线：对每个结点仅保留它与其最左一个孩子的连线，抹去该结点与其他孩子之间的连线（可理解为每个结点仅有一个孩子指针，让它指向自己的长子）。

(3). 旋转：把虚线改为实线从水平方向向下旋转成右斜下方向，原树中实线成左斜下方向。这样，树就转换成二叉树了。

例如，我们将图(1)的森林转换成二叉树。以下图（2）、（3）、（4）分别对应以上步骤。树转换得到了如图 4.11 所示的二叉树。



图（1）　　图（2）　　图（3）　　图（4）

图 4.11

上图(4)中，D 是 B 的父节点的父节点，在对应的森林中，D 是 B 的兄弟结点，所以 II 正确。A 是 C 的父节点的父节点，但是 A 是 C 的父亲结点，所以 I 正确。

D 答案描述不正确，u 的父结点与 v 的父结点是兄弟关系，则 u 和 v 分别在其父结点的左子树上，不可能有 u 是 v 的父结点的父结点。

【参考答案】B

2. 已知一棵有 2011 个结点的树，其叶结点个数为 116，该树对应的二叉树中无右孩子的结点个数是（　　）。

A. 115　　B. 116

C. 1895　　D. 1896

【2011 年统考——第 6 题】

【考查内容】树转换成二叉树。

【解析】我们可以设想这样一棵树：该树最后一层有 116 个叶子结点，其他的每层都只有一个结点。那么，除了最后一层，其他层总共有结点个数为 2011-116=1895。也就是说，这棵特殊的树共有 1896 层，除了最后一层，其他层每层都只有一个结点。

那么，将这样的一棵树转换成二叉树，则除了最后一层的 116-1=115（最后一层最右边的结点无右孩子结点）个结点之外，其他的结点都没有右孩子。所以，树对应的二叉树中无右孩子的结点个数是 2011-115=1896。

【参考答案】D

3. 将森林 F 转换为对应的二叉树 T，F 中叶结点的个数等于（　　）。

A. T 中叶结点的个数　　B. T 中度为 1 的结点个数

C. T 中左孩子指针为空的结点个数　　D. T 中右孩子指针为空的结点个数

【2014 年统考——第 5 题】

【考查内容】森林转换成二叉树。

【解析】我们给出了如图 4.10 所示的森林，该森林有三棵树，树根分别为 A、D、G。森林 F 中有叶子结点 B、C、F、H、K 共 5 个。

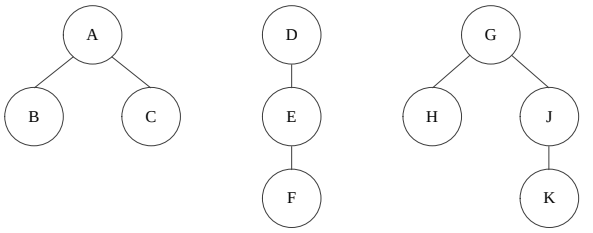


图 4.12

将图 4.12 的三棵树组成的森林，转换成图 4.13 所示的二叉树。

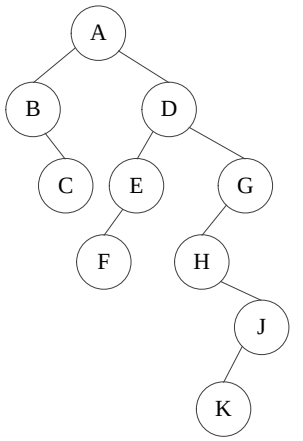


图 4.13

可见，转换后的二叉树中，共有 CFK 共 3 个叶子结点，度为 1 的结点 B、E、G、H、J 共 5 个，左孩子指针为空的结点 B、C、F、H、K 共 5 个，右孩子指针为空的结点 C、E、F、G、J、K 共 6 个。若森林中只有一棵树（如森林 F 中的第二棵树），该棵树有叶子结点 1 个，转换成 2 叉树后，度为 1 的结点有 2 个。

事实上，森林中的叶子结点没有孩子结点，在转换成二叉树时，该叶子结点也没有左孩子，所以左指针为空。故而，选择 C。

【参考答案】C

4.4 树的应用

温馨提示：树的应用部分，考查二叉排序树、平衡二叉树、哈弗曼树，哈弗曼编码。本考点是一个常见的命题点，其他的考查内容较少，我们围绕哈弗曼树给了几个考题，请同学们注意解题方法。

1. 对 n（n≥2）个权值均不相同的字符构造成哈夫曼树。下列关于该哈夫曼树的叙述中，错误的是（　　）。

A. 该树一定是一棵完全二叉树

B. 树中一定没有度为 1 的结点

C. 树中两个权值最小的结点一定是兄弟结点

D. 树中任一非叶结点的权值一定不小于下一层任一结点的权值

【2010 年统考——第 6 题】

【考查内容】哈夫曼树的特点。

【解析】哈夫曼树除了叶子结点之外，任何一个非叶子结点都是由两个下层结点合并而得，所以树中没有度为 1 的结点，B 答案正确。

因为 n 个字符的权值均不相同，所以两个权值最小的结点是权值最小的结点和权值次小的结点，具有唯一性，这两个结点一定是兄弟结点，所以 C 答案正确。

树中任一个分叶子结点的权值都是由其左右孩子结点的权值之和而得，所以权值应该大于其左右孩子的权值。若非叶子结点 G 的权值小于下一层的某一个结点 M 的权值，那么在上一轮选择两个最小的结点合并的时候，还轮不到该结点 M 合并，至少要先把结点 G 合并才对。所以，非叶子结点的权值一定不小于下一层的任何一个结点的权值。故而，D 答案正确。

但是，哈夫曼树不一定都是完全二叉树，哈夫曼树是完全二叉树属于一种特殊的情况。事实上，我们遇到的很多哈夫曼树都不是完全二叉树。

【参考答案】A

2. 下述编码中哪一个不是前缀码（　　）。

A. (00, 01, 10, 11)　　B. (1, 0, 00, 11)

C. (0, 10, 110, 111)　　D. (1, 01, 000, 001)

【2010 年——中山大学】

【考查内容】前缀码。

【解析】前缀编码确保任何一个字符的编码都不是同一字符集中另一个字符的编码的前缀。很明显，B 答案中 0 就是 00 的前缀码。

【经典总结】的前缀码比较容易观察和判别。要是再复杂一些，不太方便直接观察判断怎么办？最好的方法就是根据答案给的码子画哈夫曼树。若码子全部落在叶子结点且不出现重复，则该编码是前缀码。若出现其他的情况（如某一个码子的路线终点落在非叶子结点上）则该编码不是前缀码。

【参考答案】B

3. 已知某通信系统中可能出现九个字符：C、O、M、P、U、T、E、R、S，它们出现频率分别为：0.11、0.09、0.06、0.15、0.23、0.12、0.04、0.03、0.17，试利用它们作为叶子结点构造一棵 Huffman 树（哈夫曼树），并设计这些字符的哈夫曼编码。

【2013 年——上海海事大学】

【考查内容】哈弗曼编码。

【解析】题中已给出 8 种字符的出现频率，如表 4.1 所示。

表 4.1

字符	C	O	M	P	U	T	E	R	S
出现频率	0.11	0.09	0.06	0.15	0.23	0.12	0.04	0.03	0.17

下面，我们给出详细的哈夫曼树的构造过程：

(1). 从以上结点中，现在出现频率最小的字符 R(0.3)和 E(0.4)，合并成新的概率 0.7；

(2). 新表中有 8 个概率 0.11、0.09、0.06、0.15、0.23、0.12、0.07、0.17，选择两个最小的概率 0.06 和 0.07 合并成新的概率 0.13；

(3). 新表中有 7 个概率 0.11、0.09、0.15、0.23、0.12、0.13、0.17，选择两个最小的概率 0.09 和 0.11，合并成新的概率 0.20；

(4). 新表中有 6 个概率 0.15、0.23、0.12、0.13、0.20、0.17，选择两个最小的概率 0.12 和 0.13，合并成新的概率 0.25；

(5). 新表中有 5 个概率 0.15、0.23、0.25、0.20、0.17，选择两个最小的概率 0.15 和 0.17，合并成新的概率 0.32；

(6). 新表中有 4 个概率 0.32、0.23、0.25、0.20，选择两个最小的概率 0.20 和 0.23，合成新的概率 0.43；

(7). 新表中有 3 个元素 0.32、0.25 和 0.43，选择两个最小的概率 0.25 和 0.32，合并成新的概率 0.57；

(8). 新表中只有两个概率 0.43 和 0.57，合并成概率 1.00。

可构造哈夫曼树如图 4.14 所示。

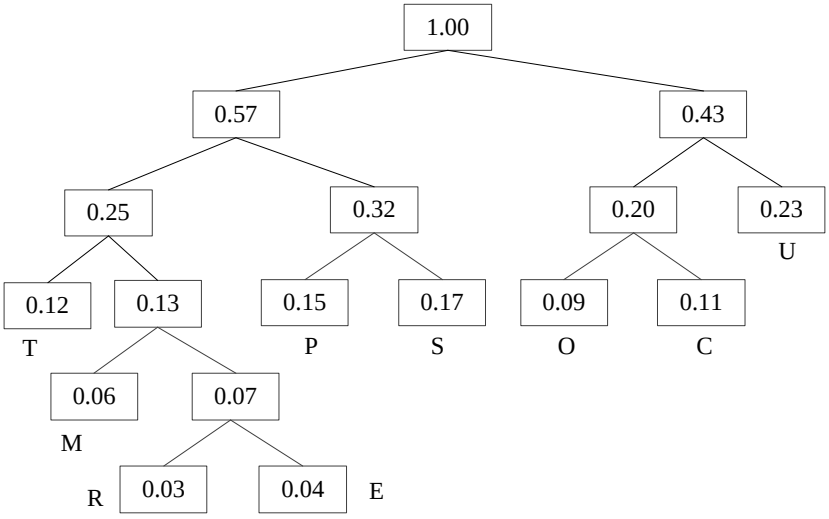


图 4.14

根据上图可知，每个字符的哈夫曼编码如表 4.2 所示。

表 4.2

字符	C	O	M	P	U
哈夫曼编码	101	100	0010	010	11
字符	T	E	R	S	
哈夫曼编码	000	00111	00110	011	

第 5 章　图

5.1　图的概念

温馨提示：本考点考查图的基本概念，属于基础知识，内容一般围绕图的基本特点、连通性等展开，请同学稍微花点功夫注意这方面的知识。可能恶心的是，零散的东西一旦命中，知识点杂乱，可能不好做。

1. 下列关于无向连通图特性的叙述中，正确的是（　　）。

Ⅰ．所有顶点的度之和为偶数

Ⅱ．边数大于顶点个数减 1

Ⅲ．至少有一个顶点的度为 1

A. 只有Ⅰ　　B. 只有Ⅱ　　C. Ⅰ和Ⅱ　　D. Ⅰ和Ⅲ

【2009 年统考——第 7 题】

【考查内容】无向连通图的特性。

【解析】无向连通图所有的顶点度数之和为边的两倍，显然该总和是一个偶数，I 正确。边数最小的情况下，可以是 n-1，所有 II 错误。无向连通图中，不一定有一个顶点的度数刚好是 1。比如由 4 个顶点 4 条边构成的类似于“四边形”图中，任意顶点的度数都是 2，无度为 1 的顶点。

【参考答案】A

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

A. 6　　B. 15　　C. 16　　D. 21

【2010 年统考——第 7 题】

【考查内容】无向连通图的最少边数。

【解析】为了保证所有顶点都是连通的，对于 7 个顶点的无向图而言，可以让其中的 6 个顶点中任意两个顶点之间都存在一条边，第 7 个结点刚好有一条边连接这 6 个顶点构成的“网络”中的一个顶点。需要的边数最少的情况为 5×6/2+1=16。

思考题：边数最少的连通图的情况，为什么是其中的 6 个顶点中任意两个顶点之间都存在一条边，第 7 个结点刚好有一条边连接这 6 个顶点构成的“网络”中的一个顶点？

【参考答案】C

3. 下列关于图的叙述中，正确的是（　　）。

Ⅰ．回路是简单路径

Ⅱ．存储稀疏图，用邻接矩阵比邻接表更省空间

Ⅲ．若有向图中存在拓扑序列，则该图不存在回路

A. 仅Ⅱ　　B. 仅Ⅰ、Ⅱ

C. 仅Ⅲ　　D. 仅Ⅰ、Ⅲ

【2011 年统考——第 8 题】

【考查内容】图的特点、存储和回路。

【解析】简单路径是指不存在回路的路径。所以，I 错误。稀疏图是边数比较少的图，对于稀疏图，采用邻接表比邻接矩阵更加节省存储空间，所以 II 错误。拓扑排序和深度优先遍历都可以判断图中是否存在着回路。若有向图存在拓扑排序，则该图不存在回路。

【参考答案】C

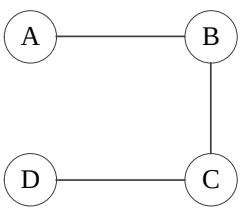
4. 要连通具有 n 个顶点的无向图，至少需要的边数是（　　）。

A. n　　B. n-1　　C. n+1　　D. 2n

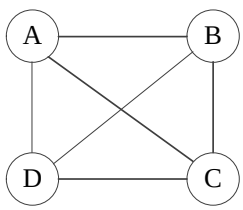
【2010 年——中山大学】

【考查内容】无向连通图的最少边数。

【解析】所谓的**连通图**，可以认为是任意两个结点之间都有通路的图。什么情况下，连通图的边数量最少呢？当然是把所有的结点都串联起来，而且还不构成圈（差一条边，如图 5.1(1)所示）的时候。对于 n 个结点的连通图，边数最少时为 n-1。



图（1）



图（2）

图 5.1

什么情况下，边数最多呢？既然连通图任意两个结点之间有边，那么边数最多时，任意两个结点之间都有一条边，我们称这样的图为无向完全图。如图 5.1(2)所示。对于有 n 个结点的连通图，边数最多时为 n(n-1)/2。

【参考答案】B

5.2　图的存储及基本操作

温馨提示：本考点主要考查图的邻接矩阵、邻接表表示，请同学们掌握这两种存储方式下，顶点的入度和出度的计算方法，以及某一种遍历下这两种存储方式得到的序列（我们将在考点 5.3 图的遍历部分给出考题）。这是本考点的两类主要题型。

1. 一个有向图，有 n 个顶点，e 条边，则对其邻接表以下说法正确的是（　　）。

A. 邻接表中有 n 个头结点和 2e 个表结点，求顶点的度很快

B. 邻接表中有 n 个头结点和 e 个表结点，求顶点的度要遍历整个邻接表

C. 邻接表中有 n 个头结点和 2e 个表结点，求顶点的度要遍历整个邻接表

D. 邻接表中有 n 个头结点和 e 个表结点，求顶点的度很快

【2013 年——华南理工大学】

【考查内容】图的邻接表表示。

【解析】对于一个具有 n 个顶点 e 条边的无向图，它的邻接表表示有 n 个顶点表结点 2e 个边表结点；对于一个具有 n 个顶点 e 条边的有向图，它的邻接表表示有 n 个顶点表结点 e 个边表结点。

在无向图中求顶点的度，使用邻接矩阵及邻接表法都很容易做到。邻接矩阵中第 i 行(或第 i 列)上非 0 元素的个数就是顶点 V_i 的度，在邻接表中第 i 个边表的结点个数就是顶点 V_i 的度。

在有向图中求顶点的度采用邻接矩阵比采用邻接表表示更方便。邻接矩阵中第 i 行非零元素的个数是顶点 V_i 的入度，邻接矩阵中第 i 列非零元素的个数是顶点 V_i 的出度，二者之和就是定点 V_i 的度数。

邻接表表示中第 i 个边表上的结点个数就是顶点 V_i 的出度,求入度较困难，需遍历个顶点的边表，逆邻接表表示中第 i 个边表上的结点个数就是顶点 V_i 的入度,求出度较困难，需遍历个顶点的边表。

【参考答案】B

2. 下列说法中错误的是（　　）。

A. 有向图的邻接矩阵不一定是对称矩阵

B. 无向图的邻接矩阵不一定是对称矩阵

C. 若图 G 的邻接矩阵是对称的，则 G 不一定是无向图

D. 若图 G 的邻接矩阵是对称的，则 G 不一定是有向图

【2014 年——中国计量学院】

【考查内容】有向图和无向图的邻接矩阵特点。

【解析】有向图的邻接矩阵不一定是对称矩阵，但是无向图的邻接矩阵却一定是对称矩阵。所以，A 答案的说法正确，B 答案的说法错误。反而言之，若图 G 的邻接矩阵是对称的，图 G 可能是有向图，也可能是无向图。

【参考答案】B

3. 设使用的邻接表表示某有向图，则顶点 v_j 在表结点中出现的次数等于（　　）。

A. 顶点 v_j 的度　　B. 顶点 v_j 的出度

C. 顶点 v_j 的入度　　D. 无法确定

【2013 年——暨南大学】

【考查内容】有向图的邻接表表示。

【解析】用邻接表来表示有向图，顶点 v_j 在表结点中出现的次数，表示 v_j 的入度。由 v_j 开头的链表的结点个数表示 v_j 的出度，其他各个链表中 v_j 结点出现的次数之和表示顶点 v_j 的入度。因为邻接表的表结点个数等于 v_j 的入度，所以选择 C 答案。

【参考答案】C

4. 采用邻接表存储的图的深度优先遍历算法类似于二叉树的（　　）。

A. 先序遍历　　B. 中序遍历　　C. 后序遍历　　D. 按层遍历

【2013 年——武汉理工大学】

【考查内容】图的邻接表表示下深度优先遍历的特点。

【解析】采用邻接表存储的图的深度优先遍历类似于二叉树的先序遍历，采用邻接表存储的图的宽度优先遍历，类似于二叉树的层次遍历。

【参考答案】A

5. 已知一个无向图的顶点集合为{a,b,c,d,f,g}，其邻接矩阵如下表示(0-无边，1-右边)。

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

(1). 画出该图形。

(2). 根据邻接矩阵从顶点 a 出发进行深度优先遍历（同一个结点的邻接结点按结点的字母序为序），画出相应的深度优先遍历树。

【2013—中山大学】

【解析】

(1). 依题意，该邻接矩阵对应的无向图如下图所示。

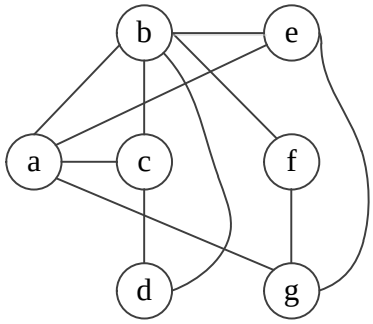
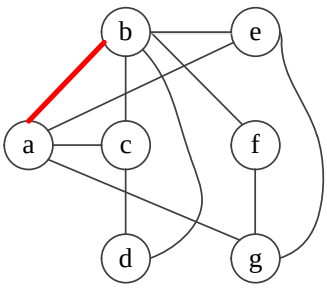
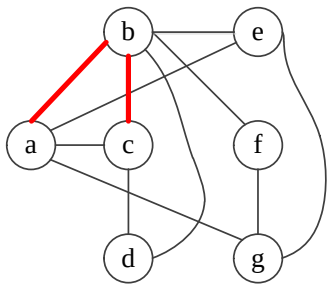


图 5.2

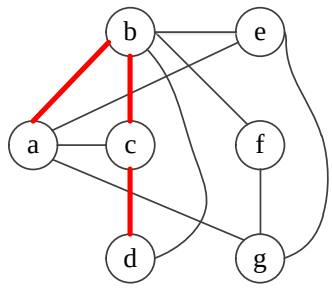
(2). 从顶点 a 开始，对图 5.2 进行深度优先遍历。题意告知，遍历的过程中若邻接点比较多，按照字母的顺序遍历。所以，首先遍历到的是 a 的邻接点 b，再接着遍历结点 b 的邻接点 c,再遍历结点 c 的邻接点 d（如图 5.3 所示）。



图（1）



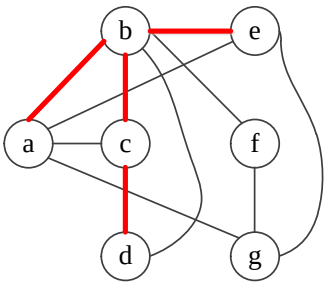
图（2）



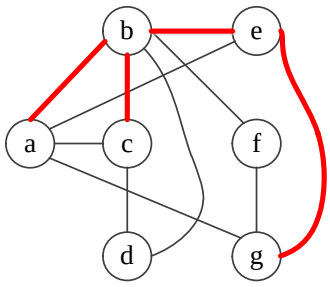
图（3）

图 5.3

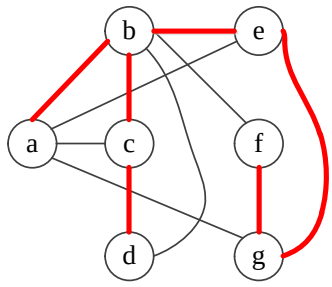
当遍历到结点 d 时，显然与结点 d 邻接的结点 a,b,c 均已经被访问过。所以，回溯到 d 的上层 c。但是，c 所有邻接结点也已经被访问过。接着回溯到结点 b。与 b 邻接的结点 e 和 f 均没有被访问过，所以访问结点 e，接着访问 e 的邻接点 g，接着访问 g 的邻接点 f。如图 5.4 所示。



图（4）



图（5）



图（6）

图 5.4

f 没有邻接点，于是回溯到上一层结点 g。g 的所以邻接点都已经被访问过，接着回溯到结点 e，e 的所有邻接点也都被访问过，接着回溯到 b，b 的所有邻接结点也都被访问过，所以回溯到 a。a 的所有邻接结点此时都已经被访问过。所以，遍历结束了。红线部分，就是题目要求的深度优先遍历树了。

注意：请同学注意递归遍历的过程。

5.3 图的遍历

温馨提示：本考点是 408 统考和高校自主命题中，常见的一类命题。主要考查两个方面的内容：1、遍历的复杂度；2、图根据某种优先遍历所得到的序列。其中，根据图的某种优先遍历所得到的关系，是一类常见的题型，请同学们务必掌握。

1. 对有 n 个结点、e 条边且使用邻接表存储的有向图进行广度优先遍历，其算法时间复杂度是（　　）。

A. O(n)　　B. O(e)　　C. O(n+e)　　D. O(n*e)

【2012 年统考——第 5 题】

【考查内容】有向图邻接表表示法的广度优先遍历的复杂度。

【解析】删除与某个顶点V相关的所有边，首先删除邻接表中下标为V的顶点表所对应的链表，因为有n个顶点，所有对于顶点V来说，出边最多的情况下有n-1条边，删除的复杂度为O（n）。

另外，还得删除其他顶点所对应的单链表中 V 顶点的入边，该删除复杂度为 O(e)。故而，删除与顶点 V 相关的所有边的时间复杂度为 O（n+e）。

【参考答案】C

2. 若对如图 5.5 所示的无向图进行遍历，则下列选项中，不是广度优先遍历序列的是（　　）。

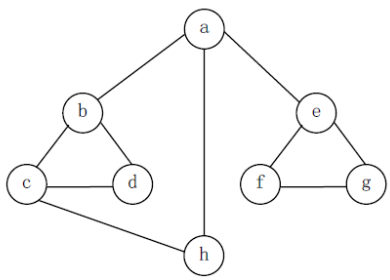


图 5.5

A. h，c，a，b，d，e，g，f　　B. e，a，f，g，b，h，c，d

C. d，b，c，a，h，e，f，g　　D. a，b，c，d，h，e，f，g

【2013 年统考——第 8 题】

【考查内容】无向图的广度优先遍历。

【解析】广度优先遍历算法我们在这里就不再细述了，都是一些基本的算法。下面我们来分析这 4 个选项。

A 答案中，从顶点 h 出发，发起广度优先遍历。与 h 邻接的顶点 c，a 依次入队列；c 出队列，与 c 邻接的顶点 b,d 依次入队列；a 出队列，与 a 邻接的顶点 e 入队列（已遍历过的顶点不再访问）；b 出队列，d 出队列；e 出队列，与 d 相邻的顶点 g 和 f 入队列；g、f 出队列。可以看出，A 答案正确的。

同理，可验证 B、C 选项也是正确的。

对于 D 答案，从顶点 a 开始，发起广度优先遍历，首先入队列的是 b 和 e，这两个顶点也应该在 a 出队列之后接着出队列的两个元素。而 D 答案中,c、d、h 都在 e 之前出队列，显然该选项不正确。事实上，D 答案所给的序列是该图的一个深度优先遍历所得到序列。

【参考答案】D

3. 对二叉树的结点从 1 开始进行连续编号，要求每个结点的编号大于其左、右孩子的编号，同一结点的左右孩子中，其左孩子的编号小于其右孩子的编号，可采用（　　）次序的遍历实现编号。

A. 先序　　B. 中序

C. 后序　　D. 从根开始按层次遍历

【2013 年——江苏大学】

【考查内容】二叉树的遍历。

【解析】对二叉树的结点从 1 开始进行连续编号，要求每个结点的编号大于其左、右孩子的编号，同一结点的左右孩子中，其左孩子的编号小于其右孩子的编号。显然后序遍历先遍历左子树、再遍历右子树，最后遍历根节点的算法性质，刚好可满足题目要求。

【参考答案】C

4. 下列说法不正确的是（　　）。

A. 图的深度遍历不适用于有向图

B. 图的遍历是从给定的顶点出发每一个顶点仅被访问一次

C. 图的遍历的基本算法有两种：深度遍历和广度遍历

D. 图的深度遍历是一个递归过程

【2013 年——江苏大学】

【考查内容】图的遍历。

【解析】有向图仍然可以使用深度优先算法或者广度优先算法进行遍历。所以，A 答案的说法错误。

【参考答案】A

5. 图的深度优先遍历类似于二叉树的（　　）。

A. 先序遍历

B. 中序遍历

C. 后序遍历

D. 层次遍历

【2014 年—武汉科技大学】

【解析】对图进行深度优先遍历，类似于考点 5.2 的第 5 题，从解析我们可以看出，图的深度优先遍历，类似于二叉树的先序遍历。

【参考答案】A

5.4 图的基本应用及其复杂度分析

温馨提示：图的基本应用部分，主要考查拓扑排序、关键路径、最短路径（Dijstra 算法、Floyd 算法）以及最小生成树（Prim 算法、Kruskal 算法）。这些内容十分重要，而且常出大题，是《数据结构》的核心考点，希望同学们能够掌握手工操作方法。我们已经将经典的例题献给大家，希望大家有所收获。

1. 对图 5.6 进行拓扑排序，可以得到不同的拓扑序列的个数是（　　）。

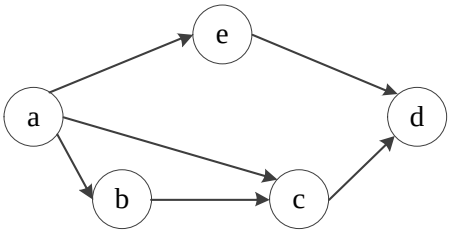


图 5.6

A. 4　　B. 3　　C. 2　　D. 1

【2010 年统考——第 8 题】

【考查内容】有向图的拓扑排序。

【解析】拓扑排序的步骤为：（1）在有向图中选一个没有前驱的顶点并且输出之；（2）从图中删除该顶点和所以以它为尾的弧。重复上述两步，直至全部顶点均已输出。由于没有前驱的顶点可能不唯一，所以拓扑排序的结果也不唯一。

我们利用以上算法思想对该图进行拓扑排序，过程如下：a 是第一个没有前驱的结点，输出该结点，并从图中删除由定点 a 发出的指向定点 b 和 e 的弧。b、e 都是没有前驱的结点，所以这两个结点都可以入队列。分以下两种情况来讨论：

(1). 假设 e 先入队列，删除 e 发出的指向 d 的边。接着找到没有前驱的结点 b 并输出，删除 b 结点发出的指向 c 的边。找到没有前驱的结点 c，输出 c 并删除边 cd。找到没有前驱的结点 d，输出 d。可得到序列 a、e、b、c、d。

(2). 如果 b 先入队列，也可以得到 a、b、c、e、d 和 a、b、e、c、d 两个序列。

【参考答案】C

2. 若用邻接矩阵存储有向图，矩阵中主对角线以下的元素均为零，则关于该图拓扑序列的结论是（　　）。

A. 存在，且唯一　　B. 存在，且不唯一

C. 存在，可能不唯一　　D. 无法确定是否存在

【2012 年统考——第 6 题】

【考查内容】有向图在邻接矩阵存储下的拓扑排序。

【解析】用邻接矩阵存储有向图，矩阵中主对角线以下的元素均为零，说明该有向图不存在顶点编号大的往顶点编号小的有向边，也即不存在回路。所以，该图的拓扑序列一定存在。

但是，仅有题目给出的条件，并不足以判断该图的拓扑序列是否唯一。

【参考答案】C

3. 对如图 5.7 所示的有向带权图，若采用迪杰斯特拉（Dijkstra）算法求从源点 a 到其他各顶点的最短路径，则得到的第一条最短路径的目标顶点是 b，第二条最短路径的目标顶点是 c，后续得到的其余各最短路径的目标顶点依次是（　　）。

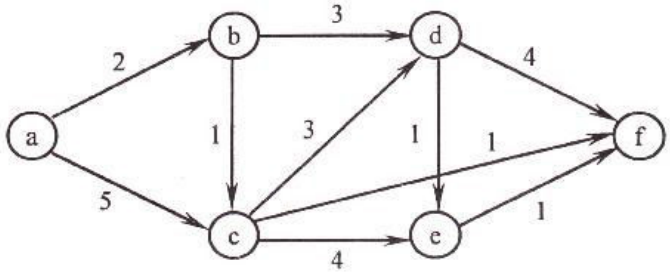


图 5.7

A. d,e,f　　B. e,d,f　　C. f,d,e　　D. f,e,d

【2012 年统考】

【考查内容】用 Dijkstra 算法求有向带权图的最短路径。

【解析】Dijkstra 算法求有向带权图的最短路径时，从起始顶点出发，逐步顺序地向外探索，每次向外延伸一步，而且保证当前路径是最短的。

具体而言，我们以本题为例。从 a 到各顶点的最短距离求解如表 5.1 所示。

表 5.1

顶点	第一次延伸	第二次延伸	第三次延伸	第四次延伸	第五次延伸
b	(a,b) 2				
c	(a,c) 5	(a,b,c) 3			
d	∞	(a,b,d) 5	(a,b,d) 5	(a,b,d) 5	
e	∞	∞	(a,b,c,f) 4		
f	∞	∞	(a,b,c,e) 7	(a,b,c,e) 7	(a,b,d,e) 6
集合 S	(a, b)	(a,b,c)	(a,b,c,f)	(a,b,d)	(a,b,c,f,d,e)

【参考答案】C

4. 设有一个 AOE 网，有 3 条关键路径，共有 15 个关键活动，下面的说法（　　）是正确的。

A. 提前完成这 15 个关键活动之外的活动可以缩短工期

B. 这三条关键路径长度相同

C. 提前完成这 3 条关键路径中的任何一个关键活动都能缩短工期

D. 改变这 15 个关键活动之外的活动不会影响工期

【2013 年——华南理工大学】

【考查内容】AOE 网。

【解析】AOE 网上有多条关键路径，这些关键路径的路径长度是一致的。所以，B 答案。提前完成非关键活动，不能缩短工期。除了修改这三条关键路径上的共同活动（假设存在），修改其他的某一个关键活动，不能缩短工期。

【参考答案】BD

5. 求单源最短路径的迪杰斯特拉算法的时间复杂度为（　　）。

A. $O(n^3)$　　B. $O(n\log n)$

C. $O(n^2)$　　D. $O(n^2\log n)$

【2013 年——华南理工大学】

【考查内容】单源最短路径的迪杰斯特拉算法的时间复杂度。

【解析】最短路径算法可以分为单源点最短路径和全源最短路径。所谓单源最短路径问题是指：已知图 G＝（V，E），我们希望找出从某给定的源结点 S∈V 到 V 中的每个结点的最短路径，时间复杂度为 $O(n^2)$。多源点最短路径是任意两个结点之间的最短路径，时间复杂度为 $O(n^3)$。

【参考答案】C

6. 关键路径是 AOE 网络中（　　）。

A. 从源点到汇点的最长路径　　B. 从源点到汇点的最短路径

C. 最长回路　　D. 最短回路

【2010 年——中山大学】

【考查内容】关键路径的基本概念。

【解析】从源点到汇点的所有路径中，权值之和最大的距离称为关键路径。关键路径可以有多条。

【参考答案】A

7. 在有向图 G 的拓扑序列中，若顶点 Vi 在顶点 Vj 之前，则下列情形不可能出现的是（　　）。

A. G 中国有弧<Vi,Vj>　　B. G 中有一条从 Vi 到 Vj 的路径

C. G 中没有弧<Vi,Vj>　　D. G 中有一条 Vj 到 Vi 的路径

【2011 年——南京邮电大学】

【考查内容】有向图的拓扑排序。

【解析】在有向图的拓扑排序中，若顶点 Vi 在顶点 Vj 之前，未必有 Vi 到 Vj 的路径，如在图 5.8 的拓扑排序得到的序列（V1, V3, V4, V6, V2, V5, V7）中，V3 在 V4 之前，但是并不存在 V3 到 V4 的路径。故而，B 答案错误。而 V1 在 V7 之前，也并没有弧<V1, V7>，故而 A、C 都不一定成立。

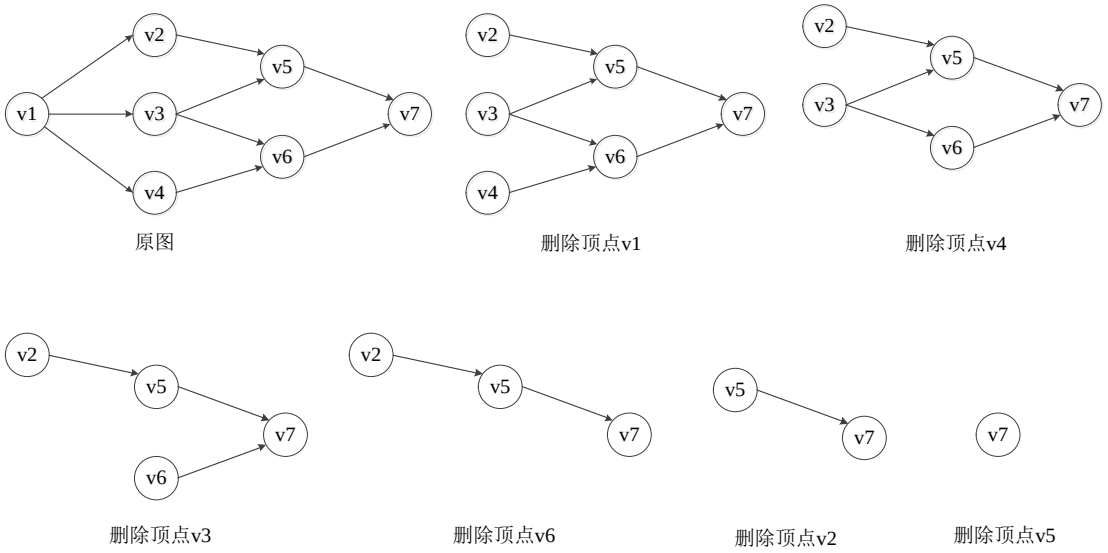


图 5.8

若存在一条 Vj 到 Vi 的路径，那么 Vi 的完成，必须以 Vj 的完成为前提，但是拓扑排序中 Vi 在 Vj 之前。显然，如果假设成立的话，图 G 中存在环，不能进行拓扑排序。故而，D 答案不成立。

【参考答案】A

8. 对于图 5.9，完成下列指定操作。

(1). 从顶点 A 出发，求它的深度优先生成树；

(2). 从顶点 E 出发。求它的广度优先生成树；

(3). 根据普利姆（Prim）算法，求她的最小生成树。

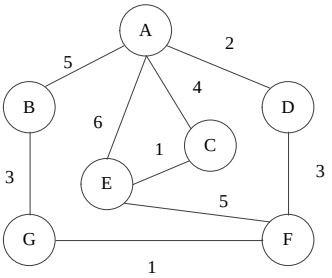


图 5.9

【2013 年—山东科技大学】

【解析】求广度优先生成树和深度优先遍历生成树、最小

(1). 从顶点 A 出发，对图进行深度优先遍历，过程如图 5.10 所示。

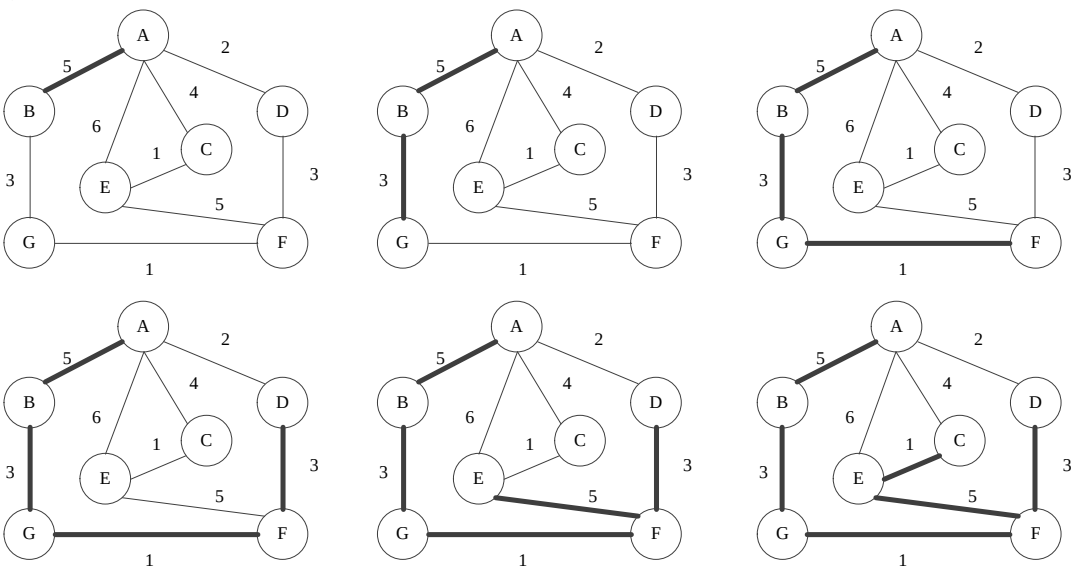


图 5.10

【注意】当遍历到结点 D 的时候，下一个结点 A 已经遍历过。因为，返回 D 的上一个节点 F，看看 F 有没有其他未遍历的相邻结点。所以，D 遍历完后，下一个访问的（未被访问过）的结点是 E。

从上图可得到从 A 点出发，对图深度优先遍历的生成树，如图 5.11 所示。

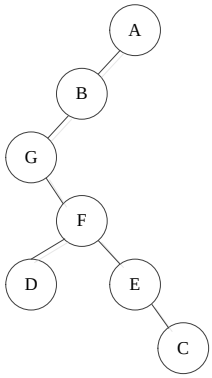


图 5.11

(2). 从顶点 E 出发，对图进行广度优先遍历，过程如图 5.12 所示。

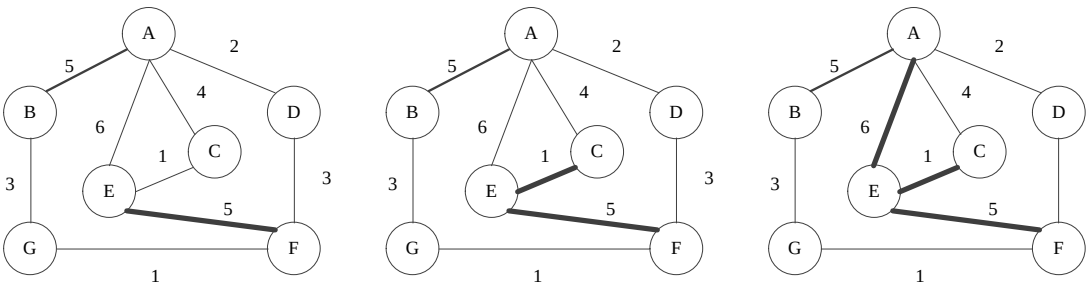


图 5.12

遍历到此，E 已经没有相邻的结点可以继续遍历。于是，开始访问 E 第一个被访问的结点的相邻结点 F。（注意，广度优先遍历使用的是队列，深度优先遍历使用的是堆栈，同学们要注意区分这两种不同的遍历算法）

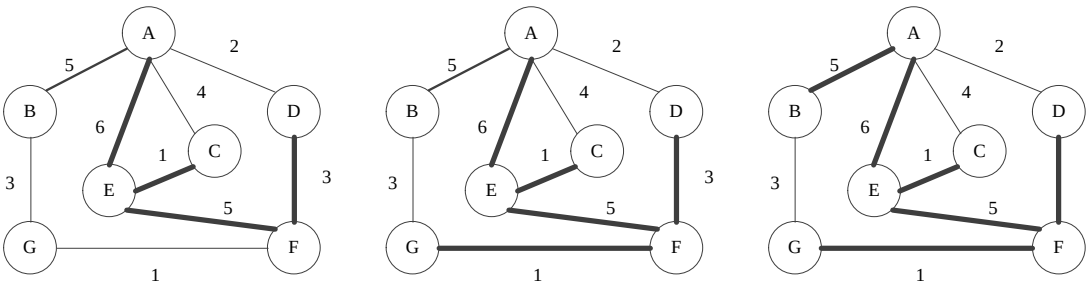


图 5.13

访问 F 的第一个相邻的结点 D，再访问第二个相邻的结点 G，至此 F 已经没有未访问过的相邻结点。于是，访问 C，C 没有未被访问过的相邻的结点。于是，访问 A，A 有未被访问过的结点 B，于是访问 B。至此，该图遍历完毕，得到的广度优先遍历生成树如下图 5.14 所示。

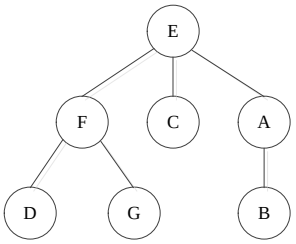


图 5.14

(3). 从顶点 A 开始根据 Prim 算法，构造最小生成树的过程如图 5.15 所示。

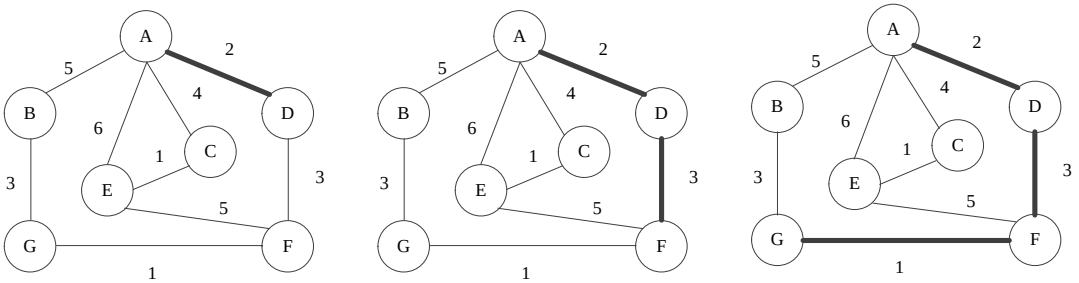


图 5.15

(A). Prim 算法开始时，集合为空集，并入一个顶点 A 后，集合变成 VT={A}，未并入的顶点集合 VX=V-VT={B,C,D,E,F,G}，从集合 VT 中的顶点到集合 VX 任意顶点的距离中，选取边最近的顶点 D，并入 VT 中；

(B). VT={A,D}，VX={ B,C,E,F,G }，从 VT 的顶点集合中，抽取到 VX 的顶点集合中，距离最小（或者权值最小）的边（D,F），把 F 并入 VT 中，并从 VX 中删除 F；

(C). 重复进行类似与（B）的操作，直到 VX 中的所有顶点都并入集合 VT 中。

(D). 最终的生成树如下图的最后一幅如图 5.16 所示。

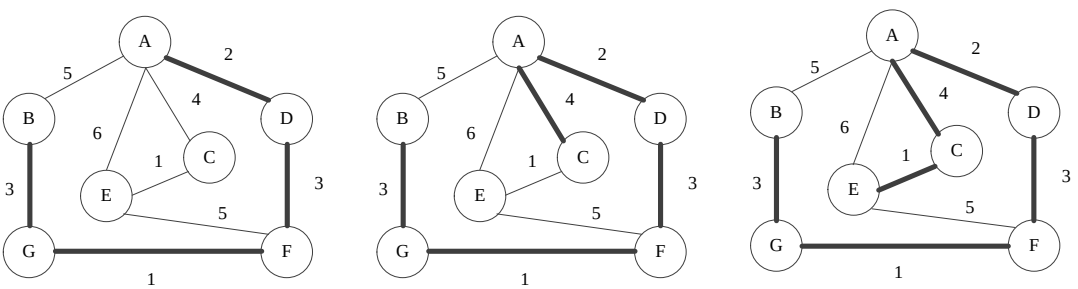


图 5.16

8. 下列关于 AOE 网的叙述中，不正确的是（　　）。

A. 关键活动不按期完成就会影响整个工程的完成时间

B. 任何一个关键活动提前完成，那么整个工程将会提前完成

C. 所有的关键活动提前完成，那么整个工程将会提前完成

D. 某些关键活动提前完成，那么整个工程将会提前完成

【2015 年—北京邮电大学】

【考查内容】AOE 网。

【解析】AOE（Activity on Edge）网的边表示活动。关键活动是指关键路径上面的活动。关键路径是工程中代价最大的路径（如时间代价）。关键路径不按期完成，将会延长整个工程的时间。所以，A 答案正确。但是，关键路径可能不止一个，所以不是所有关键路径上的共同关键活动提前完成，可能不会使得整个工期的提前完成。所以，B 答案错误。D 答案同理。

【参考答案】C

第 6 章　查找

6.1　顺序查找法

温馨提示：顺序查找法通常考查查找一个元素的平均查找长度。对于自主命题的高校而言，可能会有简单的编程题。请同学们掌握这些基础知识。

1. 若查找每个元素的概率相等，则在长度为 n 的顺序表上查找任一元素的平均查找长度为（　　）。

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

   【2013 年——上海海事大学】

   【考查内容】等概率下顺序表的平均查找长度。

   【解析】在等概率的情况下，每个元素的查找概率都是 1/n，其中查找最后一个元素需要比较 1 次，查找第 n-1 个结点需要比较 2 次。依次类推，查找第 1 个结点需要比较 n 次，平均查找长度为（1+2+3+…+n）/n=(n+1)/2。

   【参考答案】D

6.2　折半查找法

温馨提示：折半查找算法是本章的重点内容，也是数据结构的重点考点，主要考查：1、折半查找的条件；2、折半查找条件下的关键字比较次数、平均时间复杂度；3、折半查找树的建立。请同学们一定要把本考点的知识掌握，并运用自如。

1. 已知一个长度为 16 的顺序表 L，其元素按关键字有序排列。若采用折半查找法查找一个 L 中不存在的元素，则关键字的比较次数最多的是（　　）。

   A. 4　　B. 5　　C. 6　　D. 7

   【2010 年统考——第 9 题】

   【考查内容】折半查找的过程。

【解析】我们根据题目要求，假设长度为 L 的顺序表的下标为 1~16，我们利用该数组下标来构建折半查找树，如图 6.1 所示。

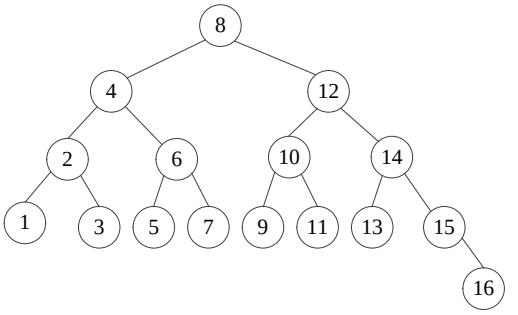


图 6.1

从图 6.1 可以看出，比较次数最多的是，位于数组下标 16（下标从 1 开始）位置的关键字，该位置的关键字需要比较 5 次。关键字查找失败最凄惨的情况下，需要从根节点比较到最深层的叶子结点，比较次数等于树的高度。

当然，这是比较愚蠢的办法，我们也可以利用公式来计算。具有 n 个关键字（结点）的折半查找（判定）树的高度为$\lfloor log_2 n \rfloor + 1$，显然，表长为 16 时，树的高度为 5,所以查找失败时最多需要比较 5 次。

【参考答案】B

2. 适用于折半查找的表的存储方式及元素排列要求为（　　）。

A. 链接方式存储,元素无序　　B. 链接方式存储,元素有序

C. 顺序方式存储,元素无序　　D. 顺序方式存储,元素有序

【2010 年——中山大学】

【考查内容】折半查找的条件。

【解析】**折半查找**是一种高效的查找方法，它可以明显减少比较次数，提高查找效率。但是，折半查找的先决条件是查找表中的数据元素必须有序。除此之外，折半查找还要求数据以顺序存储方式存储在线性表中。（简单记忆为：**“顺序存储，数据有序”**。）

2015 年，暨南大学的考研真题中，也有类似的题目（如类似题 1 所示）。请大家留意！

【参考答案】D

类似题 1：对线性表进行折半查找时,要求线性表必须（　）。

A.以顺序方式存储

B. 以顺序方式存储,且结点按关键字有序排序

C. 以链接方式存储

D.以链接方式存储,且结点按关键字有序排序

【2015 年—暨南大学】

3. 二叉搜索树中，最小元素的左子树（　　），它的右子树（　　）。

A. 一定为空，不一定为空　　B. 不一定为空，一定不为空

C. 一定不为空，不一定为空　　D. 不一定为空，不一定为空

【2011 年——南京邮电大学】

【考查内容】二分查找树的最小元素的左右子树特点。

【解析】二叉搜索树，即二分查找树。在二分查找树中，最小的元素左子树一定为空，若不为空，则该结点的左子树上的所有结点的关键字都会比该结点的关键字小。但是，最小元素的右子树却不定为空，可以有比该最小元素大的元素在该结点的右子树上。

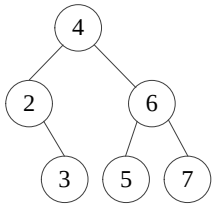


图 6.2

如图 6.2 所示，最小元素为 2，其左子树为空，但是该结点的右孩子为 3。

【参考答案】A

4. 一个有序数据序列中有 15 个数据，采用二分查找法在其中查找一个数据，（查找成功的情况下）最多要比较几次就能得到查找结果（　　）。

A. 4　　B. 5　　C. 1　　D. 15

【2014 年——中国计量学院】

【考查内容】二分查找。

【解析】我们对该有序数据序列的下标 1~15 构造如下图 6.3 所示的折半查找判定树。

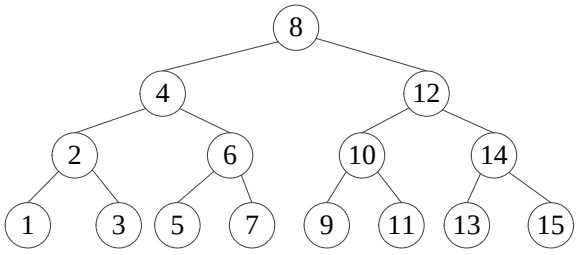


图 6.3

从图 6.3 可以看出，查找位置 13 和 15 的元素，需要比较 4 次，也即最多需要进行 4 次比较才能得到结果。

有的同学认为，判定树高度为 5，但是第 5 层实际意义上是没有节点的，因此不进行比较，所以只能比较 4 次。

【参考答案】A

5. 假设对长度 n=50 的有序表进行折半查找，则对应的判定树高度为（　　）。

A. 8　　B. 7　　C. 6　　D. 5

【2013 年——中南大学】

【考查内容】二分查找判定树的高度。

【解析】具有 n 个结点的折半查找判定树的深度为$\lfloor log_2 n \rfloor + 1$。此公式请同学们牢记！对于长度 n=50 的有序表，折半查找的判定树高度为$\lfloor log_2 50 \rfloor + 1 = 5 + 1 = 6$。

【参考答案】C

6. 折半查找（　　）存储结构。

A. 只适用于顺序

B. 只适用于链式

C. 既适用于顺序也适用于链式

D. 既不适用于顺序也不适用于链式

【2014 年——宁波大学】

【考查内容】折半查找的存储结构。

【解析】链式存储结构只适用于顺序查找，不适用于折半查找。由于折半查找要求能够直接定位线性表中任一元素，而链式结构无法做到这一点。

【参考答案】A

7. 已知二叉搜索树（Binary Search Tree）/二叉排序树(Binary Sorting Tree)的结点定义如下：

```
typedef struct _BSNode{
int key;
struct _BSNode *LChild,* RChild;//左子树的关键字比根结点小，右子树的关键字比
//根结点大
} BSNode;
```

编写函数 Find(BSNode *root, int key),其功能是在以结点 root 为根的二叉搜索树中找“比参数 key 大的最小值”。若找不到，则返回 NULL，否则，返回该结点地址。

【2013 年—中山大学】

【考查内容】

【解析】

```
BSNode * Find(BSNode *root,int key)  {
      BSNode * Max=NULL;
       if(root==NULL) return NULL;
       while(root!=NULL) {
            if(root->Key > key)  { Max=root;root=root->LChild; }
           else if(root->Key < key){ root=root->RChild;}
           else {root=root->RChild;}
        }
       return Max;
```

}

6.3 B-树

温馨提示：B-1 树部分，主要考查：1、考查方式是 B-树的基本概念；2、B-树的建立；3、结点插入和删除时，B-树的调整；4、B+树。本考点对考生提出的不是编程方面的要求，而是对 B-树的建立、插入和删除结点时对 B-树进行调整的手工操作。而且，手工操作很多考生掌握不到精髓，我们特别给了经典的总结，希望有助于大家掌握本考点的内容。

1. 下列叙述中，不符合 m 阶 B-树定义要求的是（　　）。

A. 根结点最多有 m 棵子树

B. 所有叶结点都在同一层上

C. 各结点内关键字均升序或降序排列

D. 叶结点之间通过指针链接

【2009 年统考——第 8 题】

【考查内容】考查 B-树的定义。

【解析】B-树是一种多路搜索树（并不是二叉的），它具有如下性质：

(1). 定义任意非叶子结点最多只有 M 个孩子（子树），且 M>2；

(2). 根结点的孩子数为[2, M]，除根结点以外的非叶子结点的儿子数为[$\lceil m/2 \rceil$, M]；

(3). 每个结点存放至少$\lceil m/2 \rceil - 1$和至多 M-1 个关键字；

(4). 非叶子结点的关键字：K[1], K[2], „, K[M-1]，且 K[i] < K[i+1]（亦可以降序），即结点中的数据是有序的；

(5). 非叶子结点的指针：P[1], P[2], „, P[M]；其中 P[1]指向关键字小于 K[1]的子树，P[M]指向关键字大于 K[M-1]的子树，其它 P[i]指向关键字属于(K[i-1], K[i])的子树；

(6). 所有叶子结点位于同一层；

D 选项是 B+树的特点，不是 B-树的特点。在 B-树基础上，B+树为叶子结点增加链表指针，所有关键字都在叶子结点中出现，非叶子结点作为叶子结点的索引。B+树总是到叶子结点才命中。

【参考答案】D

2. 已知一棵 3 阶 B-树，如图 6.4 所示。删除关键字 78 得到一棵新 B-树，其最右叶结点中的关键字是（　　）。

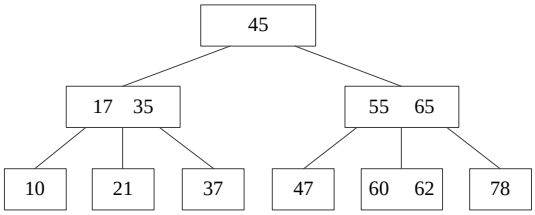


图 6.4

A. 60　　B. 60, 62　　C. 62, 65　　D. 65

【2012 年统考——第 9 题】

【考查内容】B-树的删除操作。

【解析】显然，上图中的 3 阶 B-树上，兄弟节点{60 62}可以在删除关键字 78 后，借一个关键字给原关键字 78 所在的结点。

怎么理解呢？其实借关键字不是简单地拿过来用，而是和从要借关键字的兄弟结点往父节点中最靠近自己的那个关键字换。也就是说，用兄弟结点中最靠近自己的关键字，去交换父节点中最靠近自己的关键字。

最简单的规则，就是中序遍历 B-树，要保证递增有序。这也是 B-树增加和删除结点时调整的依据。

本题中，最靠近关键字 78 所在结点的父节点的关键字 65，其兄弟中最靠近它的关键字是 62。那么，拿从兄弟结点{60 62}借来的关键字 62 与父节点的关键字 65 交换。交换后，65 代替了 78，而 62 代替了 65（如图 6.5 所示）。

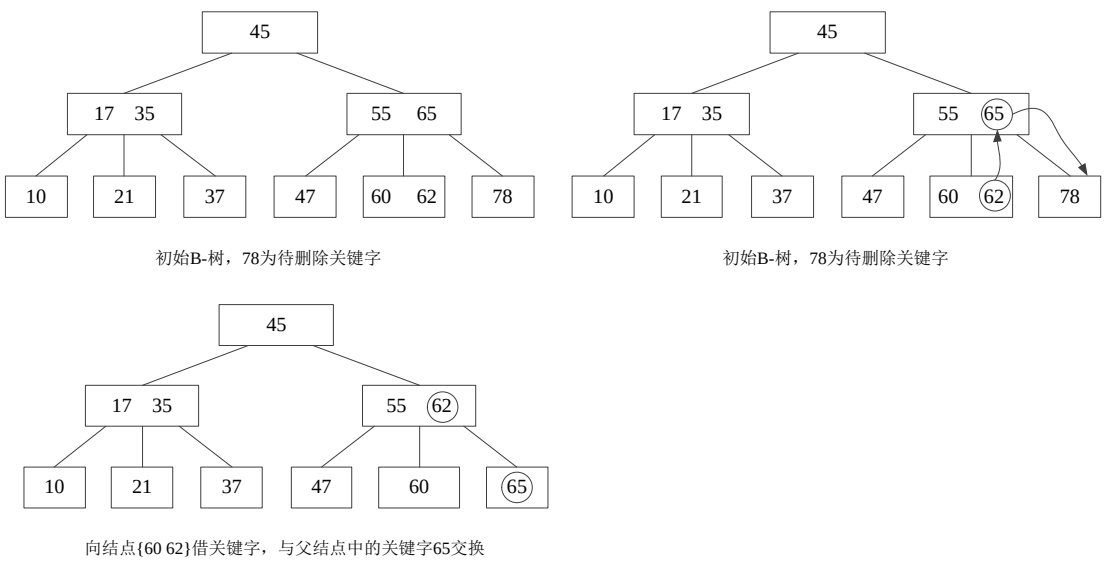


图 6.5

要注意，为了校验是否调整正确，可以看看中序遍历得到的序列是不是递增有序的。显然，我们调整后的 B-树是递增有序的。

【参考答案】D

3. 在一株高度为 2 的 5 阶 B-树中，所含关键字的个数最少是（　　）。

A. 5　　B. 7　　C. 8　　D. 14

【2013 年统考——第 10 题】

【考查内容】

【解析】关键字最少时，应该是结点刚好能分裂的时候。对于一棵高度为 2 的 5 阶 B-树，显然当第一层的根结点有关键字 5 个时，才会分裂成两层。如，依次将关键字 1、2、3、4、5 插入到 B-树中，只有当关键字 5 插入时才会出现结点分裂，如图 6.6 所示。

图 6.6

经典问题：要是本题改成结点中关键字数最多的情况呢？请同学们考虑一下这个问题。在接下来的第 4 题，我们会进行详细地解释。

【参考答案】A

4. 在一棵具有 15 个关键字的 4 阶 B-树中，含有关键字的结点个数最多是（　　）。

A. 5　　B. 6　　C. 10　　D. 15

【2014 年统考——第 9 题】

【考查内容】B-树的特点。

【解析】在第 1 题中，我们给出了 B-树的性质。对于关键字数量固定的 B-树，显然结点个数最多时，每个结点中的关键字个数尽量少。

我们知道，B-树根节点关键字最少的情况下可以是 1 个，每个非叶子结点存放至少 $\lceil 4/2 \rceil - 1 = 1$ 和至多 4-1=3 个关键字，叶子结点中最少可以只有一个关键字。这样一来，这棵 B-树就跟我们之前接触到的二叉排序树长得很像了。

我们可以假设每个结点只有一个关键字，那么，15 个关键字可以有 15 个结点。那么，B 树的第 1 层有 1 个结点，第 2 层有 2 两个结点，第 3 层有 4 个结点，第 4 层有 8 个结点，总共 15 个结点。

【参考答案】D

5. 已知关键字的集合{56，8，15，80，10，22，28，50，60,40,90}

(1). 试按给出的序列构造一棵平衡二叉树。

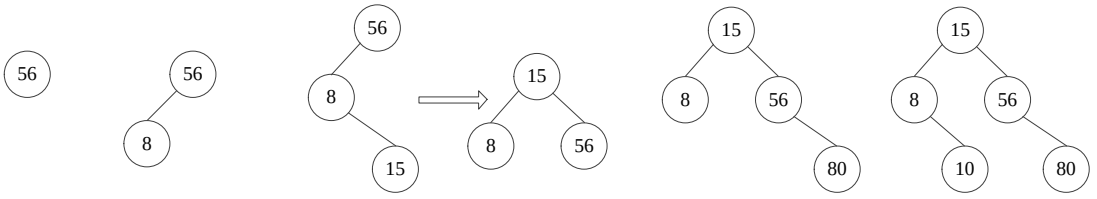
(2). 试构造 3 阶 B-树。

(3). 写出依次删除关键字 60,40 后的 B-树。

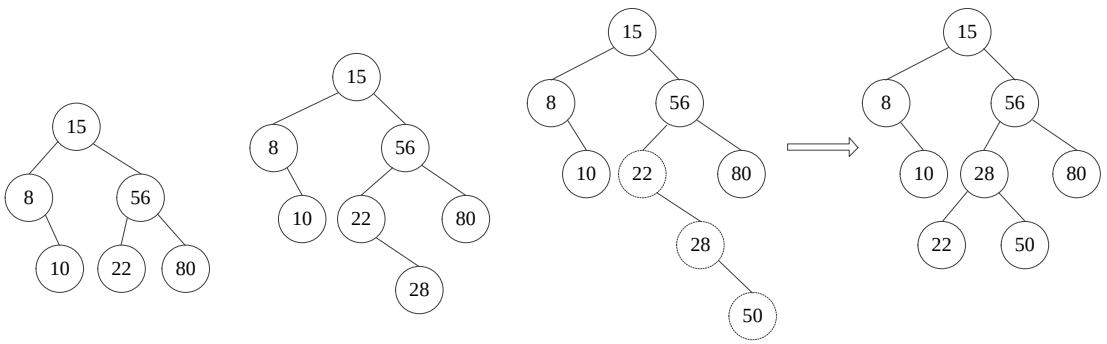
【考查内容】平衡二叉树的构造，B-树的构造方法和删除结点后的删除操作。

【解析】

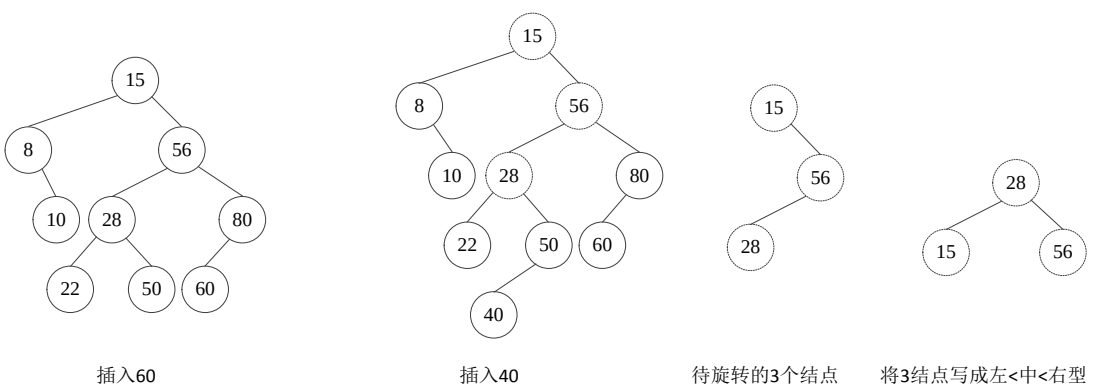
(1). 构造平衡二叉树的过程如图 6.7 所示。



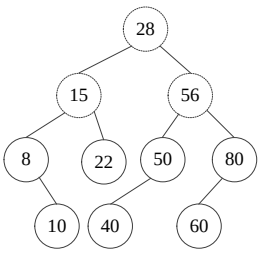
插入56　插入8　插入56，LR型不平衡　旋转　插入80　插入10



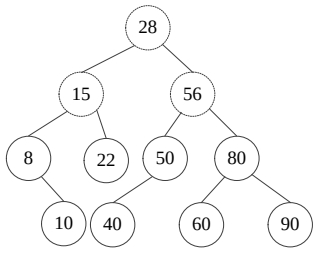
插入22　插入28　插入50，RR型不平衡　旋转



插入60　插入40　待旋转的3个结点　将3结点写成左<中<右型



接入分支，形成旋转后的平衡



插入90

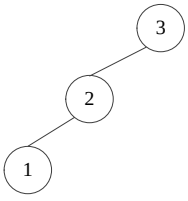
图 6.7

经典总结：

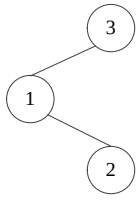
平衡二叉树的构建，说到底，就 3 个问题：

① **转哪儿的问题**。即哪儿不平衡，需要旋转的问题。简单的查找不平衡方法，可以沿着插入结点往根结点方向观察，首次出现不平衡的地方就是等待旋转的地方。

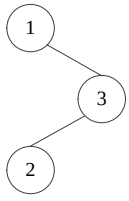
② **怎么转的问题**。很多书上都会写成 LL、LR、RL、RR 型不平衡，但是这个不好掌握，很多同学看不懂。我们的看法是，不管是这四种不平衡的哪一种不平衡，都只涉及到 3 个结点的旋转。所以，我们总结了一个简单的观察 LL、LR、RL、RR 型不平衡的方法。



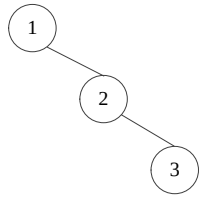
LL型不平衡



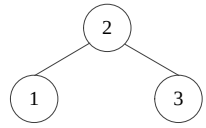
LR型不平衡



RL型不平衡



RR型不平衡



三种不平衡最终都要旋转成左中右型

图 6.8

如 LL 型不平衡，为什么是 LL？因为 2 是 3 的左(L)孩子，1 是 2 的左(L)孩子。左(L)左(L)所以为 LL 型。再如 RL 型，3 是 1 的右(R)孩子，2 是 3 的左(L)孩子，右(R)左(L)所以为 RL 型。这样是不是好理解了。

不管是 LL 型、LR 型、RL 型还是 RR 型不平衡，最终都需要调整成左（左孩子）中（父结点）右（右孩子）型。而且，父结点大于左孩子，小于右孩子，保证中序遍历三个结点，得到一个递增的序列。简单的说，就是：先画出上图中左右变的图的形状，然后把中间值放入父结点，把最小值放入左孩子结点，把最大值放入右孩子结点，无需理解太多，仅此即可。

③ **怎么挂的问题。**需要调整的不平衡部分，在调节平衡之后，原来的树中会落下许多等待接入的分支，如本题中出现如图 6.9 所示的 4 个等待接入的分支。

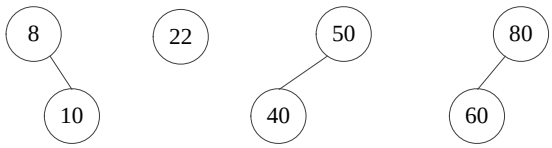


图 6.9

那么，怎么接入呢？一般情况下，不会拆开分支的。即把整个分支作为一个类似于结点的整体插入，并保证中序遍历整棵二叉树是递增有序的。口诀：“哪儿适合挂哪儿，整个分支一起挂，中序遍历非递减”。

有的书上，也写了“怎么转的问题”和“怎么挂的问题”，但是讲得比较复杂，不太好理解。还是上面总结的技巧，便于记忆和理解。

本题的总结，就到这里了，不太明白的同学，请结合第四章(树)的平衡二叉树考点来进行理解。

(2).3 阶 B-树的构造过程如图 6.10 所示。

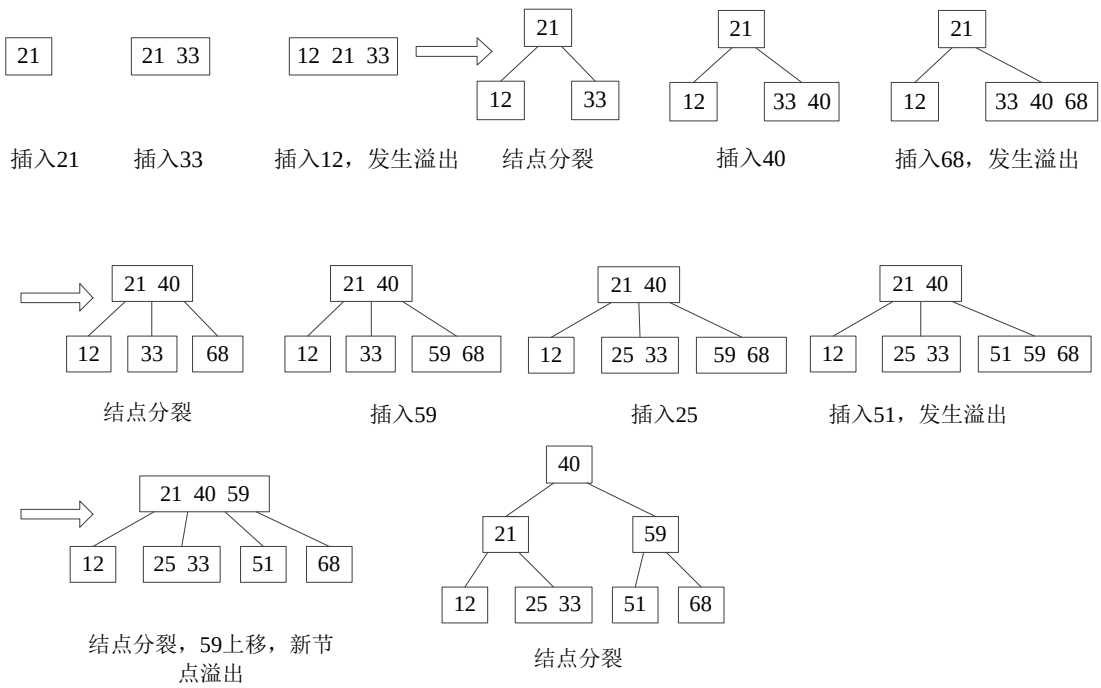
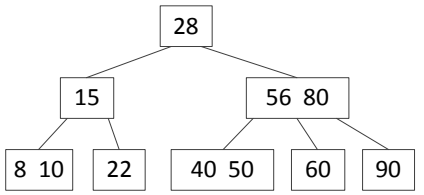
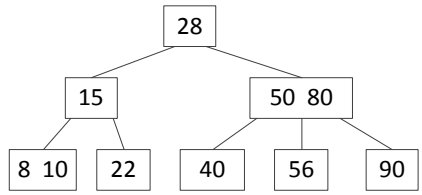


图 6.10

(3).依次删除 60,40 之后的 B-树如下。（注意，不是分别删除，而是依次删除）。首先看看删除 60 之后的 B-树，如图 6.11 所示。



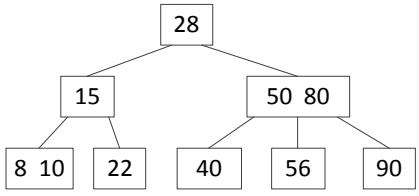
删除60



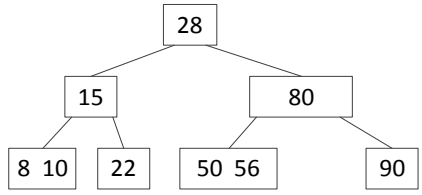
向兄弟借一个值

图 6.11

在删除了 60 的基础上，再删除 40。



删除40



把50拉下来，与56合并成一个大的结点

图 6.12

试画出从空树开始，由字符序列(t,d,e,s,u,g,b,j,a,k)构成的二叉平衡树，并为每一次的平衡处理指明旋转类型。再次插入字符 a，画出此时的平衡二叉树

【2015 年—北京邮电大学】

6.4 散列（Hash）表及其查找

温馨提示：本考点主要命题有以下几种形式：1、什么是哈希表的特点？影响哈希表查找效率的因素有哪些？有哪些解决冲突的方法？2、怎么构建哈希表、怎么样利用冲突解决方案（如链地址法、二次探测再散列等）来解决冲突。这部分需要考生手工画图；3、计算哈希表的在查找成功和查找失败的情况下平均查找长度（或查找某一个元素的比较次数）。哈希表在 408 统考或者高校自主命题中，都可能出大题，请同学们务必掌握。

1. 为提高散列（Hash）表的查找效率，可以采取的正确措施是（　　）。

Ⅰ．增大装填（载）因子

Ⅱ．设计冲突（碰撞）少的散列函数

Ⅲ．处理冲突（碰撞）时避免产生聚集（堆积）现象

A. 仅Ⅰ
B. 仅Ⅱ
C. 仅Ⅰ、Ⅱ
D. 仅Ⅱ、Ⅲ

【2011 年统考——第 9 题】

【考查内容】提高哈希表查找效率的方法。

【解析】散列表的查找效率，由三个因素决定：散列函数、装填因子和处理冲突的方法。其中，装填因子表示的是关键字的个数与哈希表表长的比值。一般情况下，装填因子越大，冲突的概率也越大。显然，I 错误。

设计好的散列函数，可以减少冲突，提高查找效率。故而，II 正确。

堆积现象，我们可以理解成某一个位置附近有很多的关键字。这种现象可以通过设计好的处理冲突的函数。例如，链地址法能够很好的避免冲突。

【参考答案】B

2. 用哈希（散列）方法处理冲突（碰撞）时可能出现堆积（聚集）现象，下列选项中，会受堆积现象直接影响的是（　　）。

A. 存储效率
B. 散列函数
C. 装填（装载）因子
D. 平均查找长度

【2014 年统考——第 8 题】

【考查内容】影响散列冲突处理方法的堆积现象。

【解析】堆积现象，用通俗易懂的话来说，就是产生了冲突。存储效率我们可以理解成存储空间的利用率，显然，对于一定的元素个数，装入一定表长的存储单元中，存储效率是一定的，不会受到堆积现象的影响。装填因子表示的是关键字的个数与表长的比值，显然也是个定值，不会收到堆积现象的影响。散列函数显然不会收到堆积现象的影响，而是散列函数影响堆积现象。

显然，产生了堆积现象，会使得平均查找长度增大。所以，D 选项正确。

【参考答案】D

3. 在存储信息的过程中，通过关键字的计算来确定其存储位置的数据结构是（　　）。

A. Hash 表
B. 二叉搜索树
C. 链式结构
D. 顺序结构

【2014 年—中山大学】

【考查内容】哈希表的定义。

【解析】哈希表存储的基本思想是：以数据表中的每个记录的关键字 key 为自变量，通过一种函数 Hash(key)计算出函数值。把这个函数值解释为一块连续存储空间（即数组空间）的单元地址（即下标），将该记录存储到这个单元中。在此称该函数 Hash 为哈希函数或者散列函数。按这种方法建立的表称为哈希表或者散列表。

【参考答案】A

4. 在 Hash 函数 H(k)=k MOD m 中，一般来讲 m 应取（　　）。

A. 奇数

B. 偶数

C. 素数

D. 充分大的数

【2014 年——武汉科技大学】

【考查内容】哈希表的模数的取值。

【考查内容】显然，题目描述的是哈希表构造方法中的除留余数法。该方法取关键字 k 除以哈希表长度 m 所得余数作为哈希函数地址。

该方法的关键是选择好哈希函数的长度 m，使得数据集合中的每一个关键字通过该函数转化后映射到哈希表的任意地址上的概率相等。理论研究表明，在 m 取值为素数时，冲突可能性相对比较小。

【经典误区】请同学们注意哈希表的构造法和冲突解决方法，这二者是不同的。前者是用来构建哈希表，后者是构建哈希表的过程中产生了冲突，怎么样解决冲突，将关键字插入到哈希表中的策略。

哈希表构造方法有:

(1). 直接定址法

(2). 除留余数法

(3). 平方取中法

(4). 折叠法

(5). 数值分析法

解决冲突的方法有:

(1). 开放定址法（包括线性探测法和平方探测法）

(2). 链地址法

【参考答案】C

5. 设散列表为 HT[0..18]，即表的长度为 19。散列函数为：H(key) = key % 19，采用线性探测再散列法解决冲突，若插入的关键码序列为{63, 251, 191, 164, 133, 125, 118, 161, 157, 134, 87, 291, 386, 153, 59, 206}。

(1). 试画出插入这 16 个关键码后的散列表。

(2). 计算在等概率情况下查找成功的平均查找长度 ASL。

【2013 年——上海海事大学】

【解析】首先，我们根据哈希函数 H(key)=key mod 19（用来计算 key 的第一个地址），并利用线性探测再散列方法

$$H_i = (H(key) + i) mod\ 19 \quad \text{(其中 19 为表长 m)}$$

来处理冲突。处理冲突是(mod m)而不是(mod p)，本题中 m=p 可以看作是特殊情况，一般取素数 p 为小于等于 m 的最大素数（也有很多书直接取 p 等于表长），同学们千万要注意了。

下面，我们来计算关键字的位置。

H(63)=63 mod 19=**6**；

H(251)= 251 mod 19 =**4**；

H(191)= 191 mod 19 =**1**；

H(164)= 164 mod 19 =**12**；

H(133)=133 mod 19 =**0**；

H(125)=125 mod 19 = **11**；

H(118)=118 mod 19 =4 (该位置存放的元素是 251，冲突)，利用线性探测再散列方法来探测下一个位置 H(118)=(118+1) mod 19=**5**；

H(161)=161 mod 19 =**9**；

H(157)=157 mod 19 =5 (该位置已经存放元素 118，冲突)，利用线性探测再散列方法来探测下一个位置 H(157)=(157+1) mod 19=6（该位置存放元素 63，仍然冲突），利用线性探测再散列方法来探测下一个位置 H(157)=(157+2) mod 19=**7**；

H(134)=134 mod 19=1(该位置已经存放 191，发生冲突)，利用线性探测再散列方法来探测下一个位置 H(134)=(134+1) mod 19=**2**；

H(87)=87 mod 19 = 11 (该位置已存放元素 125，冲突)，利用线性探测再散列方法来探测下一个位置 H(87)=(87+1) mod 19=12（该位置已存放元素 164，冲突），利用线性探测再散列方法来探测下一个位置 H(87)=(87+2) mod 19=**13**；

H(291)=291 mod 19 = 6 (该位置已存放元素 63，冲突) ，利用线性探测再散列方法来探测下一个位置 H(291)=(291+1) mod 19=7（该位置已存放元素 157，冲突），利用线性探测再散列方法来探测下一个位置 H(291)=(291+2) mod 19=**8**；

H(386)=386 mod 19 = 6(该位置已存放元素 63，冲突) ，利用线性探测再散列方法来探测下一个位置 H(386)=(386+1) mod 19=7（该位置已存放元素 157，冲突），利用线性探测再散列方法来探测下一个位置 H(386)=(386+2) mod 19=8（该位置已存放元素 291，冲突），利用线性探测再散列方法来探测下一个位置 H(386)=(386+3) mod 19=9（该位置已存放元素 161，冲突），利用线性探测再散列方法来探测下一个位置 H(386)=(386+4) mod 19=**10**；

H(153)=153 mod 19 = 1(该位置已经存放 191，发生冲突)，利用线性探测再散列方法来探测下一个位置 H(153)=(153+1) mod 19=2(该位置已经存放 134，发生冲突)，利用线性探测再散列方法来探测下一个位置 H(153)=(153+2) mod 19=**3**；

H(59)=59 mod 19 = 2(该位置已经存放 134，发生冲突)，利用线性探测再散列方法来探测下一个位置 H(59)=(59+1) mod 19=3(该位置已经存放 153，发生冲突)，利用线性探测再散列方法来探测下一个位置 H(59)=(59+2) mod 19=**4** (该位置存放的元素是 251，冲突)，利用

线性探测再散列方法来探测下一个位置 H(59)=(59+3) mod 19=**5**(该位置存放的元素是 118，冲突)，利用线性探测再散列方法来探测下一个位置 H(59)=(59+4) mod 19=**6**，…，如此探测下去，H(59)=(59+10) mod 19=12（该位置已存放元素 164，冲突），利用线性探测再散列方法来探测下一个位置 H(59)=(59+11) mod 19=13（该位置已存放元素 87，冲突），利用线性探测再散列方法来探测下一个位置 H(59)=(59+12) mod 19=**14**;

H(206)=206 mod 19 =**16**。

所得哈希表如表 6.1 所示。

表 6.1

数组下标	0	1	2	3	4	5	6	7	8	9
关键字	133	191	134	153	251	118	63	157	291	161
查找成功的比较次数	1	1	2	3	1	2	1	3	3	1
数组下标	10	11	12	13	14	15	16	17	18	
关键字	386	125	164	87	59		206			
查找成功的比较次数	5	1	1	3	13		1			

表 6.5 的最后两行分别表示等概率情况下，查找成功或者不成功时关键字的比较次数。根据上表，我们可以计算得到，在查找成功的情况下的平均查找长度为

$$ASL_{succ} = \frac{1 \times 8 + 2 \times 2 + 3 \times 4 + 5 + 13}{16} = \frac{42}{16} = \frac{21}{8}$$

【特别注意】考查散列表时，通常都会考查同学们会不会根据散列函数和冲突解决办法来构建哈希表，并且要求同学们计算查找成功和查找失败的平均查找长度。请同学们务必掌握。

第 7 章　内部排序

7.1　插入排序

温馨提示：本考点主要考查折半插入排序和直接插入排序。请同学们注意插入排序的复杂度、平均移动元素次数，并围绕这两个点展开复习。其实，希尔排序也是一种插入排序，但是我们参考 408 统考大纲，将该内容独立出来考查。

1. 对一待排序序列分别进行折半插入排序和直接插入排序，两者之间可能的不同之处是（　　）。

A. 排序的总趟数　　B. 元素的移动次数

C. 使用辅助空间的数量　　D. 元素之间的比较次数

【2012 年统考——第 11 题】

【考查内容】直接插入排序和折半插入排序的区别。

【解析】所谓排序算法过程，就是不断的依次将元素插入前面已排好序的序列中。直接插入排序每次从无序表中取出第一个元素，把它插入到有序表的合适位置，使有序表仍然有序。

折半插入排序是对直接插入排序算法的一种改进。由于前半部分为已排好序的数列，这样我们可以不用按顺序依次寻找插入点，而是采用折半查找的方法来加快寻找插入点的速度。折半查找的方法来寻找插入位置，可以减少比较次数。但不影响排序的趟数（仍然是 n-1 趟排序）、使用得辅助空间的数量、和元素的移动次数。

【参考答案】D

7.2　冒泡排序（bubble sort）

温馨提示：冒泡排序的命题方式有两种：1、围绕记录交换次数展开；2、让同学们简单地写一个冒泡排序程序（这种命题方式在 408 中可能性很小，自主命题高校出题的可能性比较大）。请同学们围绕常见的命题方式来复习。

1. 一组记录（50，40，95，20，15，70，60，45，80）进行冒泡排序时，第一趟需进行相邻记录的交换的次数为（　　）。

A. 5　　B. 6　　C. 7　　D. 8

【2013 年——暨南大学】

【考查内容】冒泡排序。

【解析】冒泡排序的基本思想是：假设待排序的 n 个对象的序列为 R[0]、R[1]、…、R[n-1]，起始时排序范围是从 R[0]到 R[n-1]。在当前的排序范围之内，自右至左对相邻的两个结点依次进行比较，让值较大的结点往下移(下沉)，让值较小的结点往上移(上冒)。每趟起泡都能保证值最小的结点上移至最左边，下一遍的排序范围为从下一结点到 R[n-1]。

第一趟排序的过程如下：

(1). 排序开始时，50 和 40 比较，50 比 40 大，二者交换位置；

(2). 50 和 95 比较，50 小于 95，无需交换；

(3). 95 和 20 比较，95 比 20 大，二者交换位置；

(4). 95 和 15 比较，95 比 15 大，二者交换位置；

(5). 95 和 70 比较，95 比 70 大，二者交换位置；

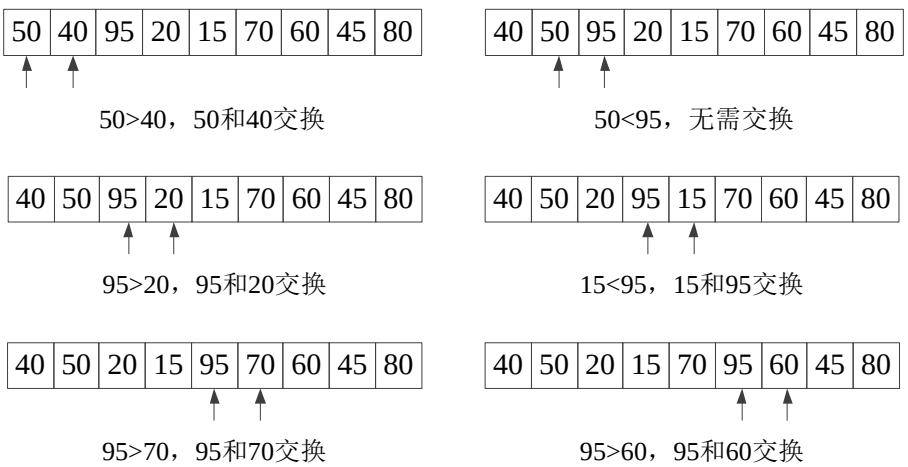
(6). 95 和 60 比较，95 比 60 大，二者交换位置；

(7). 95 和 45 比较，95 比 45 大，二者交换位置；

(8). 95 和 80 比较，95 比 80 大，二者交换位置。此时，95 已经落在最终的位置上，第一趟冒泡排序结束。

整个排序过程如下图所示。

其过程如图 7.1 所示。



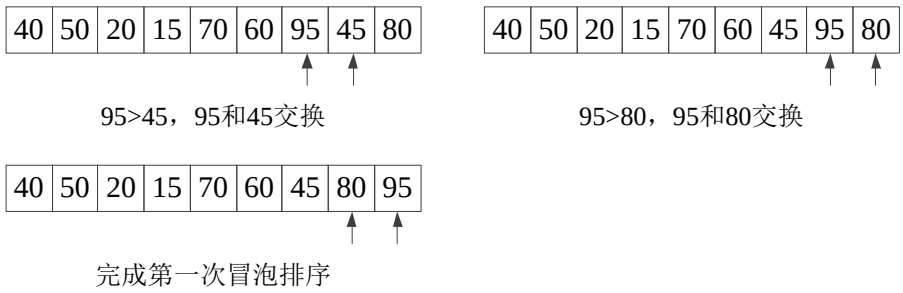


图 7.1

从上面的第一趟冒泡排序过程可知，第一趟排序需进行相邻记录的交换的次数为7次。

【参考答案】C

2. 若用冒泡排序方法对序列{10,14,26,29,41,52}从大到小排序，需进行（　　）次比较。

A. 3　　B. 10　　C. 15　　D. 25

【2013 年——厦门大学】

【考查内容】关键字序列冒泡排序过程中元素的比较次数。

【解析】第 1 题问的是第一趟排序记录的交换次数，本题则问的是冒泡排序算法完整的排序过程关键字的比较次数。其实，每一趟排序过程中，未有序的部分序列的两个相邻关键字都要进行一次比较。为了说明清楚问题，我们详解一下，给大家加深一点印象。

对序列{10,14,26,29,41,52}从大到小排序，过程如下：

第一趟排序过程中，52 和 41 进行比较，因为 52 大于 41，所以二者交换位置；52 和 29 进行比较，因为 52 大于 29，所以二者交换位置；52 和 26 进行比较，因为 52 大于 26，所以二者交换位置；52 和 14 进行比较，因为 52 大于 14，所以二者交换位置；52 和 10 进行比较，因为 52 大于 10，所以二者交换位置。第一趟排序结束，52 落在第一个位置。总共比较了 5 次。

第二趟排序，从未有序的序列{10,14,26,29,41}中选择最大元素放在第二个位置，比较方法同第一趟排序，需比较 4 次。

同理，第 3、4、5 趟排序分别比较了 3、2、1 次。所以，整个排序过程总共比较了 1+2+3+4+5=15 次。

【参考答案】C

【经典总结】冒泡排序的比较次数只与关键字的个数有关，与关键字序列的初始状态无关。对于长度为 n 的序列，排序要进行 n-1 趟，分别需要比较 n-1,n-2，…，3,2,1 次。整个排序过程总共比较 1+2+3+…+(n-1)=n(n-1)/2 次。

7.3 简单选择排序

温馨提示：本考点主要考查简单选择排序下的比较次数和移动次数、排序的时间复杂度和稳定性，也可能考查简单选择排序算法的编写，请同学们多注意这些常见的考查点。

1. 采用简单选择排序，比较次数与移动次数分别为（　　）。

A. O(n), O(logn)　　B. O(logn), O(n*n)

C. O(n*n), O(n)　　D. 0(nlogn), O(n)

【模拟题】

【考查内容】简单选择排序的比较次数和移动次数。

【解析】简单选择排序考点的内容比较简单，在历年的考研真题上出现得很少。简单选择排序的比较次数 KCN 与对象的初始排列无关。设整个待排序对象序列有 n 个对象，则第 i 趟选择具有最小排序码对象所需的比较次数总是 n-i-1 次。总的排序码比较次数为：

$$KCN=(n-1) + (n-2) +\ldots\ldots+ 2 + 1 =n(n-1)/2$$

当这组对象的初始状态是按其关键字从小到大有序的时候， 对象的移动次数达到最少，此时 RMN = 0。最坏情况是每一趟都要进行交换，总的对象移动次数为 RMN = 3(n-1)。

故而比较次数时 $O(n^2)$，移动次数是 O(n)。

【参考答案】C

【经典补充】

利用数组实现时，有如下实现方法：

表 7.1

序号	1	2	3	4	5	6	7	8
数据	49	38	65	97	76	13	27	49

步骤如下：

(1). 先找到最小的数 13，和第一个数 49 交换；

(2). 在 2~8 中找到最小的数 27，和第二个数 38 交换；

(3). 在 3~8 中找到最小的数 38，与第三个数 65 交换；

…

如此进行下去，直到数组中的元素递增有序。

7.4 希尔排序（shell sort）

温馨提示：本考点主要有两种命题方式：1、希尔排序的增量选择（相对较少）；2、对关键字记录进行希尔排序（相对较多）。请同学们抓住复习重点，提高复习效率。

1. 对记录的关键字为{50 , 26 , 38 , 80 , 70 , 90 , 8 , 30 , 40 , 20 }进行排序，各趟排序结束时的结果为：

50 , 26 , 38 , 80 , 70 , 90 , 8 , 30 , 40 , 20
50 , 8 , 30 , 40 , 20 , 90 , 26 , 38 , 80 , 70
26 , 8 , 30 , 40 , 20 , 80 , 50 , 38 , 90 , 70
8 , 20 , 26 , 30 , 38 , 40 , 50 , 70 , 80 , 90

其使用的排序方法是（　　）。

A. 快速排序　　B. 冒泡排序　　C. 希尔排序　　D. 插入排序

【模拟题】

【考查内容】希尔排序。

【解析】快速排序的第一个元素往往作为枢轴元素。第一趟排序完毕，枢轴元素大于其左边所有元素，小于其右边所有元素。很显然，题中的排序不符合快速排序的特点，排除 A 答案。

对于冒泡排序，第一趟排序结束，必有一个最大或者最小的元素落在最终位置。很显然，本题的排序方法也不是冒泡排序。

插入排序从未排序序列中依次取出元素与已排序序列中的元素进行比较，将其放入已排序序列的正确位置上，直到所有对象全部插入为止。很显然，本题也不是插入排序。

本题只剩下希尔排序了，事实上，本题是利用增量为 d=5、3、1 来对关键字{50 , 26 , 38 , 80 , 70 , 90 , 8 , 30 , 40 , 20 }进行希尔排序。其排序过程如图 7.2 所示。

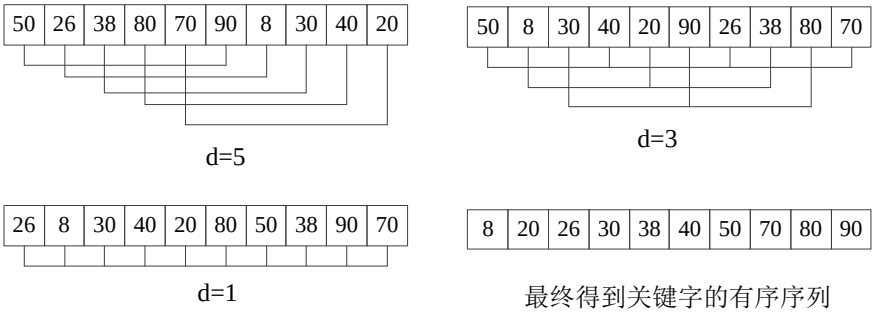


图 7.2

由上图可以看出，对关键字{50 , 26 , 38 , 80 , 70 , 90 , 8 , 30 , 40 , 20 }进行希尔排序，各趟排序的结果跟题中所给的各趟排序结果一致，故而是希尔排序。

【参考答案】C

7.5 快速排序

温馨提示：快速排序主要考查两点：1、快速排序算法的特点；2、快速排序算法实现；3、快速排序的过程或者一趟排序的结果。本考点历年考查很多，是复习的重点，请同学们务必掌握。

1. 采用递归方式对顺序表进行快速排序。下列关于递归次数的叙述中，正确的是（　　）。

A. 递归次数与初始数据的排列次序无关

B. 每次划分后，先处理较长的分区可以减少递归次数

C. 每次划分后，先处理较短的分区可以减少递归次数

D. 递归次数与每次划分后得到的分区的处理顺序无关

【2010 年统考——第 10 题】

【考查内容】快速排序的趟数与分区处理次序的关系。

【解析】快速排序可以利用递归算法来实现，这也是我们常见的方法。递归次数与关键字的初始排列有关。如果每一次划分后分区的关键字个数比较平衡，则递归次数少；如果划分后分区的关键字个数不平衡，则递归次数多。举一个简单的例子：我们常说快速排序对于随机分布的无序的关键字序列进行排序的效率最高，对基本有序或者已经有序的序列排序的效率最低，也是这个原因。

但是，要特别注意，递归次数与处理顺序无关。不同的处理顺序并不影响递归次数，每一次排序枢轴元素都将分区划分成更小的两个分区，而枢轴元素不再参加下一轮的排序过程。

【参考答案】D

2. 为实现快速排序算法，待排序序列宜采用的存储方式是（　　）。

A. 顺序存储　　B. 散列存储　　C. 链式存储　　D. 索引存储

【2011 年统考——第 10 题】

【考查内容】快速排序的关键字存储结构。

【解析】快速排序算法在排序的过程中，会从后往前查找比枢轴元素小的元素，也会从前往后查找比枢轴元素大的元素，最终使得枢轴元素大于其左边元素，小于其右边元素。散列存储、链式存储、索引存储，都不能适合作为快速排序的存储结构。与之相反，顺序存储结构更加方便从前往后和从后往前查找，能够很好的适用于快速排序。

【参考答案】A

3. 下列选项中，不可能是快速排序第 2 趟排序结果的是（　　）。

A. 2,3,5,4,6,7,9　　B. 2,7,5,6,4,3,9

C. 3,2,5,4,7,6,9　　D. 4,2,3,5,7,6,9

【2014 年统考——第 11 题】

【考查内容】快速排序。

【解析】我们都知道，每一趟快速排序都有枢轴元素落在最终位置上，枢轴元素左边和右边的元素分别小于和大于该枢轴元素。第 2 趟排序结束，应该有 2 个元素落在最终的位置上。第 i 趟排序结束，应该有 i 个元素落在最终的位置上。

A 答案中，2、3、6、7、9 中任意两个元素都显然满足条件。B 答案中，2、9 显然也都满足条件。C 答案中，只有一个 9 满足条件。D 答案中，5、9 满足条件。显然，C 答案是唯一一个不可能时快速排序第 2 趟排序的结果。

【参考答案】C

4. 给定关键字的集合为{20,15,14,18,21,36,40,10}，一趟快速排序结束时关键字的排列为（　　）。

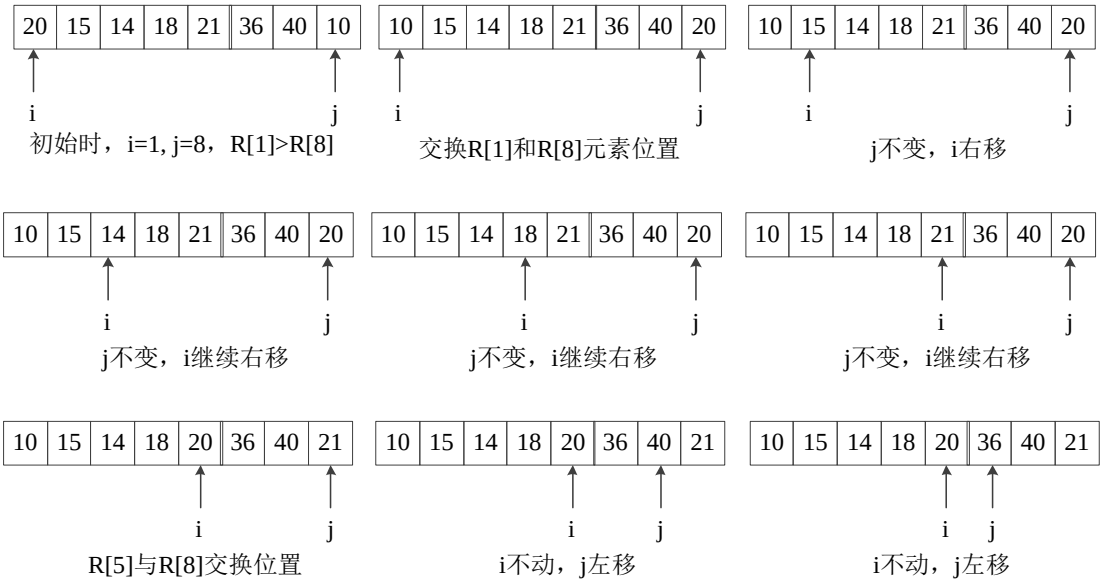
A. 10,15,14,18,20,36,40,21　　B. 10,15,14,18,20,40,36,21

C. 10,15,14,20,18,40,36,21　　D. 15,10,14,18,20,36,40,21

【2012——昆明理工大学】

【考查内容】快速排序。

【解析】一组初始记录关键字为(20，15，14，18，21，36，40，10) ，设该关键字存放在数组 R[8]中，数组下标为 1~8。我们来详细解析以 20 为基准记录的一趟快速排序结束后的过程，如图 7.3 所示。



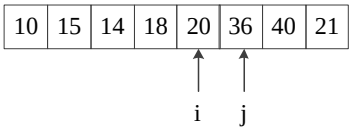


图 7.3

一趟快速排序结束之后，可知得到序列为 10、15、14、18、20、36、40、21，故而选择 A 答案。

【参考答案】A

7.6 堆排序

温馨提示：堆排序主要命题有两种方式：1、堆排序的特点；2、堆排序的过程（要求同学们手工操作），包括堆的建立和输出、调整过程。堆排序过程希望同学们会手工操作。

1. 已知关键字序列 5，8，12，19，28，20，15，22 是小根堆（最小堆），插入关键字 3，调整后得到的小根堆是（　　）。

   A. 3，5，12，8，28，20，15，22，19

   B. 3，5，12，19，20，15，22，8，28

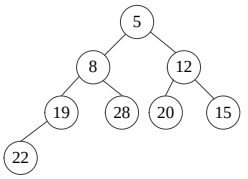
   C. 3，8，12，5，20，15，22，28，19

   D. 3，12，5，8，28，20，15，22，19

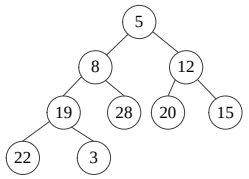
【2009 年统考——第 9 题】

【考查内容】小根堆的调整操作。

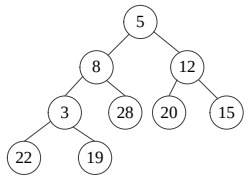
【解析】首先，我们画出给定关键字序列 5，8，12，19，28，20，15，22 对应的小根堆，如图 7.4 所示。



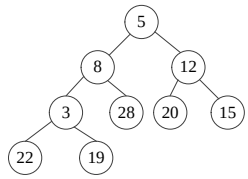
初始堆



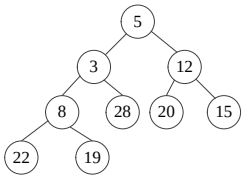
向堆中插入关键字3



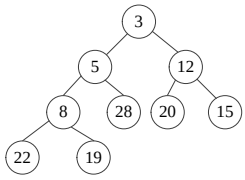
从第1个非叶子结点19开始调整堆



调整12



调整8，8与3交换



调整5，5与3交换，完成堆的调整

图 7.4

从上图可以看到，堆调整完成之后得到的小根堆为 3、5、12、8、28、20、15、22、19。故而，选择 A 答案。

注意，有的书只对插入新结点到堆顶的路径进行调整。这样做可以节省时间，减少比较次数。

【参考答案】A

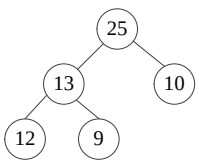
2. 已知序列 25，13，10，12，9 是大根堆，在序列尾部插入新元素 18，将其再调整为大根堆，调整过程中元素之间进行的比较次数是（　　）。

A. 1　　B. 2　　C. 4　　D. 5

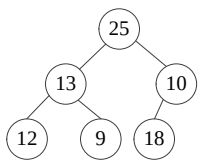
【2011 年统考——第 11 题】

【考查内容】大根堆的调整。

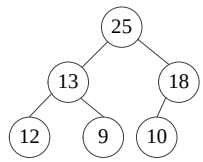
【解析】在由关键字 25，13，10，12，9 构成的大根堆中，插入新元素 18，过程如下图 7.5 所示。



初始堆



插入关键字18



调整第一个非叶子结点18

图 7.5

上图 7.5 中，插入新元素 18 之后，18 与其父节点 10 进行比较，18>10。于是，将 18 与 10 交换。18 继续与父节点 25 比较，发现 18 小于 25，所以 18 和 25 不需要交换。综上，总共发生了 2 次比较。

【参考答案】B

3. 如果只想得到 2009 个元素组成的序列中的前 3 个最小元素，那么用（　　）方法最快。

A. 起泡排序　　B. 直接选择排序

C. 堆排序　　D. 快速排序

【2010 年——湖南大学】

【考查内容】堆排序的优点。

【解析】堆排序最适合从大量的元素中抽取少许最大的或者最小的元素。堆排序的前期工作堆调整，需要耗费时间。一旦调整完毕，每一次排序都可以从堆顶取下最大（或者最小）的元素。

【参考答案】C

4. 下面 4 个序列中，（　　）是堆。

A. 75,65,30,15,25,45,20,10　　B. 75,65,45,10,30,25,20,15

C. 75,45,65,30,15,25,20,10　　D. 75,45,65,10,25,30,20,15

【2013 年——中南大学】

【考查内容】堆定义的应用。

【解析】根据堆的定义，我们知道大顶堆中，任意结点的关键字都小于其父节点的关键字，而同一个父节点的两个结点之间的谁大谁小都可以。我们将题目所给 4 个选项对应的完全二叉树画成图，如图 7.6 所示。

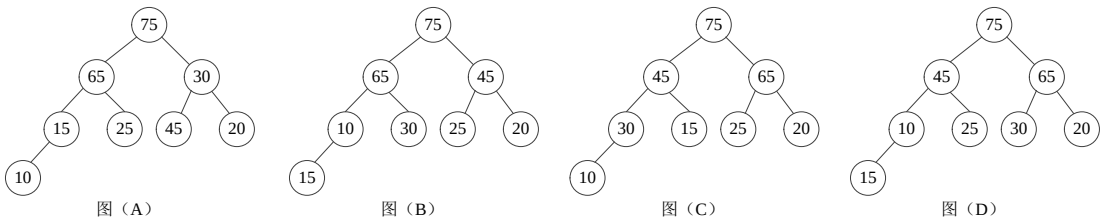


图 7.6

可以看见，上图 7.8 中 4 个选项中，A 答案中 75>65，且 75>30,但是 30<45，所以不是堆；B 答案中，75>65，且 75>45,但是 10 小于 15，所以也不是堆；C 答案是大顶堆；D 答案中，75>65，且 75>45,但是 10 小于 15，所以不是堆。

【参考答案】C

5. 设使用堆排序法对关键字序列 T=(10, 27, 5, 50, 60, 7, 40, 43, 75）进行排序：

(1). 画出初始大根堆对应的完全二叉树；

(2). 写出大根堆序列；

(3). 画出第一趟排序后新堆对应的完全二叉树。

【2013 年——暨南大学】

【考查内容】堆排序。

【解析】堆排序大的综合题出现的可能性并不大，但是选择题上出题的可能性还是比较大的。几年的真题里面考查的都是堆调整，同学们多注意一下。

(1). 我们先画出初始的完全二叉树，然后在进行调整，使其成大根堆，过程如图 7.7 所示。

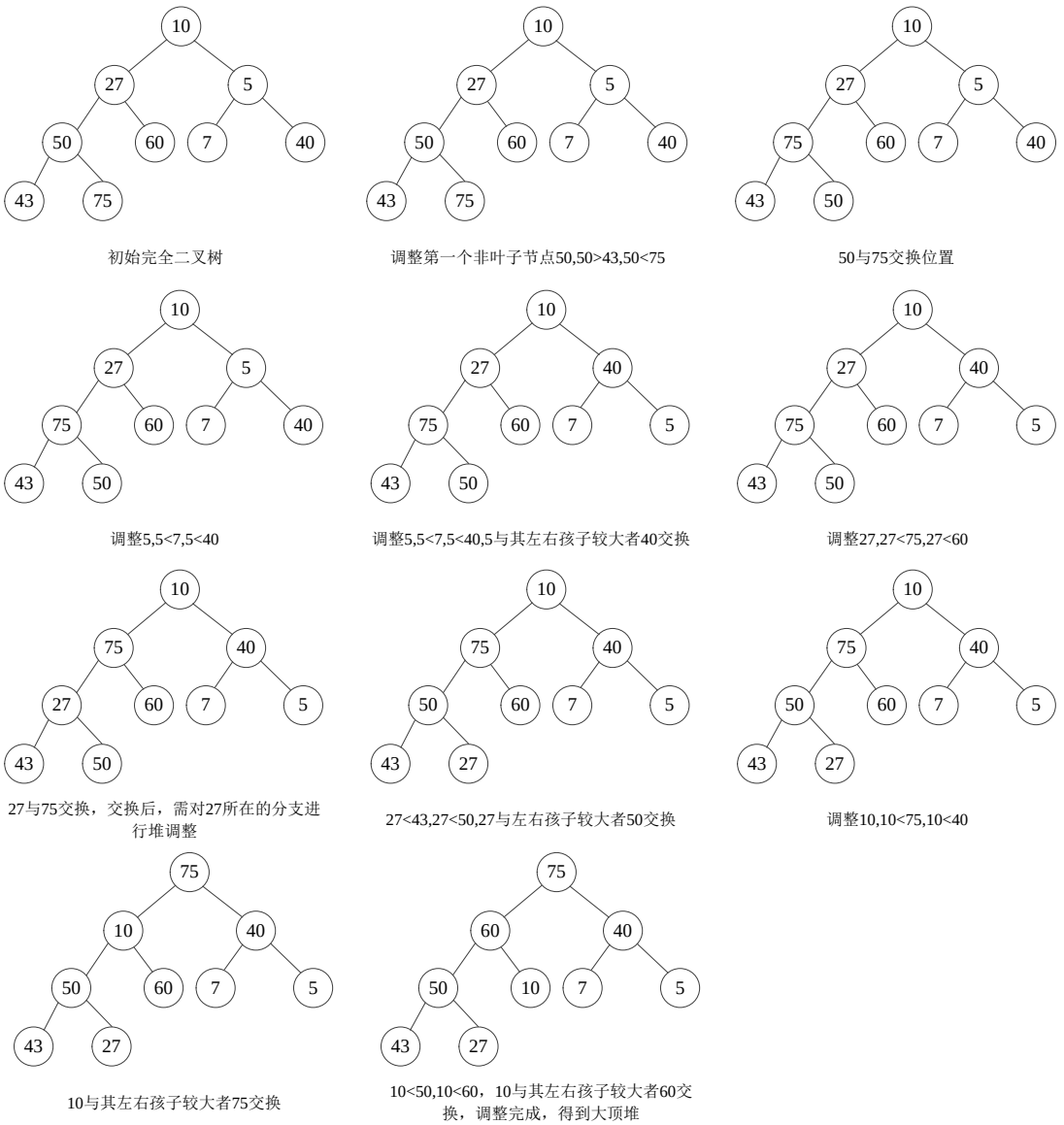


图 7.7

大顶堆对应的完全二叉树如上图最后一个小图所示，即进行堆调整完成自后得到的大顶堆。

(2). 根据(1)中的大根堆调整过程，可知大根堆序列为：{75,60,40,50,10,7,5,43,27}。

(3). 对(1)中的大根堆进行第一趟排序，输出堆顶元素 75，再调整堆，过程如图 7.8 所示。

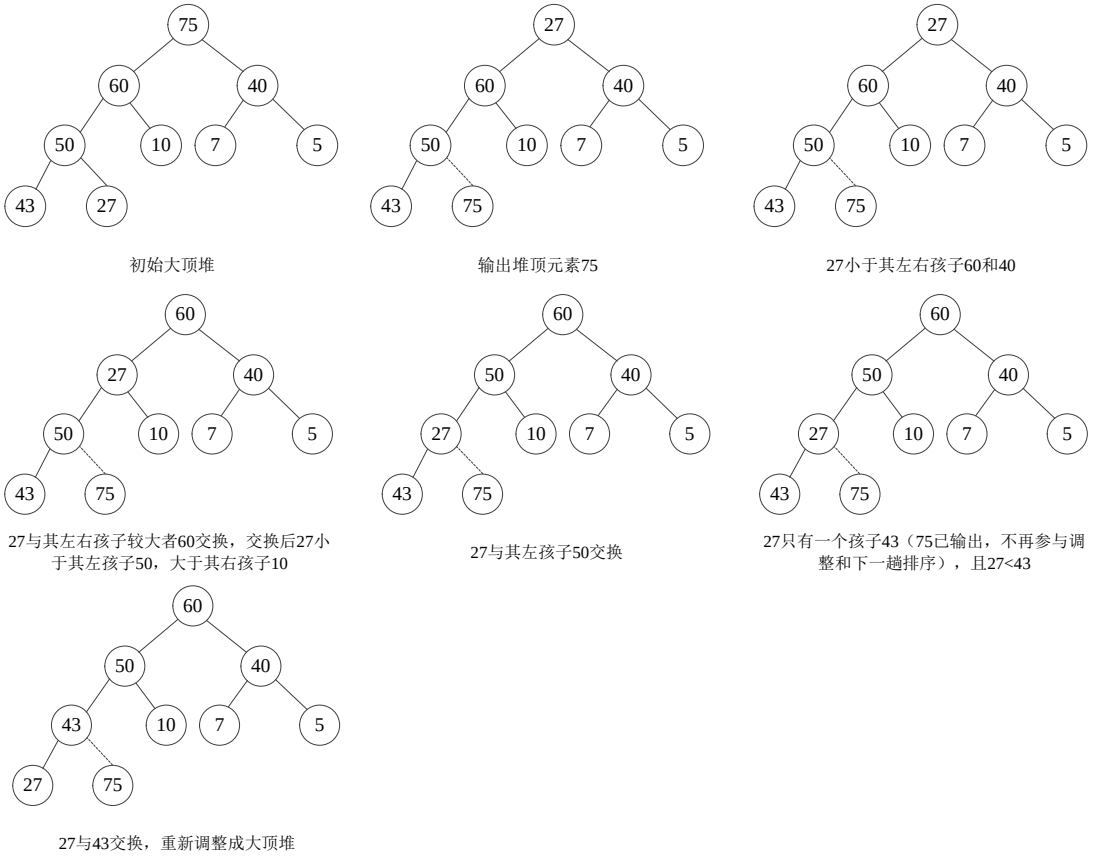


图 7.8

6. 对关键码{46,79,56,38,40,84}采用堆排序,则初始化堆(小堆)后最后一个元素是（　　）。

A. 84

B. 46

C. 56

D. 38

【2014 年—武汉科技大学】

【考查内容】堆的建立。

【解析】依题意，我们给出了小顶堆的建立过程，如下图所示。

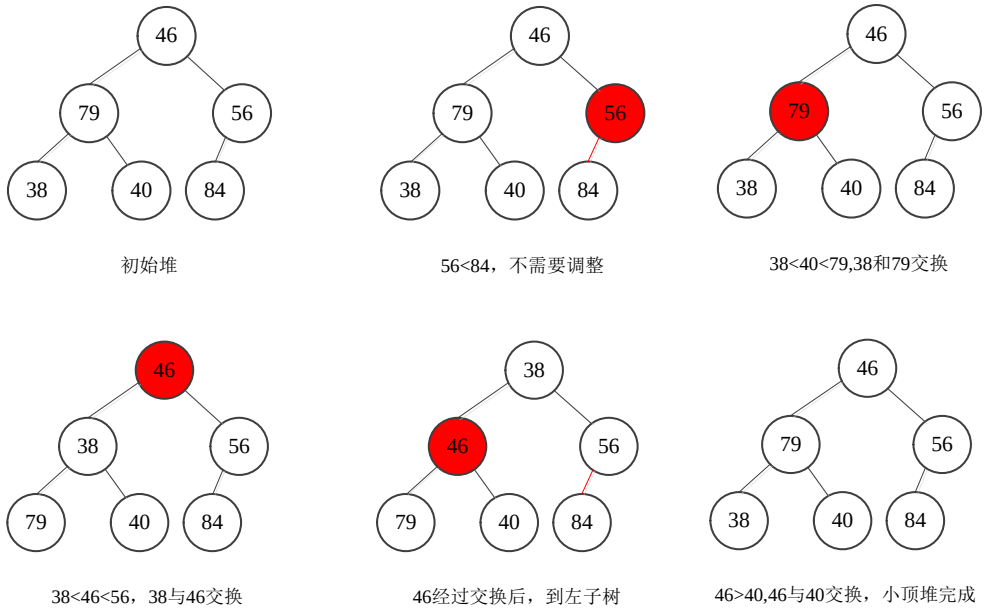


图 7.9

从最终构建的小顶堆，可知该堆的最后一个元素是 84。

【参考答案】A

7. 简单叙述堆排序算法（HeapSort）的基本思想。按“由小到大”排序所给的数值，并按排序步骤列出每次已堆化好的待排数值序列。可不列出“已排好”的数值和堆化过程中的堆形状或者数值序列。如果给额外信息，将判断这些信息的正确性。

初始已堆化好的待排数值序列：93, 67,78,15,34,40,33

【2013 年—中山大学】

【考查内容】堆排序的基本思想，堆排序过程。

【解析】堆排序的基本思想（以大顶堆为例）为：

① 先将初始文件 R[1..n]建成一个大根堆，此堆为初始的无序区

② 再将关键字最大的记录 R[1](即堆顶)和无序区的最后一个记录 R[n]交换，由此得到新的无序区 R[1..n-1]和有序区 R[n]，且满足 R[1..n-1].keys≤R[n].key

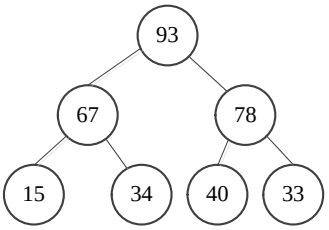
③ 由于交换后新的根 R[1]可能违反堆性质，故应将当前无序区 R[1..n-1]调整为堆。然后再次将 R[1..n-1]中关键字最大的记录 R[1]和该区间的最后一个记录 R[n-1]交换，由此得到新的无序区 R[1..n-2]和有序区 R[n-1..n]，且仍满足关系 R[1..n-2].keys≤R[n-1..n].keys，同样要将 R[1..n-2]调整为堆。 ……

直到无序区只有一个元素为止。

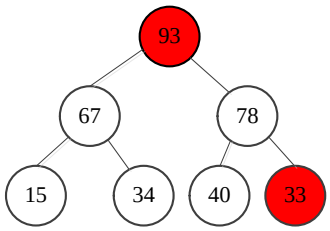
从大顶堆的堆排序过程可以看出，堆顶的元素是下一个待输出的元素，这个元素存放在数组 R 的未有序区的末尾。显然，每一趟排序，未有序的序列中的最大的元素被输出。

所以，堆排序将最大的元素放在 R[n]，将次最大的元素放在 R[n-1]，…，将最小的元素放在 R[1]。所以，大顶堆的堆排序所得到的序列就是一个从小到大的序列。很多书没有讲到这个，或者给出了错误的解释，这里我们进行纠正。

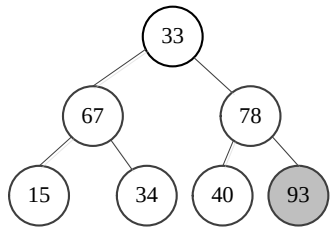
题目给出的初始堆其实已经是大顶堆。我们将堆排序的过程给图如下。



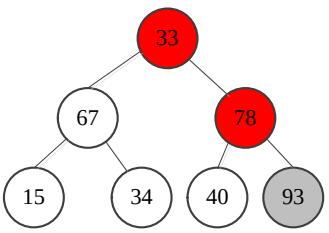
初始大顶堆



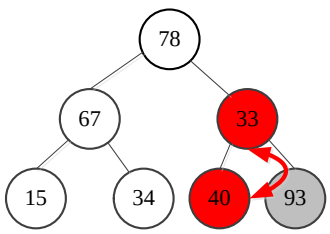
输出93,93与最后一个元素33交换



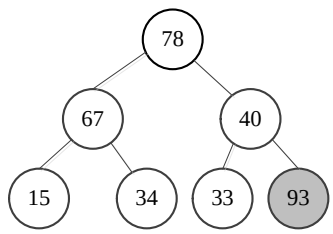
调整堆，使其成为新的大顶堆



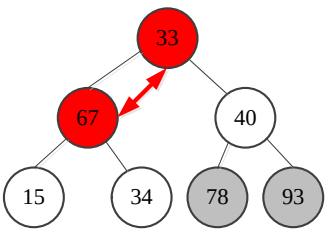
33小于67和78，和78交换



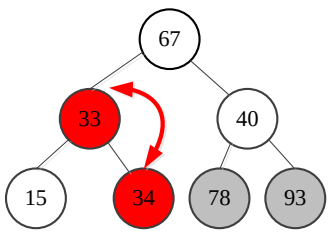
33小于40,33和40交换



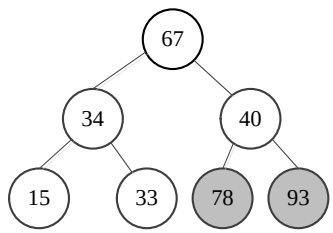
形成新的大顶堆



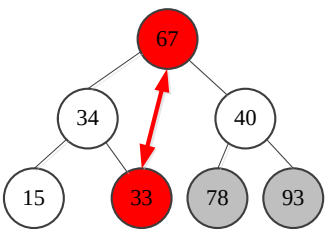
33小于40和67，与67交换



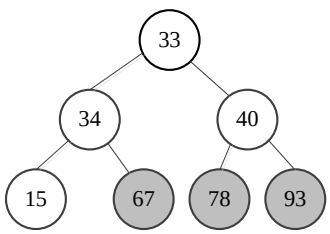
33小于34，和34交换



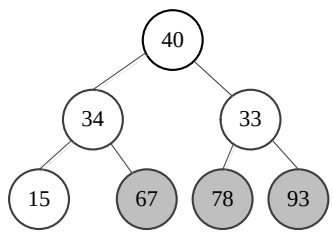
形成新的大顶堆，输出67



33与67交换



33小于40，与40交换



形成新的大顶堆

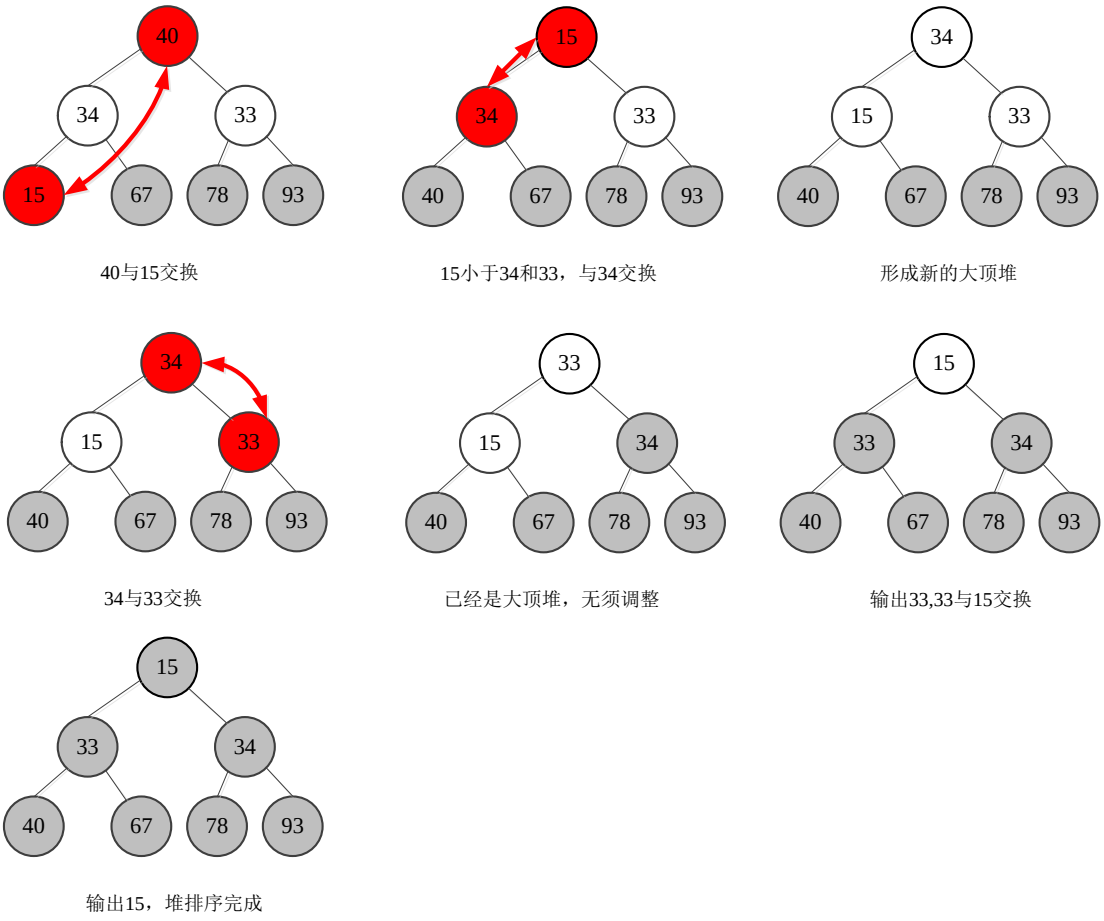


图 7.10

7.7 归并排序（merge sort）

温馨提示：本考点主要考查归并排序的方法和某一趟归并排序的结果，希望同学们会手工对关键字进行归并排序。

1. 以整数序列{26,59,77,31,51,11,19,42}，以二路归并排序从小到大排序，第一趟的归并结果为（　　）。

   A. 31,51,11,42,26,77,59,19　　B. 26,11,19,31,51,59,77,42

C. 11,19,26,31,42,59,51,77　　D. 26,59,31,77,11,51,19,42

【2012 年——青岛理工大学】

【考查内容】归并排序的原理。

【解析】归并排序在我们所学的排序算法中，算是比较简单的一个了。我们来看看整数序列{26,59,77,31,51,11,19,42}的归并排序过程，如图 7.9 所示。

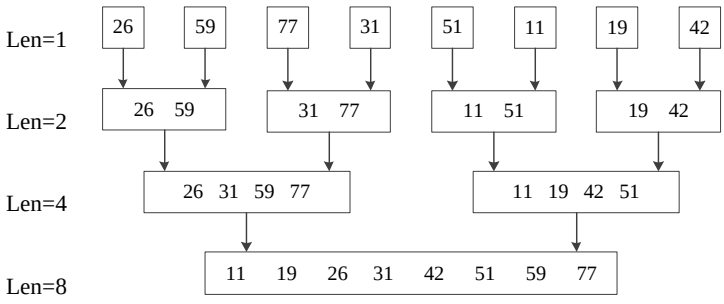


图 7.11

从上图 7.11 可以看出，第一趟归并排序的结果对应 Len=2 的序列{26,59,31,77,11,51,19,42}。

【参考答案】D

7.8 基数排序

温馨提示：本考点主要考查桶排序的方法，可能会要求同学们手工操作，并给出某一趟分配和收集的结果。

1. 对给定的关键字序列 110，119，007，911，114，120，122 进行基数排序，则第 2 趟分配收集后得到的关键字序列是（　　）。

A. 007，110，119，114，911，120，122

B. 007，110，119，114，911，122，120

C. 007，110，911，114，119，120，122

D. 110，120，911，122，114，007，119

【2013 年统考——第 11 题】

【考查内容】基数排序。

【解析】对给定的关键字进行一趟基数排序，排序的结果，使得最低位有序，即个位有序，对应的排序过程如图 7.12 所示。

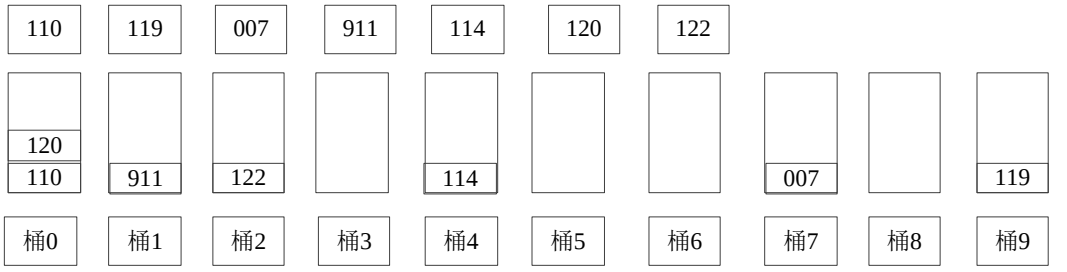


图 7.12

对上图的基数排序结果进行收集，得到图 7.13 的排序结果。

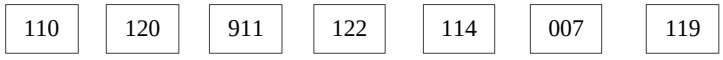


图 7.13

再按照十位数进行第二趟排序（本趟排序结果即为题目中的正确答案），过程如 7.14 图所示。

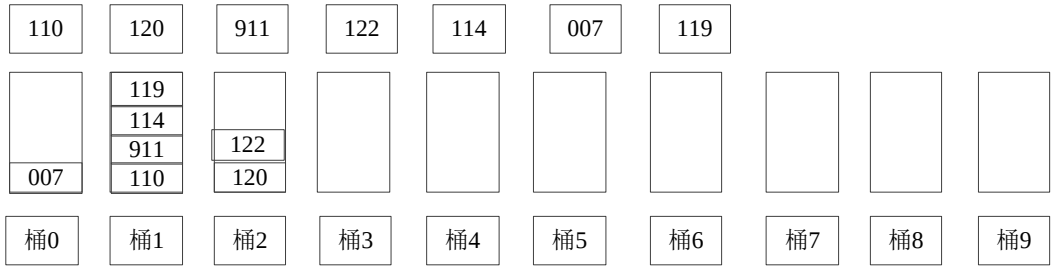


图 7.14

对第二趟基数排序结果进行收集，得到如图 7.15 的排序结果。

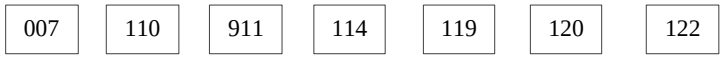


图 7.15

再进行第三趟排序，该趟排序的依据是百分位的大小。该趟排序过程如图 7.16 所示。

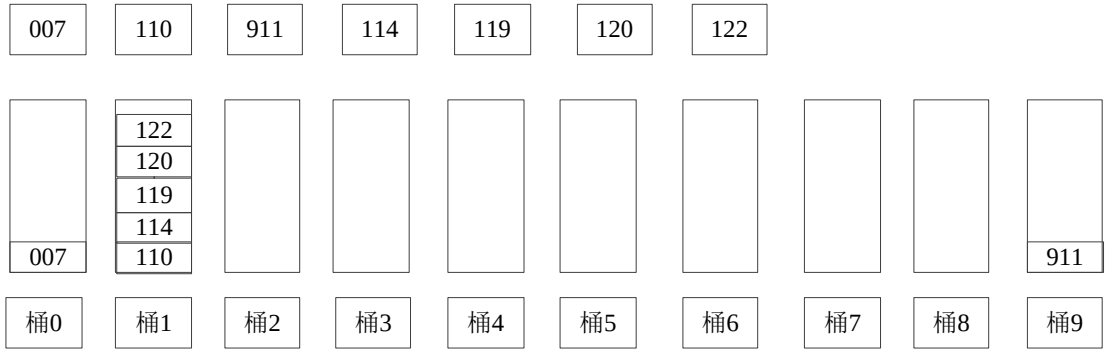


图 7.16

对第三趟排序结束后，各个桶中的数据进行收集。要注意，桶其实上是一个队列，满足先进先出的规则。在梦享团队所著的数据结构图书中，桶的队头在下方，队尾在上方。我们进行第三趟收集，得到序列{007,110,114,119,120,122,911}，显然已经是一个有序序列。

【参考答案】C

7.9 各种内部排序算法的比较和应用

温馨提示：本题主要考查各种内部排序的区别，请同学们掌握本章几种常见的内部排序算法的特点、稳定性和时间复杂度等方面的知识。

1. 在内部排序过程中，对尚未确定最终位置的所有元素进行一遍处理称为一趟排序。下列排序方法中，每一趟排序结束都至少能够确定一个元素最终位置的方法是（　　）。

Ⅰ．简单选择排序　　Ⅱ．希尔排序　　Ⅲ．快速排序

Ⅳ．堆排序　　Ⅴ．二路归并排序

A. 仅Ⅰ、Ⅲ、Ⅳ　　B. 仅Ⅰ、Ⅲ、Ⅴ

C. 仅Ⅱ、Ⅲ、Ⅳ　　D. 仅Ⅲ、Ⅳ、Ⅴ

【2012 年统考——第 10 题】

【考查内容】常见排序算法的特点、定义。

【解析】简单选择排序算法每次都从未有序的序列中选择最小的元素放在最终的位置上，直到所有元素最终有序。希尔排序以增量方式对子表进行排序，每一趟排序结束，子表内的元素有序，但未必有元素能放在其最终位置上。每轮快速排序结束，枢轴元素都可以放在最终的位置上。堆排序每一趟排序都将输出当前堆中最大的或者最小的元素，输出的元素放在最终的位置上。二路归并排序每一趟归并结束，都会形成若干有序的子表，但未必有元素回落在最终的位置上。

【参考答案】A

2. 将上万个一组无序并且不相等的正整数序列，存放于顺序存储结构中，采用（　　）方法能够最快地查找出其中最大的正整数。

A. 快速排序　　B. 插入排序　　C. 选择排序　　D. 归并排序

【2012 年——华南理工大学】

【考查内容】选择排序的特点。

【解析】我们假设 10000 个元素的无序的不相等的正整数序列，存放在顺序存储结构中。采用插入排序、快速排序、归并排序都不能在一趟排序之后确定任何一个元素的最终

位置，需要进行多次排序才能将最大的正整数查找出来。直接选择排序仅需要一趟排序，对顺序表进行一次遍历比较就能找到最大的正整数。

【参考答案】C

3. 下列排序方法中，平均情况下的时间复杂度是 O(nlogn)且稳定的方法是（　　）。

A. 归并排序　　B. 快速排序

C. 简单插入排序　　D. 堆排序

【2013 年——华南理工大学】

【考查内容】排序算法的时间复杂度和稳定性。

【解析】在本考点中，我们需要总结一下常见排序算法的平均时间复杂度和稳定性，如下表所示（该表在第 5 题中也有应用）。

表 7.2

排序方法	平均时间	最坏情况	辅助空间	稳定性
直接插入排序	$O(n^2)$	$O(n^2)$	O(1)	稳定
折半插入排序	$O(n^2)$	$O(n^2)$	O(1)	稳定
起泡排序	$O(n^2)$	$O(n^2)$	O(1)	稳定
直接选择排序	$O(n^2)$	$O(n^2)$	O(1)	稳定
希尔排序	$O(n^{1.3})$	$O(n^{1.3})$	O(1)	不稳定
快速排序	$O(n\log_2 n)$	$O(n^2)$	$O(\log_2 n)$	不稳定
堆排序	$O(n\log_2 n)$	$O(n\log_2 n)$	O(1)	不稳定
2-路归并排序	$O(n\log_2 n)$	$O(n\log_2 n)$	O(n)	稳定
基数排序	O(d*(rd+n))	O(d*(rd+n))	O (rd)	稳定

根据上表可知，平均时间复杂度为 $O(n\log_2 n)$的常见排序算法有快速排序、堆排序和归并排序，但是这三个算法中只有归并排序是稳定的。

【参考答案】A

［注意］有的教材认为简单（直接）选择排序属于不稳定的排序算法，我们参照严蔚敏教材中简单（直接）选择排序的思想，把该排序算法当做稳定的排序算法。

4. 稳定的排序算法是（　　）。

A. 快速排序　　B. 希尔排序

C. 堆排序　　D. 插入排序

【2010 年——中山大学】

【考查内容】常见算法的稳定性。

【解析】排序算法的稳定性通俗地讲就是能保证排序前 2 个相等的数其在序列的前后位置顺序和排序后它们两个的前后位置顺序相同。从算法的稳定性而言，所有排序算法可分为两类：

(1). 插入排序、冒泡排序、归并排序和基数排序为第一类，均属于稳定的算法。

(2). 希尔排序、快速排序和堆排序则属于第二类，是不稳定算法。

【参考答案】D

5. 下列排序算法中，平均时间复杂度最小的是（　　）。

A. 归并排序　　B. 起泡排序

C. 简单选择排序　　D. 直接插入排序

【2014 年——中国计量学院】

【考查内容】常见算法的时间复杂度比较。

【解析】我们对各个常见的排序算法的平均时间复杂度、最坏情况下的时间复杂度和辅助空间大小做一个总结，如**表 7.2（本考点第三题）**所示。

由**表 7.2（本考点第 3 题）**可知，直接插入排序、简单选择排序和起泡（冒泡）排序的平均时间复杂度为 $O(n^2)$，而归并排序的平均时间复杂度为 $O(n\log_2 n)$。

【参考答案】A

6. 下列排序方法在排序过程中，关键码比较的次数与记录的初始排列顺序无关的是（　　）。

A. 直接插入排序和快速排序

B. 快速排序和归并排序

C. 直接选择排序和归并排序

D. 直接插入排序和归并排序

【2012 年——昆明理工大学】

【考查内容】常见算法的特点，关键字比较次数与记录初始状态无关的算法。

【解析】我们说，快速排序在关键字基本无序时，算法的性能最好，在关键字基本有序时，算法的性能最差。显然，快速排序在排序过程中关键字的比较次数与记录的初始排列顺序有关。

直接插入排序我们也可以列举两种极端的情况，即元素递增有序和递减有序的情况。在元素递增有序的情况下，每一个元素只需要比较一次即可。在元素递减有序的情况下，每一个元素都需要与它所在位置之前所有元素进行比较。显然也与关键字的初始排列顺序有关。

【参考答案】C

“梦享考研系列”辅导书——答疑解惑

尊敬的读者朋友！

您好！首先，欢迎您使用我梦享团队编写的“梦享考研系列”辅导书，我们衷心地感谢您的支持！

让每一个考研人圆梦，是我们团队的梦想；默默陪伴着您走过每一个寻梦的日日夜夜，做您不离不弃的朋友，是我们的愿望。

宝剑锋从磨砺出，梅花香自苦寒来。每一个甜蜜的果实，都在挥洒汗水之后才能摘取到。对于每一个考生来说，一本绝佳的考研辅导书，就像一把利剑，拥有这把利剑，我们能够百战百胜，所向披靡。我们深知我们团队的能力有限，但我们一直希望，我们这套书，就是大家手中的那一把利剑！为此，我们团队特意向各位考生征求好的教材写法，好的题目解析方法等，希望把大家的精华汇聚到一本书里，为更多的考生提供更好的辅导书。

除了“梦享考研系列”辅导书之外，梦享团队致力于给大家更好的考研服务。为了给广大读者提供一个答疑和交流的平台，梦享团队开通了微信号、微信公众号和新浪微博账号等三种平台。欢迎大家来跟我们交流，一起成长。

梦享团队微信号　　梦享团队官方微信公众号　　梦享团队新浪微博

因为有你，所以有梦享！

梦享团队祝愿每一个考研人梦想成真！

参考文献

[1] 汤子瀛. 计算机操作系统[M]. 西安：西安电子科技大学出版社，2001.

[2] Tanenbaum A.S. 现代操作系统[M]. 北京：机械工业出版社，2009.

[3] 翔高教育. 计算机学科专业基础综合习题精编[M]. 上海：复旦大学出版社，2010.

[4] 崔巍，等. 计算机学科专业基础综合辅导讲义[M]. 北京：原子能出版社，2011.

[5] 严蔚敏. 数据结构（C 语言版）[M]. 北京：清华大学出版社，2007.

[6] 严蔚敏，等. 数据结构题集（C 语言版）[M]. 北京：清华大学出版社，1999.

[7] 谢希仁. 计算机网络（第 5 版）[M]. 北京：电子工业出版社，2008.

[8] 谢希仁. 计算机网络释疑与习题解答[M]. 北京：电子工业出版社，2011.

[9] 唐朔飞. 计算机组成原理（第 2 版）[M]. 北京：高等教育出版社，2008.

[10] 唐朔飞.计算机组成原理学习指导与习题解答（第 2 版）[M]. 北京：高等教育出版社，2012.

[11] 白中英. 计算机组成原理（第四版）[M]. 北京：科学出版社，2008.

[12] 蒋本珊. 计算机组成原理（第三版）[M]. 北京：清华大学出版社，2013.

[13] 梦享团队. 2016 年考研核心考点命题思路解密——数据结构[M]. 北京: 北京邮电大学出版社，2015.

[14] 梦享团队. 2016 年考研核心考点命题思路解密——计算机组成原理[M]. 北京: 北京邮电大学出版社，2015.

[15] 梦享团队. 2016 年考研核心考点命题思路解密——计算机操作系统[M]. 北京: 北京邮电大学出版社，2015.

[16] 梦享团队. 2016 年考研核心考点命题思路解密——计算机网络[M]. 北京: 北京邮电大学出版社，2015.