第一章 绪论

一、单项选择题

1. 数据结构是指（ ）。

A. 数据元素的组织形式　　B. 数据类型

C. 数据存储结构　　D. 数据定义

2. 数据在计算机存储器内表示时，物理地址与逻辑地址不相同的，称之为（ ）。

A. 存储结构　　B. 逻辑结构

C. 链式存储结构　　D. 顺序存储结构

3. 树形结构是数据元素之间存在一种（ ）。

A. 一对一关系　　B. 多对多关系

C. 多对一关系　　D. 一对多关系

4. 设语句 x++的时间是单位时间，则以下语句的时间复杂度为（ ）。

```
for(i=1; i<=n; i++)
    for(j=i; j<=n; j++)
        x++;
```

A. O(1)　　B. $O(n^2)$　　C. O(n)　　D. $O(n^3)$

5. 算法分析的目的是（1），算法分析的两个主要方面是（2）。

（1） A. 找出数据结构的合理性　　B. 研究算法中的输入和输出关系

C. 分析算法的效率以求改进　　D. 分析算法的易懂性和文档性

（2） A. 空间复杂度和时间复杂度　　B. 正确性和简明性

C. 可读性和文档性　　D. 数据复杂性和程序复杂性

6. 计算机算法指的是（1），它具备输入，输出和（2）等五个特性。

（1） A. 计算方法　　B. 排序方法

C. 解决问题的有限运算序列　　D. 调度方法

（2） A. 可行性，可移植性和可扩充性　　B. 可行性，确定性和有穷性

C. 确定性，有穷性和稳定性　　D. 易读性，稳定性和安全性

7. 数据在计算机内有链式和顺序两种存储方式，在存储空间使用的灵活性上，链式存储比顺序存储要（ ）。

A. 低　　B. 高　　C. 相同　　D. 不好说

8. 数据结构作为一门独立的课程出现是在（ ）年。

A. 1946　　B. 1953　　C. 1964　　D. 1968

9. 数据结构只是研究数据的逻辑结构和物理结构，这种观点（ ）。

A. 正确　　B. 错误

C. 前半句对，后半句错　　D. 前半句错，后半句对

10. 计算机内部数据处理的基本单位是（　）。

A.数据　　B.数据元素　　C.数据项　　D.数据库

二、填空题

1. 数据结构按逻辑结构可分为两大类，分别是_____和_____。

2. 数据的逻辑结构有四种基本形态，分别是_____、_____、_____和_____。

3. 线性结构反映结点间的逻辑关系是_____的，非线性结构反映结点间的逻辑关系是_____的。

4. 一个算法的效率可分为_____效率和_____效率。

5. 在树型结构中，树根结点没有_____结点，其余每个结点的有且只有_____个前趋驱结点；叶子结点没有_____结点；其余每个结点的后续结点可以_____。

6. 在图型结构中，每个结点的前趋结点数和后续结点数可以_____。

7. 线性结构中元素之间存在_____关系；树型结构中元素之间存在_____关系；图型结构中元素之间存在_____关系。

8. 下面程序段的时间复杂度是_____。

```
for(i=0;i<n;i++)
   for(j=0;j<n;j++)
      A[i][j]=0;
```

9. 下面程序段的时间复杂度是_____。

```
i=s=0;
while(s<n)
{   i++;
    s+=i;
}
```

10. 下面程序段的时间复杂度是_____。

```
s=0;
for(i=0;i<n;i++)
    for(j=0;j<n;j++)
        s+=B[i][j];
sum=s;
```

11. 下面程序段的时间复杂度是_____。

```
i=1;
while(i<=n)
    i=i*3;
```

12. 衡量算法正确性的标准通常是_____。

13. 算法时间复杂度的分析通常有两种方法，即_____和_____的方法，通常我们对算法求时间复杂度时，采用后一种方法。

三、求下列程序段的时间复杂度。

1.
```
x=0;
for(i=1;i<n;i++)
    for(j=i+1;j<=n;j++)
        x++;
```

2.
```
x=0;
for(i=1;i<n;i++)
   for(j=1;j<=n-i;j++)
        x++;
```

3.
```
int i,j,k;
  for(i=0;i<n;i++)
     for(j=0;j<=n;j++)
        {   c[i][j]=0;
            for(k=0;k<n;k++)
                c[i][j]=a[i][k]*b[k][j]
        }
```

4.
```
 i=n-1;
  while((i>=0)&&A[i]!=k))
      j--;
  return (i);
```

5.
```
 fact(n)
 {   if(n<=1)
         return (1);
     else
         return (n*fact(n-1));
 }
```

第一章参考答案

一、单项选择题

1. A　2. C　3. D　4. B　5. C、A　6. C、B　7. B　8. D　9. B　10. B

二、填空题

1. 线性结构，非线性结构
2. 集合，线性，树，图
3. 一对一，一对多或多对多
4. 时间，空间
5. 前趋，一，后继，多
6. 有多个
7. 一对一，一对多，多对多

8. $O(n^2)$

9. $O(\sqrt{n})$

10. $O(n^2)$

11. $O(\log_3 n)$

12. 程序对于精心设计的典型合法数据输入能得出符合要求的结果。
13. 事后统计，事前估计

三、算法设计题

1. $O(n^2)$　2. $O(n^2)$　3. $O(n^3)$　4. $O(n)$　5. $O(n)$

第二章 线性表、栈和队列

一、单项选择题

1. 线性表是_____。

A．一个有限序列，可以为空　　B．一个有限序列，不可以为空

C．一个无限序列，可以为空　　D．一个无限序列，不可以为空

2. 在一个长度为 n 的顺序表中删除第 i 个元素(0<=i<=n)时，需向前移动____个元素。

A．n-i　　B．n-i+1　　C．n-i-1　　D．i

3. 线性表采用链式存储时，其地址_____。

A．必须是连续的　　B．一定是不连续的

C．部分地址必须是连续的　　D．连续与否均可以

4. 从一个具有 n 个结点的单链表中查找其值等于 x 的结点时，在查找成功的情况下，需平均比较_____个元素结点。

A．n/2　　B．n　　C．（n+1）/2　　D．（n-1）/2

5. 在双向循环链表中，在 p 所指的结点之后插入 s 指针所指的结点，其操作是____。

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

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

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

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

6. 设单链表中指针 p 指向结点 m，若要删除 m 之后的结点（若存在），则需修改指针的操作为_____。

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

C．p=p->next->next;　　D．p->next=p;

7. 在一个长度为 n 的顺序表中向第 i 个元素(0< i<n+1)之前插入一个新元素时，需向后移动_____个元素。

A．n-i　　B．n-i+1　　C．n-i-1　　D．i

8. 在一个单链表中，已知 q 结点是 p 结点的前趋结点，若在 q 和 p 之间插入 s 结点，则须执行

A．s->next=p->next; p->next=s

B．q->next=s; s->next=p

C．p->next=s->next; s->next=p

D．p->next=s; s->next=q

9. 以下关于线性表的说法不正确的是_____。

A．线性表中的数据元素可以是数字、字符、记录等不同类型。

B．线性表中包含的数据元素个数不是任意的。

C．线性表中的每个结点都有且只有一个直接前趋和直接后继。

D．存在这样的线性表：表中各结点都没有直接前趋和直接后继。

10. 线性表的顺序存储结构是一种_____的存储结构。

A．随机存取　　B．顺序存取　　C．索引存取　　D．散列存取

11. 在顺序表中，只要知道_____，就可在相同时间内求出任一结点的存储地址。

A．基地址　　B．结点大小

C．向量大小　　D．基地址和结点大小

12. 在等概率情况下，顺序表的插入操作要移动_____结点。

A．全部　　B．一半

C．三分之一　　D．四分之一

13. 在_____运算中，使用顺序表比链表好。

A．插入　　B．删除

C．根据序号查找　　D．根据元素值查找

14. 在一个具有 n 个结点的有序单链表中插入一个新结点并保持该表有序的时间复杂度是_____。

A．O(1)　　B．O(n)

C．$O(n^2)$　　D．O(log2n)

15. 设有一个栈，元素的进栈次序为 A, B, C, D, E,下列是不可能的出栈序列_____。

A．A, B, C, D, E　　B．B, C, D, E, A

C．E, A, B, C, D　　D．E, D, C, B, A

16. 在一个具有 n 个单元的顺序栈中，假定以地址低端（即 0 单元）作为栈底，以 top 作为栈顶指针，当做出栈处理时，top 变化为_____。

A．top 不变　　B．top=0　　C．top--　　D．top++

17. 向一个栈顶指针为 hs 的链栈中插入一个 s 结点时，应执行_____。

A．hs->next=s;

B．s->next=hs; hs=s;

C．s->next=hs->next;hs->next=s;

D．s->next=hs; hs=hs->next;

18. 在具有 n 个单元的顺序存储的循环队列中，假定 front 和 rear 分别为队头指针和队尾指针，则判断队满的条件为_____。

A．rear%n= = front　　B．（front+1）%n= = rear

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

19. 在具有 n 个单元的顺序存储的循环队列中，假定 front 和 rear 分别为队头指针和队

尾指针，则判断队空的条件为_____。

A. rear%n= = front　　B. front+l= rear

C. rear= = front　　D. (rear+l)%n= front

20. 在一个链队列中，假定 front 和 rear 分别为队首和队尾指针，则删除一个结点的操作为_____。

A. front=front->next　　B. rear=rear->next

C. rear=front->next　　D. front=rear->next

二、填空题

1. 线性表是一种典型的_____结构。

2. 在一个长度为 n 的顺序表的第 i 个元素之前插入一个元素，需要后移____个元素。

3. 顺序表中逻辑上相邻的元素的物理位置_____。

4. 要从一个顺序表删除一个元素时，被删除元素之后的所有元素均需_____一个位置，移动过程是从_____向_____依次移动每一个元素。

5. 在线性表的顺序存储中，元素之间的逻辑关系是通过_____决定的；在线性表的链接存储中，元素之间的逻辑关系是通过_____决定的。

6. 在双向链表中，每个结点含有两个指针域，一个指向_____结点，另一个指向_____结点。

7. 当对一个线性表经常进行存取操作，而很少进行插入和删除操作时，则采用_____存储结构为宜。相反，当经常进行的是插入和删除操作时，则采用_____存储结构为宜。

8. 顺序表中逻辑上相邻的元素，物理位置_____相邻，单链表中逻辑上相邻的元素，物理位置_____相邻。

9. 线性表、栈和队列都是_____结构，可以在线性表的_____位置插入和删除元素；对于栈只能在_____位置插入和删除元素；对于队列只能在_____位置插入元素和在_____位置删除元素。

10. 根据线性表的链式存储结构中每个结点所含指针的个数，链表可分为_____和_____；而根据指针的联接方式，链表又可分为_____和_____。

11. 在单链表中设置头结点的作用是_____。

12. 对于一个具有 n 个结点的单链表，在已知的结点 p 后插入一个新结点的时间复杂度为_____，在给定值为 x 的结点后插入一个新结点的时间复杂度为_____。

13. 对于一个栈作进栈运算时，应先判别栈是否为_____，作退栈运算时，应先判别栈是否为_____，当栈中元素为 m 时，作进栈运算时发生上溢，则说明栈的可用最大容量为_____。为了增加内存空间的利用率和减少发生上溢的可能性，由两个栈共享一片连续的内存空间时，应将两栈的_____分别设在这片内存空间的两端，这样只有当_____时才产生上溢。

14. 设有一空栈，现有输入序列 1，2，3，4，5，经过 push, push, pop, push, pop, push, push 后，输出序列是_____。

15. 无论对于顺序存储还是链式存储的栈和队列来说，进行插入或删除运算的时间复杂度均相同为_____。

三、简答题

1. 描述以下三个概念的区别：头指针，头结点，表头结点。

2. 线性表的两种存储结构各有哪些优缺点？

3. 对于线性表的两种存储结构，如果有 n 个线性表同时并存，而且在处理过程中各表的长度会动态发生变化，线性表的总数也会自动改变，在此情况下，应选用哪一种存储结构？为什么？

4. 对于线性表的两种存储结构，若线性表的总数基本稳定，且很少进行插入和删除操作，但要求以最快的速度存取线性表中的元素，应选用何种存储结构？试说明理由。

5. 在单循环链表中设置尾指针比设置头指针好吗？为什么？

6. 假定有四个元素 A，B，C，D 依次进栈，进栈过程中允许出栈，试写出所有可能的出栈序列。

7. 什么是队列的上溢现象？一般有几种解决方法，试简述之。

8. 下述算法的功能是什么？

```
LinkList    *Demo(LinkList *L)
{   // L 是无头结点的单链表
    LinkList *q,*p;
    if(L&&L->next)
    {   q=L; L=L->next; p=L;
        while (p->next)
            p=p->next;
        p->next=q; q->next=NULL;
     }
     return (L);
}
```

四、算法设计题

1. 设计在无头结点的单链表中删除第 i 个结点的算法。

2. 在单链表上实现线性表的求表长 ListLength(L)运算。

3. 设计将带表头的链表逆置算法。

4. 假设有一个带表头结点的链表，表头指针为 head，每个结点含三个域：data，next 和 prior。其中 data 为整型数域，next 和 prior 均为指针域。现在所有结点已经由 next 域连接起来，试编一个算法，利用 prior 域（此域初值为 NULL）把所有结点按照其值从小到大的顺序链接起来。

5. 已知线性表的元素按递增顺序排列，并以带头结点的单链表作存储结构。试编写一个删除表中所有值大于 min 且小于 max 的元素（若表中存在这样的元素）的算法。

6. 已知线性表的元素是无序的，且以带头结点的单链表作为存储结构。设计一个删除表中所有值小于 max 但大于 min 的元素的算法。

7. 假定用一个单循环链表来表示队列（也称为循环队列），该队列只设一个队尾指针，

不设队首指针，试编写下列各种运算的算法：

（1）向循环链队列插入一个元素值为 x 的结点；

（2）从循环链队列中删除一个结点。

8. 设顺序表 L 是一个递减有序表，试写一算法，将 x 插入其后仍保持 L 的有序性。

第二章参考答案

一、单项选择题

1. A　2. A　3. D　4. C　5. D　6. A　7. B　8. B　9. C　10. A　11. D　12. B
13. C　14. B　15. C　16. C　17. B　18. D　19. C　20. A

二、填空题

1．线性
2．n-i+1
3．相邻
4．前移，前，后
5．物理存储位置，链域的指针值
6．前趋，后继
7．顺序，链接
8．一定，不一定
9．线性，任何，栈顶，队尾，队头
10．单链表，双链表，非循环链表，循环链表
11．使空表和非空表统一；算法处理一致
12．O(1)，O(n)
13．栈满，栈空，m，栈底，两个栈的栈顶在栈空间的某一位置相遇
14．2、3
15．O(1)

三、简答题

1．头指针是指向链表中第一个结点（即表头结点）的指针；在表头结点之前附设的结点称为头结点；表头结点为链表中存储线性表中第一个数据元素的结点。若链表中附设头结点，则不管线性表是否为空表，头指针均不为空，否则表示空表的链表的头指针为空。

2. 线性表具有两种存储结构即顺序存储结构和链接存储结构。线性表的顺序存储结构可以直接存取数据元素，方便灵活、效率高，但插入、删除操作时将会引起元素的大量移动，因而降低效率：而在链接存储结构中内存采用动态分配，利用率高，但需增设指示结点之间关系的指针域，存取数据元素不如顺序存储方便，但结点的插入、删除操作较简单。

3. 应选用链接存储结构，因为链式存储结构是用一组任意的存储单元依次存储线性表中的各元素，这里存储单元可以是连续的，也可以是不连续的：这种存储结构对于元素的删除或插入运算是不需要移动元素的，只需修改指针即可，所以很容易实现表的容量的扩充。

4．应选用顺序存储结构，因为每个数据元素的存储位置和线性表的起始位置相差一个和数据元素在线性表中的序号成正比的常数。因此，只要确定了其起始位置，线性表中的任一个数据元素都可随机存取，因此，线性表的顺序存储结构是一种随机存取的存储结构，而链表则是一种顺序存取的存储结构。

5．设尾指针比设头指针好。尾指针是指向终端结点的指针，用它来表示单循环链表可以使得查找链表的开始结点和终端结点都很方便，设一带头结点的单循环链表，其尾指针为 rear，则开始结点和终端结点的位置分别是 rear->next->next 和 rear，查找时间都是 O(1)。若用头指针来表示该链表，则查找终端结点的时间为 O(n)。

6．共有 14 种可能的出栈序列，即为：

ABCD, ABDC，ACBD, ACDB，BACD，ADCB，BADC，BCAD, BCDA，BDCA,CBAD, CBDA，CDBA, DCBA

7．在队列的顺序存储结构中，设队头指针为 front，队尾指针为 rear，队列的容量（即存储的空间大小）为 maxnum。当有元素要加入队列（即入队）时，若 rear=maxnum，则会发生队列的上溢现象，此时就不能将该元素加入队列。对于队列，还有一种“假溢出”现象，队列中尚余有足够的空间，但元素却不能入队，一般是由于队列的存储结构或操作方式的选择不当所致，可以用循环队列解决。

一般地，要解决队列的上溢现象可有以下几种方法：

（1）可建立一个足够大的存储空间以避免溢出，但这样做往往会造成空间使用率低，浪费存储空间。

（2）要避免出现“假溢出”现象可用以下方法解决：

第一种：采用移动元素的方法。每当有一个新元素入队，就将队列中已有的元素向队头移动一个位置，假定空余空间足够。

第二种：每当删去一个队头元素，则可依次移动队列中的元素总是使 front 指针指向队列中的第一个位置。

第三种：采用循环队列方式。将队头、队尾看作是一个首尾相接的循环队列，即用循环数组实现，此时队首仍在队尾之前，作插入和删除运算时仍遵循“先进先出”的原则。

8．该算法的功能是：将开始结点摘下链接到终端结点之后成为新的终端结点，而原来的第二个结点成为新的开始结点，返回新链表的头指针。

四、算法设计题

1．算法思想为：

（1）应判断删除位置的合法性，当 i<0 或 i>n-1 时，不允许进行删除操作；

（2）当 i=0 时，删除第一个结点：

（3）当 0 <i<n 时，允许进行删除操作，但在查找被删除结点时，须用指针记住该结点的前趋结点。算法描述如下：

```
delete(LinkList *q,int i)
{   //在无头结点的单链表中删除第 i 个结点
    LinkList   *p,*s;
    int   j;
```

```
if(i<0)
    printf("Can't delete");
else if(i= =0)
    {   s=q;
        q=q->next;
        free(s);
     }
 else
    {   j=0; s=q;
         while((j<i) && (s! = NULL))
            {   p=s;
                s=s->next;
                j++;
             }
         if (s= =NULL)
             printf("Cant't    delete");
         else
             {     p->next=s->next;
                   free(s);
              }
        }
}
```

2．由于在单链表中只给出一个头指针，所以只能用遍历的方法来数单链表中的结点个数了。算法描述如下：

```
int ListLength ( LinkList *L )
{   //求带头结点的单链表的表长
    int len=0;
    ListList *p;
    p=L;
    while ( p->next!=NULL )
    {   p=p->next;
        len++;
    }
    return (len);
 }
```

3．设单循环链表的头指针为 head，类型为 LinkList。逆置时需将每一个结点的指针域作以修改，使其原前趋结点成为后继。如要更改 q 结点的指针域时，设 s 指向其原前趋结点，p 指向其原后继结点，则只需进行 q->next=s;操作即可，算法描述如下：

```
void invert(LinkList   *head)
{   //逆置 head 指针所指向的单循环链表
    linklist *p, *q, *s;
    q=head;
    p=head->next;
    while (p!=head)   //当表不为空时，逐个结点逆置
    {   s=q;
        q=p;
        p=p->next;
        q->next=s;
     }
     p->next=q;
  }
```

4．定义类型 LinkList 如下：

```
typedef struct node
{   int data;
    struct node *next,*prior;
}LinkList;
```

此题可采用插入排序的方法，设 p 指向待插入的结点，用 q 搜索已由 prior 域链接的有序表找到合适位置将 p 结点链入。算法描述如下：

```
insert (LinkList *head)
{    LinkList *p,*s,*q;
     p=head->next;   //p 指向待插入的结点，初始时指向第一个结点
     while(p!=NULL)
     {   s=head;   // s 指向 q 结点的前趋结点
         q=head->prior;   //q 指向由 prior 域构成的链表中待比较的结点
         while((q!=NULL) && (p->data>q->data))   //查找插入结点 p 的合适的插入位置
             {   s=q;
                 q=q->prior;
             }
         s->prior=p;
         p->prior=q;   //结点 p 插入到结点 s 和结点 q 之间
         p=p->next;
      }
  }
```

5．算法描述如下：

```
delete(LinkList *head, int max, int min)
{   linklist *p, *q;
```

```
    if (head!=NULL)
    {   q=head;
        p=head->next;
        while((p!=NULL) && (p->data<=min))
        {   q=p;
            p=p->next;
        }
        while((p!=NULL) && (p->data<max))
            p=p->next;
        q->next=p;
     }
}
```

6．算法描述如下：

```
delete(LinkList *head, int max, int min)
{   LinkList *p,*q;
    q=head;
    p=head->next;
    while (p!=NULL)
        if((p->data<=min) || (p->data>=max))
        {   q=p;
            p=p->next;
        }
        else
        {   q->next=p->next;
            free(p);
            p=q->next;
        }
 }
```

7．本题是对一个循环链队列做插入和删除运算，假设不需要保留被删结点的值和不需要回收结点，算法描述如下：

（1）插入（即入队）算法：

```
insert(LinkList *rear, elemtype x)
{   //设循环链队列的队尾指针为 rear,x 为待插入的元素
    LinkList *p;
    p=(LinkList *)malloc(sizeof(LinkList));
    if(rear= =NULL)   //如为空队，建立循环链队列的第一个结点
    {   rear=p;
        rear->next=p;    //链接成循环链表
```

```
    }
    else   //否则在队尾插入 p 结点
    {   p->next=rear->next;
        rear->next=p;
        rear=p;
    }
}
```

（2）删除（即出队）算法：

```
delete(LinkList   *rear)
{   //设循环链队列的队尾指针为 rear
    if (rear= =NULL)   //空队
        printf("underflow\n");
    if(rear->next= =rear)   //队中只有一个结点
        rear=NULL;
    else
        rear->next=rear->next->next;   //rear->next 指向的结点为循环链队列的队头结点
}
```

8．只要从终端结点开始往前找到第一个比 x 大(或相等)的结点数据，在这个位置插入就可以了。算法描述如下：

```
int InsertDecreaseList( SqList *L, elemtype x )
{   int i;
    if ( (*L).len>= maxlen)
    {   printf(“overflow");
        return(0);
    }
    for ( i=(*L).len ; i>0 && (*L).elem[ i-1 ] < x ; i--)
        (*L).elem[ i ]=(*L).elem[ i-1 ] ;     //  比较并移动元素
    (*L).elem[ i ] =x;
    (*L).len++;
    return(1);
}
```

第三章 串

一、单项选择题

1. 空串与空格字符组成的串的区别在于（　）。

A. 没有区别　　B. 两串的长度不相等

C. 两串的长度相等　　D. 两串包含的字符不相同

2. 一个子串在包含它的主串中的位置是指（　）。

A. 子串的最后那个字符在主串中的位置

B. 子串的最后那个字符在主串中首次出现的位置

C. 子串的第一个字符在主串中的位置

D. 子串的第一个字符在主串中首次出现的位置

3. 下面的说法中，只有（　）是正确的。

A. 字符串的长度是指串中包含的字母的个数

B. 字符串的长度是指串中包含的不同字符的个数

C. 若 T 包含在 S 中，则 T 一定是 S 的一个子串

D. 一个字符串不能说是其自身的一个子串

4. 两个字符串相等的条件是（　）。

A. 两串的长度相等

B. 两串包含的字符相同

C. 两串的长度相等，并且两串包含的字符相同

D. 两串的长度相等，并且对应位置上的字符相同

5. 若 SUBSTR（S，i，k）表示求 S 中从第 i 个字符开始的连续 k 个字符组成的子串的操作，则对于 S=“Beijing＆Nanjing”，SUBSTR（S，4，5）=（　）。

A. “ijing”　　B. “jing＆”

C. “ingNa”　　D. “ing＆N”

6. 若 INDEX（S，T）表示求 T 在 S 中的位置的操作，则对于 S=“Beijing＆Nanjing”，T=“jing”，INDEX（S，T）=（　）。

A. 2　　B. 3　　C. 4　　D. 5

7. 若 REPLACE（S，S1，S2）表示用字符串 S2 替换字符串 S 中的子串 S1 的操作，则对于 S=“Beijing＆Nanjing”，S1=“Beijing”，S2=“Shanghai”，REPLACE（S，S1，S2）=（　）。

A. “Nanjing＆Shanghai”　　B. “Nanjing＆Nanjing”

C. “ShanghaiNanjing”　　D. “Shanghai＆Nanjing”

8. 在长度为 n 的字符串 S 的第 i 个位置插入另外一个字符串，i 的合法值应该是（　）。

A. i＞0　　B. i≤n

C. $1 \leqslant i \leqslant n$　　D. $1 \leqslant i \leqslant n+1$

9. 字符串采用结点大小为 1 的链表作为其存储结构，是指（　）。

A. 链表的长度为 1

B. 链表中只存放 1 个字符

C. 链表的每个链结点的数据域中不仅只存放了一个字符

D. 链表的每个链结点的数据域中只存放了一个字符

二、填空题

1. 计算机软件系统中，有两种处理字符串长度的方法：一种是_____，第二种是_____。

2. 两个字符串相等的充要条件是_____和_____。

3. 设字符串 S1= “ABCDEF”，S2= “PQRS”，则运算 S=CONCAT（SUB（S1，2，LEN（S2））），SUB（S1，LEN（S2），2））后的串值为_____。

4. 串是指_____。

5. 空串是指_____，空格串是指_____。

三、算法设计题

1. 设有一个长度为 s 的字符串，其字符顺序存放在一个一维数组的第 1 至第 s 个单元中（每个单元存放一个字符）。现要求从此串的第 m 个字符以后删除长度为 t 的子串，$m<s$，$t<(s-m)$，并将删除后的结果复制在该数组的第 s 单元以后的单元中，试设计此删除算法。

2. 设 s 和 t 是表示成单链表的两个串，试编写一个找出 s 中第 1 个不在 t 中出现的字符（假定每个结点只存放 1 个字符）的算法。

第三章参考答案

一、单项选择题

1．B　2．C　3．C　4．D　5．B　6．C　7．D　8．C　9．D

二、填空题

1. 固定长度，设置长度指针
2. 两个串的长度相等，对应位置的字符相等
3. “BCDEDE”
4. 含 n 个字符的有限序列 （n≥0）
5. 不含任何字符的串，仅含空格字符的字符串

三、算法设计题

1．算法描述为：

```
int delete(r,s,t,m)   //从串的第 m 个字符以后删除长度为 t 的子串
char r[ ];
int s,t,m;
{   int i,j;
    for(i=1;i<=m;i++)
        r[s+i]=r[i];
    for(j=m+t-i;j<=s;j++)
        r[s-t+j]=r[j];
    return (1);
}   //delete
```

2．算法思想为：

（1）链表 s 中取出一个字符；将该字符与单链表 t 中的字符依次比较；

（2）当 t 中有与从 s 中取出的这个字符相等的字符，则从 t 中取下一个字符重复以上比较；

（3）当 t 中没有与从 s 中取出的这个字符相等的字符，则算法结束。

设单链表类型为 LinkList；注意，此时类型 LinkList 中的 data 成分为字符类型。

```
LinkString find(s,t)
LinkString   *s, *t;
{   LinkString *ps, *pt;
    ps=s;
    while(ps!=NULL)
    {   pt=t;
        while((pt!=NULL)&&(ps->data!=pt->data))
            pt=pt->next;
```

```
            if(pt= =NULL)
                ps=NULL;
            else
            {   ps=ps->next;
                s=ps;
            }
        }
        return s;
}   //find
```

第四章 数组与广义表

一、单项选择题

1. 设二维数组A[0…m-1][0…n-1]按行优先顺序存储在内存中，第一个元素的地址为p，每个元素占k个字节，则元素a_{ij}的地址为（ ）。

A. p +(i*n+j)*k　　B. p+[(i-1)*n+j-1]*k

C. p+[(j-1)*n+i-1]*k　　D. p+[j*n+i-1]*k

2. 已知二维数组$A_{10\times10}$中，元素a_{20}的地址为560，每个元素占4个字节，则元素a_{10}的地址为（ ）。

A. 520　　B. 522　　C. 524　　D. 518

3. 若数组A[0…m][0…n]按列优先顺序存储，则a_{ij}地址为（ ）。

A. LOC(a_{00})+[j*(m+1)+i]　　B. LOC(a_{00})+[j*n+i]

C. LOC(a_{00})+[(j-1)*n+i-1]　　D. LOC(a_{00})+[(j-1)*m+i-1]

4. 若下三角矩阵$A_{n\times n}$，按列顺序压缩存储在数组Sa[0…(n+1)n/2]中，则非零元素a_{ij}的地址为（ ）。（设每个元素占d个字节）

A. $[(j-1)*n-\frac{(j-2)(j-1)}{2}+i-1]*d$

B. $[(j-1)*n-\frac{(j-2)(j-1)}{2}+i-j]*d$

C. $[(j-1)*n-\frac{(j-2)(j-1)}{2}+i+1]*d$

D. $[(j-1)*n-\frac{(j-2)(j-1)}{2}+i-2]*d$

5. 设有广义表D=(a,b,D)，其长度为（ ），深度为（ ）。

A. 无穷大　　B. 3　　C. 2　　D. 5

6. 广义表A=（a），则表尾为（ ）。

A. a　　B. (( ))　　C. 空表　　D. (a)

7. 广义表A=((x,(a,B)),(x,(a,B),y))，则运算head(head(tail(A)))的结果为（ ）。

A. x　　B. (a,B)　　C. (x,(a,B))　　D. A

8. 下列广义表用图来表示时，分支结点最多的是（ ）。

A. L=((x,(a,B)),(x,(a,B),y))　　B. A=(s,(a,B))

C. B=((x,(a,B),y))　　D. D=((a,B),(c,(a,B),D)

9. 通常对数组进行的两种基本操作是（ ）。

A. 建立与删除　　B. 索引和修改

C. 查找和修改　　D. 查找与索引

10. 假定在数组 A 中，每个元素的长度为 3 个字节，行下标 i 从 1 到 8，列下标 j 从 1 到 10，从首地址 SA 开始连续存放在存储器内，存放该数组至少需要的单元数为（　）。

A. 80　　B. 100　　C. 240　　D. 270

11. 数组 A 中，每个元素的长度为 3 个字节，行下标 i 从 1 到 8，列下标 j 从 1 到 10，从首地址 SA 开始连续存放在存储器内，该数组按行存放时，元素 A[8][5]的起始地址为（　）。

A. SA+141　　B. SA+144　　C. SA+222　　D. SA+225

12. 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

13. 若采用三元组压缩技术存储稀疏矩阵，只要把每个元素的行下标和列下标互换，就完成了对该矩阵的转置运算，这种观点（　）。

A. 正确　　B. 不正确

14. 一个广义表的表头总是一个（　）。

A. 广义表　　B. 元素　　C. 空表　　D. 元素或广义表

15. 一个广义表的表尾总是一个（　）。

A. 广义表　　B. 元素　　C. 空表　　D. 元素或广义表

16. 数组就是矩阵，矩阵就是数组，这种说法（　）。

A. 正确　　B. 错误

C. 前句对，后句错　　D. 后句对

二、填空题

1. 一维数组的逻辑结构是_____，存储结构是_____；对于二维或多维数组，分为_____和_____两种不同的存储方式。

2. 对于一个二维数组 A[m][n]，若按行序为主序存储，则任一元素 A[i][j]相对于 A[0][0]的地址为_____。

3. 一个广义表为(a,(a,b),d,e,((i,j),k))，则该广义表的长度为_____，深度为_____。

4. 一个稀疏矩阵为 $\begin{bmatrix} 0 & 0 & 2 & 0 \\ 3 & 0 & 0 & 0 \\ 0 & 0 & -1 & 5 \\ 0 & 0 & 0 & 0 \end{bmatrix}$，则对应的三元组线性表为_____。

5. 一个 n×n 的对称矩阵，如果以行为主序或以列为主序存入内存，则其容量为_____。

6. 已知广义表 A=((a,b,c),(d,e,f))，则运算 head(tail(head(tail(A))))=_____。

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序存储，a_{00} 为第一个元素，其存储地址为 0，每个元素占有 1 个存储地址空间，则 a_{85} 的地址为_____。

8. 已知广义表 Ls=(a,(b,c,d),e)，运用 head 和 tail 函数取出 Ls 中的原子 b 的运算是_____。

9. 三维数组 R[c_1…d_1, c_2…d_2, c_3…d_3]共含有_____个元素。（其中：$c_1 \leqslant d_1$, $c_2 \leqslant d_2$, $c_3 \leqslant d_3$）

10. 数组 A[1…10，-2…6，2…8]以行优先的顺序存储，设第一个元素的首地址是 100，每个元素占 3 个存储长度的存储空间，则元素 A[5，0，7]的存储地址为_____。

三、判断题

1. 数组可看作基本线性表的一种推广，因此与线性表一样，可以对它进行插入、删除等操作。（ ）

2. 多维数组可以看作数据元素也是基本线性表的基本线性表。（ ）

3. 以行为主序或以列为主序对于多维数组的存储没有影响。（ ）

4. 对于不同的特殊矩阵应该采用不同的存储方式。（ ）

5. 采用压缩存储之后，下三角矩阵的存储空间可以节约一半。（ ）

6. 在一般情况下，采用压缩存储之后，对称矩阵是所有特殊矩阵中存储空间节约最多的。（ ）

7. 矩阵不仅是表示多维数组，而且是表示图的重要工具。（ ）

8. 距阵中的数据元素可以是不同的数据类型。（ ）

9. 矩阵中的行列数往往是不相等的。（ ）

10. 广义表的表头可以是广义表，也可以是单个元素。（ ）

11. 广义表的表尾一定是一个广义表。（ ）

12. 广义表的元素可以是子表，也可以是单元素。（ ）

13. 广义表不能递归定义。（ ）

14. 广义表实际上是基本线性表的推广。（ ）

15. 广义表的组成元素可以是不同形式的元素。（ ）

第四章参考答案

一、单项选择题

1.A 2.A 3.A 4. B 5. BA 6. C 7.A 8.A 9. C 10. C 11.C 12. C 13. B 14.D 15.A 16.B

二、填空题

1. 线性结构，顺序结构，以行为主序，以列为主序
2. i×n+j 个元素位置
3. 5，3
4. ((0，2，2)，(1，0，3)，(2，2，-1)，(2，3，5))
5. n×(n+1)/2
6. e
7. 41
8. head(head(tail(Ls)))
9. $(d_1-c_1+1)\times(d_2-c_2+1)\times(d_3-c_3+1)$
10. 913

三、判断题

1.× 2.√ 3.√ 4.√ 5.× 6.× 7.√ 8.× 9.× 10.√ 11.√ 12.√ 13.× 14.√ 15.√

第五章 树与二叉树

一、单项选择题

1. 在一棵度为 3 的树中，度为 3 的结点数为 2 个，度为 2 的结点数为 1 个，度为 1 的结点数为 2 个，则度为 0 的结点数为（　）个。

A. 4　　B. 5　　C. 6　　D. 7

2. 假设在一棵二叉树中，双分支结点数为 15，单分支结点数为 30 个，则叶子结点数为（　）个。

A. 15　　B. 16　　C. 17　　D. 47

3. 假定一棵三叉树的结点数为 50，则它的最小高度为（　）。

A. 3　　B. 4　　C. 5　　D. 6

4. 在一棵二叉树上第 4 层的结点数最多为（　）。

A. 2　　B. 4　　C. 6　　D. 8

5. 用顺序存储的方法将完全二叉树中的所有结点逐层存放在数组中 R[1..n]，结点 R[i] 若有左孩子，其左孩子的编号为结点（　）。

A. R[2i+1]　　B. R[2i]　　C. R[i/2]　　D. R[2i-1]

6. 由权值分别为 3,8,6,2,5 的叶子结点生成一棵哈夫曼树，它的带权路径长度为（　）。

A. 24　　B. 48　　C. 72　　D. 53

7. 线索二叉树是一种（　）结构。

A. 逻辑　　B. 逻辑和存储　　C. 物理　　D. 线性

8. 线索二叉树中，结点 p 没有左子树的充要条件是（　）。

A. p->lc=NULL　　B. p->ltag=1

C. p->ltag=1 且 p->lc=NULL　　D. 以上都不对

9. 设 n , m 为一棵二叉树上的两个结点，在中序遍历序列中 n 在 m 前的条件是（　）。

A. n 在 m 右方　　B. n 在 m 左方

C. n 是 m 的祖先　　D. n 是 m 的子孙

10. 如果 F 是由有序树 T 转换而来的二叉树，那么 T 中结点的前序就是 F 中结点的（　）。

A. 中序　　B. 前序　　C. 后序　　D. 层次序

11. 欲实现任意二叉树的后序遍历的非递归算法而不必使用栈，最佳方案是二叉树采用（　）存储结构。

A. 三叉链表　　B. 广义表　　C. 二叉链表　　D. 顺序

12. 下面叙述正确的是（　）。

A. 二叉树是特殊的树

B. 二叉树等价于度为 2 的树

C. 完全二叉树必为满二叉树

D. 二叉树的左右子树有次序之分

13. 任何一棵二叉树的叶子结点在先序、中序和后序遍历序列中的相对次序（　　）。

A. 不发生改变　　B. 发生改变

C. 不能确定　　D. 以上都不对

14. 已知一棵完全二叉树的结点总数为 9 个，则最后一层的结点数为（　　）。

A. 1　　B. 2　　C. 3　　D. 4

15. 根据先序序列 ABDC 和中序序列 DBAC 确定对应的二叉树，该二叉树（　　）。

A. 是完全二叉树　　B. 不是完全二叉树

C. 是满二叉树　　D. 不是满二叉树

二、判断题

1. 二叉树中每个结点的度不能超过 2，所以二叉树是一种特殊的树。（　）

2. 二叉树的前序遍历中，任意结点均处在其子女结点之前。（　）

3. 线索二叉树是一种逻辑结构。（　）

4. 哈夫曼树的总结点个数（多于 1 时）不能为偶数。（　）

5. 由二叉树的先序序列和后序序列可以唯一确定一颗二叉树。（　）

6. 树的后序遍历与其对应的二叉树的后序遍历序列相同。（　）

7. 根据任意一种遍历序列即可唯一确定对应的二叉树。（　）

8. 满二叉树也是完全二叉树。（　）

9. 哈夫曼树一定是完全二叉树。（　）

10. 树的子树是无序的。（　）

三、填空题

1. 假定一棵树的广义表表示为 A（B（E），C（F（H，I，J），G），D），则该树的度为_____，树的深度为_____，终端结点的个数为_____，单分支结点的个数为_____，双分支结点的个数为_____，三分支结点的个数为_____，C 结点的双亲结点为_____，其孩子结点为_____和_____结点。

2. 设 F 是一个森林，B 是由 F 转换得到的二叉树，F 中有 n 个非终端结点，则 B 中右指针域为空的结点有_____个。

3. 对于一个有 n 个结点的二叉树，当它为一棵_____二叉树时具有最小高度，即为_____，当它为一棵单支树具有_____高度，即为_____。

4. 由带权为 3，9，6，2，5 的 5 个叶子结点构成一棵哈夫曼树，则带权路径长度为___。

5. 在一棵二叉排序树上按_____遍历得到的结点序列是一个有序序列。

6. 对于一棵具有 n 个结点的二叉树，当进行链接存储时，其二叉链表中的指针域的总数为_____个，其中_____个用于链接孩子结点，_____个空闲着。

7. 在一棵二叉树中，度为 0 的结点个数为 n_0，度为 2 的结点个数为 n_2，则 n_0=_____。

8. 一棵深度为 k 的满二叉树的结点总数为_____，一棵深度为 k 的完全二叉树的结

点总数的最小值为_____，最大值为_____。

9. 由三个结点构成的二叉树，共有____种不同的形态。

10. 设高度为 h 的二叉树中只有度为 0 和度为 2 的结点，则此类二叉树中所包含的结点数至少为____。

11. 一棵含有 n 个结点的 k 叉树，_____形态达到最大深度，____形态达到最小深度。

12. 对于一棵具有 n 个结点的二叉树，若一个结点的编号为 i(1≤i≤n)，则它的左孩子结点的编号为_____，右孩子结点的编号为_____，双亲结点的编号为_____。

13. 对于一棵具有 n 个结点的二叉树，采用二叉链表存储时，链表中指针域的总数为_____个，其中_____个用于链接孩子结点，_____个空闲着。

14. 哈夫曼树是指_____的二叉树。

15. 空树是指_____，最小的树是指_____。

16. 二叉树的链式存储结构有_____和_____两种。

17. 三叉链表比二叉链表多一个指向_____的指针域。

18. 线索是指_____。

19. 线索链表中的 rtag 域值为_____时，表示该结点无右孩子，此时_____域为指向该结点后继线索的指针。

20. 本节中我们学习的树的存储结构有_____、_____和_____。

四、应用题

1. 已知一棵树边的集合为{<i，m>，<i，n>，<e，i>，<b，e>，<b，d>，<a，b>，<g，j>，<g，k>，<c，g>，<c，f>，<h，l>，<c，h>，<a，c>}，请画出这棵树，并回答下列问题：

（1）哪个是根结点？

（2）哪些是叶子结点？

（3）哪个是结点 g 的双亲？

（4）哪些是结点 g 的祖先？

（5）哪些是结点 g 的孩子？

（6）哪些是结点 e 的孩子？

（7）哪些是结点 e 的兄弟？哪些是结点 f 的兄弟？

（8）结点 b 和 n 的层次号分别是什么？

（9）树的深度是多少？

（10）以结点 c 为根的子树深度是多少？

2. 一棵度为 2 的树与一棵二叉树有何区别。

3. 试分别画出具有 3 个结点的树和二叉树的所有不同形态？

4. 已知用一维数组存放的一棵完全二叉树：ABCDEFGHIJKL，写出该二叉树的先序、中序和后序遍历序列。

5. 一棵深度为 H 的满 k 叉树有如下性质：第 H 层上的结点都是叶子结点，其余各层上每个结点都有 k 棵非空子树，如果按层次自上至下，从左到右顺序从 1 开始对全部结点编号，回答下列问题：

（1）各层的结点数目是多少？

（2）编号为 n 的结点的父结点如果存在，编号是多少？

（3）编号为 n 的结点的第 i 个孩子结点如果存在，编号是多少？

（4）编号为 n 的结点有右兄弟的条件是什么？其右兄弟的编号是多少？

6. 找出所有满足下列条件的二叉树：

（1）它们在先序遍历和中序遍历时，得到的遍历序列相同；

（2）它们在后序遍历和中序遍历时，得到的遍历序列相同；

（3）它们在先序遍历和后序遍历时，得到的遍历序列相同；

7. 假设一棵二叉树的先序序列为 EBADCFHGIKJ，中序序列为 ABCDEFGHIJK，请写出该二叉树的后序遍历序列。

8. 假设一棵二叉树的后序序列为 DCEGBFHKJIA，中序序列为 DCBGEAHFIJK，请写出该二叉树的先序遍历序列。

9. 给出如图 5-14 所示的森林的先根、后根遍历结点序列，然后画出该森林对应的二叉树。

10．给定一组权值（5，9，11，2，7，16），试设计相应的哈夫曼树。

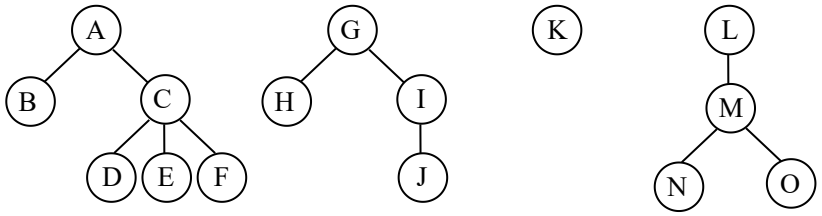


图 5-14

五、算法设计题

1. 一棵具有 n 个结点的完全二叉树以一维数组作为存储结构，试设计一个对该完全二叉树进行先序遍历的算法。

2. 给定一棵用二叉链表表示的二叉树，其中的指针 t 指向根结点，试写出从根开始，按层次遍历二叉树的算法，同层的结点按从左至右的次序访问。

3. 写出在中序线索二叉树中结点 P 的右子树中插入一个结点 s 的算法。

4. 给定一棵二叉树，用二叉链表表示，其根指针为 t，试写出求该二叉树中结点 n 的双亲结点的算法。若没有结点 n 或者该结点没有双亲结点，分别输出相应的信息；若结点 n 有双亲，输出其双亲的值。

第五章参考答案

一、单项选择题

1. C　2. B　3. C　4. D　5. B　6. D　7. C　8. B　9. B　10. B　11. A　12. D　13. A　14. B　15. AD

二、判断题

1.×　2.√　3.×　4.√　5.×　6.×　7.×　8.√　9.×　10.×

三、填空题

1. 3，4，6，1，1，2，A，F，G

2. n+1

3. 完全，$\lceil \log_2(n+1) \rceil$，最大，n

4. 55

5. 中序

6. 2n，n-1，n+1

7. n_2+1

8. 2^k-1，2^{k-1}，2^k-1

9. 5

10. 2h-1

11. 单支树，完全 k 叉树

12. 2i，2i+1，i/2（或⌊i/2⌋）

13. 2n，n-1，n+1

14. 带权路径长度最小

15. 结点数为 0，只有一个根结点的树

16. 二叉链表，三叉链表

17. 双亲结点

18. 指向结点前驱和后继信息的指针

19. 1，RChild

20. 孩子表示法，双亲表示法，长子兄弟表示法

四、应用题

1. 解答：

根据给定的边确定的树如图 5-15 所示。

其中根结点为 a；

叶子结点有：d、m、n、j、k、f、l；

c 是结点 g 的双亲；

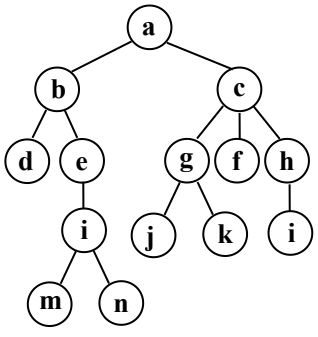


图 5-15

a、c 是结点 g 的祖先；
j、k 是结点 g 的孩子；
i 是结点 e 的孩子；
d 是结点 e 的兄弟，g、h 是结点 f 的兄弟；
结点 b 和 n 的层次号分别是 2 和 5；
树的深度为 5；
以 c 为根的子树深度为 3。

2. 解答：

度为 2 的树有两个分支，但分支没有左右之分；一棵二叉树也有两个分支，但有左右之分，左右子树不能交换。

3. 解答： 略

4. 解答：

先序序列：ABDHIEJKCFLG
中序序列：HDIBJEKALFCG
后序序列：HIDJKEBLFGCA

5. 解答：

（1）第 i 层上的结点数目是 m^{i-1}。
（2）编号为 n 的结点的父结点如果存在，编号是((n-2)/m)+1。
（3）编号为 n 的结点的第 i 个孩子结点如果存在，编号是(n-1)*m+i+1。
（4）编号为 n 的结点有右兄弟的条件是(n-1)%m≠0。其右兄弟的编号是 n+1。

6. 解答：

（1）先序序列和中序序列相同的二叉树为：空树或者任一结点均无左孩子的非空二叉树；
（2）中序序列和后序序列相同的二叉树为：空树或者任一结点均无右孩子的非空二叉树；
（3）先序序列和后序序列相同的二叉树为：空树或仅有一个结点的二叉树。

7. 解答：后序序列：ACDBGJKIHFE

8. 解答：先序序列：ABCDGEIHFJK

9. 解答：

先根遍历：ABCDEFGHIJKLMNO
后根遍历：BDEFCAHJIGKNOML
森林转换成二叉树如图 5-16 所示。

10. 解答：构造而成的哈夫曼树如图 5-17 所示。

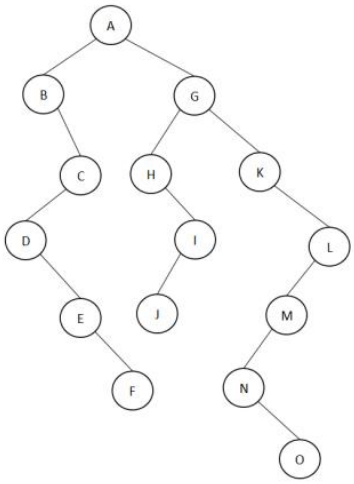


图 5-16

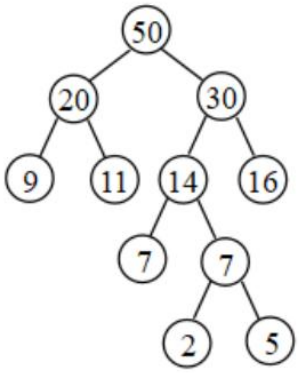


图 5-17

五、算法设计题

1. 解答：这个问题可以用递归算法，也可用非递归算法，下面给出的为非递归算法。假设该完全二叉树的结点以层次为序，按照从上到下，同层从左到右顺序编号，存放在一个一维数组 R[1..n]中，且用一个有足够大容量为 maxlen 的顺序栈作辅助存储，算法描述如下：

```
preorder (R)   //先序遍历二叉树 R
{   int R[n];
    int root;
    SqStack *s;   //s 为一个指针栈，类型为 seqstack，其中包含 top 域和数组 data
    s->top= -1;   //s 栈置空
    root=1;
    while ((root<=n) || (s->top>-1))
    {   while (root<=n)
        {   printf(R[root]);
            s->top++;
            s->data[s->top]=root;
            root=2*root;
        }
        if (s->top>-1)   //栈非空访问，遍历右子树
        {   root=s->data[s->top]*2+1;
            s->top--;
        }
    }
}
```

2. 解答：考虑用一个顺序队 que 来保存遍历过程中的各个结点，由于二叉树以二叉链表存储，所以可设 que 为一个指向数据类型为 bitree 的指针数组，最大容量为 maxnum，下标从 1 开始，同层结点从左到右存放。算法中的 front 为队头指针，rear 为队尾指针。

```
levelorder (BiTree *t)  //按层次遍历二叉树 t
{   BiTree *que[maxnum];
    int rear,front;
    if (t!=NULL)
    {   front=0;  //置空队列
        rear=1;
        que[1]=t;
        do
        {   front=front%maxsize+1;  //出队
            t=que[front];
            printf(t->data);
            if (t->lchild!=NULL)  //左子树的根结点入队
            {   rear=rear%maxsize+1;
                que[rear]=t->lchild;
            }
            if (t->rchild!=NULL)  //右子树的根结点入队
            {   rear=rear%maxsize+1;
                que[rear]=t->rchild;
            }
        }while (rear= =front);  //队列为空时结束
    }
}
```

3. 解答：设该线索二叉树类型为 bithptr，包含 5 个域：lchild，ltag，data，rchild，rtag。

```
insert(p, s)  //将 s 结点作为 p 的右子树插入
BiThrNode *p,*s;
{   BiThrNode *q;
    if (p->rtag= =1)  //无右子树，则有右线索
    {   s->rchild=p->rchild;
        s->rtag=1;
        p->rchild=s;
        p->rtag=0;
    }
    else
    {   q=p->rchild;
        while(q->ltag= =0)  //查找 p 所指结点中序后继，即为其右子树中最左下的结点
```

```
            q=q->lchild;
        q->lchild=p->rchild;
        s->rtag=0;
        p->rchild=s;
    }
    s->lchild=p;   //将 s 结点的左线索指向 p 结点
    s->ltag=1;
}
```

4. 解答：利用一个队列来完成，设该队列类型为指针类型，最大容量为 maxnum。算法中的 front 为队头指针，rear 为队尾指针，若当前队头结点的左、右子树的根结点不是所求结点，则将两子树的根结点入队，否则，队头指针所指结点即为结点的双亲。

```
parentjudge(t,n)
BiTree *t;
int n;
{   BiTree *que[maxnum];
    int front,rear;
    BiTree *parent;
    parent=NULL;
    if (t)
        if (t->data= =n)
            printf("no parent!");   //n 是根结点，无双亲
        else
        {   front=0;   //初始化队列
            rear=1;
            que[1]=t;   //根结点进队
            do
            {   front=front%maxsize+1;
                t=que[front];
                if((t->lchild->data= =n)|| (t->rchild->data= =n))   //结点 n 有双亲
                {   parent=t;
                    front=rear;
                    printf("parent",t->data);
                }
                else
                {   if (t->lchild!=NULL)   //左子树的根结点入队
                    {   rear=rear%maxsize+1;
                        que[rear]=t->lchild;
                    }
```

```
                if (t->rchild!=NULL)   //右子树的根结点入队
                {   rear=rear%maxsize+1;
                    que[rear]=t->rchild;
                }
            }
        }while(rear= =front);   //队空时结束
        if (parent = =NULL)
            printf("结点不存在");
    }
}
```

第六章 图

一、单项选择题

1. 在一个具有 n 个顶点的有向图中，若所有顶点的出度数之和为 s，则所有顶点的入度数之和为(　　)。

A. s　　B. s-1　　C. s+1　　D. n

2. 在一个具有 n 个顶点的有向图中，若所有顶点的出度数之和为 s，则所有顶点的度数之和为(　　)。

A. s　　B. s-1　　C. s+1　　D. 2s

3. 在一个具有 n 个顶点的无向图中，若具有 e 条边，则所有顶点的度数之和为(　　)。

A. n　　B. e　　C. n+e　　D. 2e

4. 在一个具有 n 个顶点的无向完全图中，所含的边数为(　　)。

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

5. 在一个具有 n 个顶点的有向完全图中，所含的边数为(　　)。

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

6. 在一个无向图中，若两顶点之间的路径长度为 k，则该路径上的顶点数为(　　)。

A. k　　B. k+1　　C. k+2　　D. 2k

7. 对于一个具有 n 个顶点的无向连通图，它包含的连通分量的个数为(　　)。

A. 0　　B. 1　　C. n　　D. n+1

8. 若一个图中包含有 k 个连通分量，若要按照深度优先搜索的方法访问所有顶点，则必须调用(　　)次深度优先搜索遍历的算法。

A. k　　B. 1　　C. k-1　　D. k+1

9. 若要把 n 个顶点连接为一个连通图，则至少需要(　　)条边。

A. n　　B. n+1　　C. n-1　　D. 2n

10. 在一个具有 n 个顶点和 e 条边的无向图的邻接矩阵中，表示边存在的元素（又称为有效元素）的个数为(　　)。

A. n　　B. n×e　　C. e　　D. 2×e

11. 在一个具有 n 个顶点和 e 条边的有向图的邻接矩阵中，表示边存在的元素个数为(　　)。

A. n　　B. n×e　　C. e　　D. 2×e

12. 在一个具有 n 个顶点和 e 条边的无向图的邻接表中，边结点的个数为(　　)。

A. n　　B. n×e　　C. e　　D. 2×e

13. 在一个具有 n 个顶点和 e 条边的有向图的邻接表中，保存顶点单链表的表头指针向量的大小至少为(　　)。

A. n　　B. 2n　　C. e　　D. 2e

14. 在一个无权图的邻接表表示中，每个边结点至少包含(　　)域。

A. 1　　B. 2　　C. 3　　D. 4

15. 对于一个有向图，若一个顶点的度为k1，出度为k2，则对应邻接表中该顶点单链表中的边结点数为(　　)。

A. k1　　B. k2　　C. k1-k2　　D. k1+k2

16. 对于一个有向图，若一个顶点的度为k1，出度为k2，则对应逆邻接表中该顶点单链表中的边结点数为(　　)。

A. k1　　B. k2　　C. k1-k2　　D. k1+k2

17. 对于一个无向图，下面(　　)种说法是正确的。

A. 每个顶点的入度等于出度　　B. 每个顶点的度等于其入度与出度之和

C. 每个顶点的入度为0　　D. 每个顶点的出度为0

18. 在一个有向图的邻接表中，每个顶点单链表中结点的个数等于该顶点的(　　)。

A. 出边数　　B. 入边数　　C. 度数　　D. 度数减1

19. 若一个图的边集为{(A,B),(A,C),(B,D),(C,F),(D,E),(D,F)}，则从顶点A开始对该图进行深度优先搜索，得到的顶点序列可能为(　　)。

A. A,B,C,F,D,E　　B. A,C,F,D,E,B

C. A,B,D,C,F,E　　D. A,B,D,F,E,C

20. 若一个图的边集为{(A,B),(A,C),(B,D),(C,F),(D,E),(D,F)}，则从顶点A开始对该图进行广度优先搜索，得到的顶点序列可能为(　　)。

A. A,B,C,D,E,F　　B. A,B,C,F,D,E

C. A,B,D,C,E,F　　D. A,C,B,F,D,E

21. 若一个图的边集为{<1,2>,<1,4>,<2,5>,<3,1>,<3,5>,<4,3>}，则从顶点1开始对该图进行深度优先搜索，得到的顶点序列可能为(　　)。

A. 1, 2, 5, 4, 3　　B. 1, 2, 3, 4, 5

C. 1, 2, 5, 3, 4　　D. 1, 4, 3, 2, 5

22. 若一个图的边集为{<1,2>,<1,4>,<2,5>,<3,1>,<3,5>,<4,3>}，则从顶点1开始对该图进行广度优先搜索，得到的顶点序列可能为(　　)。

A. 1, 2, 3, 4, 5　　B. 1, 2, 4, 3, 5

C. 1, 2, 4, 5, 3　　D. 1, 4, 2, 5, 3

23. 由一个具有n个顶点的连通图生成的最小生成树中，具有(　　)条边。

A. n　　B. n-1　　C. n+1　　D. 2×n

24. 已知一个有向图的边集为{<a,b>,<a,c>,<a,d>,<b,d>,<b,e>,<d,e>}，则由该图产生的一种可能的拓扑序列为(　　)。

A. a,b,c,d,e　　B. a,b,d,e,b　　C. a,c,b,e,d　　D. a,c,d,b,e

二、填空题

1. 在一个图中，所有顶点的度数之和等于所有边数的_____倍。

2. 在一个具有n个顶点的无向完全图中，包含有_____条边，在一个具有n个顶点的有向完全图中，包含有_____条边。

3. 假定一个有向图的顶点集为{a, b, c, d, e, f}，边集为{<a,c>, <a,e>, <c,f>, <d,c>, <e,b>, <e,d>}，则出度为 0 的顶点个数为_____，入度为 1 的顶点个数为_____。

4. 在一个具有 n 个顶点的无向图中，要连通所有顶点则至少需要_____条边。

5. 表示图的两种存储结构为_____和_____。

6. 在一个连通图中存在着_____个连通分量。

7. 图中的一条路径长度为 k，该路径所含的顶点数为_____。

8. 若一个图的顶点集为{a,b,c,d,e,f}，边集为{(a,b),(a,c),(b,c),(d,e)}，则该图含有_____个连通分量。

9. 对于一个具有 n 个顶点的图，若采用邻接矩阵表示，则矩阵大小至少为_____×_____。

10. 对于具有 n 个顶点和 e 条边的有向图和无向图，在它们对应的邻接表中，所含边结点的个数分别为_____和_____。

11. 在有向图的邻接表和逆邻接表表示中，每个顶点邻接表分别链接着该顶点的所有_____和_____结点。

12. 对于一个具有 n 个顶点和 e 条边的无向图，当分别采用邻接矩阵和邻接表表示时，求任一顶点度数的时间复杂度分别为_____和_____。

13. 假定一个图具有 n 个顶点和 e 条边，则采用邻接矩阵和邻接表表示时，其相应的空间复杂度分别为_____和_____。

14. 一个图的边集为{(a,c),(a,e),(b,e),(c,d),(d,e)}，从顶点 a 出发进行深度优先搜索遍历得到的顶点序列为_____，从顶点 a 出发进行广度优先搜索遍历得到的顶点序列为_____。

15. 一个图的边集为{<a,c>,<a,e>,<c,f>,<d,c>,<e,b>,<e,d>}，从顶点 a 出发进行深度优先搜索遍历得到的顶点序列为_____，从顶点 a 出发进行广度优先搜索遍历得到的顶点序列为_____。

16. 图的_____优先搜索遍历算法是一种递归算法，图的_____优先搜索遍历算法需要使用队列。

17. 对于一个具有 n 个顶点和 e 条边的连通图，其生成树中的顶点数和边数分别为_____和_____。

18. 若一个连通图中每个边上的权值均不同，则得到的最小生成树是_____（唯一/不唯一）的。

19. 根据图的存储结构进行某种次序的遍历，得到的顶点序列是__（唯一/不唯一）的。

20. 假定一个有向图的边集为{<a,c>,<a,e>,<c,f>,<d,c>,<e,b>,<e,d>}，对该图进行拓扑排序得到的顶点序列为_____。

三、应用题

1. 对于一个无向图 6-11(a)，假定采用邻接矩阵表示，试分别写出从顶点 0 出发按深度优先搜索遍历得到的顶点序列和按广度优先搜索遍历得到的顶点序列。

注：每一种序列都是唯一的，因为都是在存储结构上得到的。

2. 对于一个有向图 6-11(b)，假定采用邻接表表示，并且假定每个顶点单链表中的边

结点是按出边邻接点序号从大到小的次序链接的，试分别写出从顶点 0 出发按深度优先搜索遍历得到的顶点序列和按广度优先搜索遍历得到的顶点序列。

注：每一种序列都是唯一的，因为都是在存储结构上得到的。

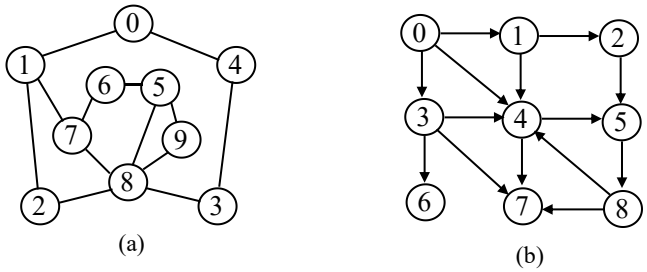


图 6-11

3. 已知一个无向图的邻接矩阵如图 6-12(a)所示，试写出从顶点 0 出发分别进行深度优先和广度优先搜索遍历得到的顶点序列。

4. 已知一个无向图的邻接表如图 6-12(b)所示，试写出从顶点 0 出发分别进行深度优先和广度优先搜索遍历得到的顶点序列。

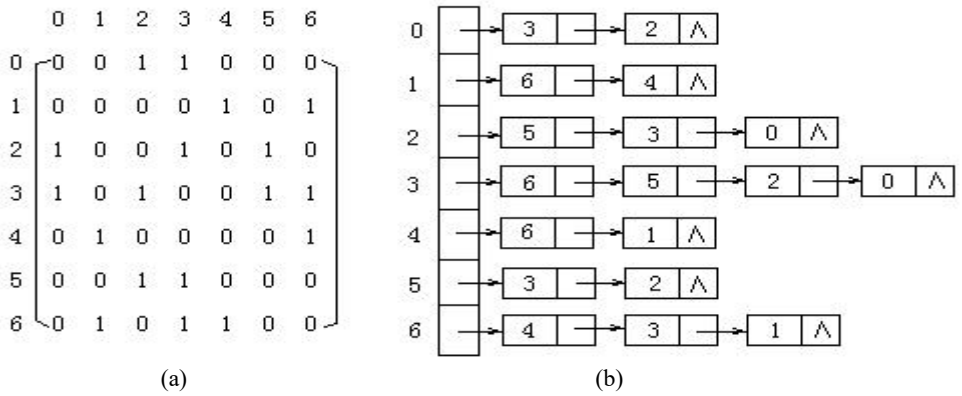


图 6-12

5. 已知图 6-13 所示的一个网，按照 Prim 方法，从顶点 1 出发，求该网的最小生成树的产生过程。

6. 已知图 6-13 所示的一个网，按照 Kruskal 方法，求该网的最小生成树的产生过程。

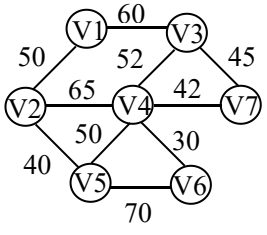
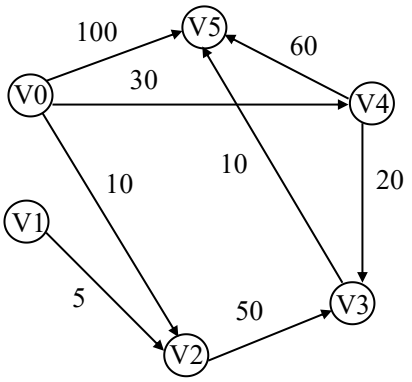


图 6-13

7. 图 6-14 所示为一个有向网图及其带权邻接矩阵，要求对有向图采用 Dijkstra 算法，求从 V0 到其余各顶点的最短路径。



(a) 有向带权图

$$\begin{pmatrix} \infty & \infty & 10 & \infty & 30 & 100 \\ \infty & \infty & 5 & \infty & \infty & \infty \\ \infty & \infty & \infty & 50 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & 10 \\ \infty & \infty & \infty & 20 & \infty & 60 \\ \infty & \infty & \infty & \infty & \infty & \infty \end{pmatrix}$$

(b) 带权邻接矩阵

图 6-14 有向带权图及其邻接矩阵

8. 图 6-15 给出了一个具有 15 个活动、11 个事件的工程的 AOE 网，求关键路径。

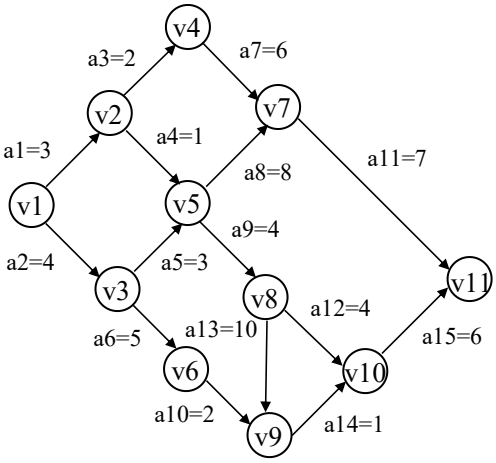


图 6-15

四、算法设计题

1. 编写一个算法，求出邻接矩阵表示的无向图中序号为 numb 的顶点的度数。

int degree1(Graph & ga, int numb)

2. 编写一个算法，求出邻接矩阵表示的有向图中序号为 numb 的顶点的度数。

int degree2(Graph & ga, int numb)

3. 编写一个算法，求出邻接表表示的无向图中序号为 numb 的顶点的度数。

int degree3(GraphL & gl, int numb)

4. 编写一个算法，求出邻接表表示的有向图中序号为 numb 的顶点的度数。

int degree4(GraphL & gl, int numb)

第六章参考答案

一、单项选择题

1. A　2. D　3. D　4. C　5. B　6. B　7. B　8. A　9. C　10. D　11. C　12. D　13. A　14. B
15. B　16. C　17. A　18. A　19. B　20. D　21. A　22. C　23. B　24. A

二、填空题

1. 2
2. n(n-1)/2，n(n-1)
3. 2，4
4. n-1
5. 邻接矩阵，邻接表
6. 1
7. k+1
8. 3
9. n，n
10. e，2e
11. 出边，入边
12. O（n），O（e/n）
13. O（n^2），O（n+e）
14. acdeb，acedb （答案不唯一）
15. acfebd，acefbd （答案不唯一）
16. 深度，广度
17. n，n-1
18. 唯一
19. 唯一
20. aebdcf（答案不唯一）

三、应用题

1. 深度优先搜索序列：0, 1, 2, 8, 3, 4, 5, 6, 7, 9
   广度优先搜索序列：0, 1, 4, 2, 7, 3, 8, 6, 5, 9
2. 深度优先搜索序列：0, 4, 7, 5, 8, 3, 6, 1, 2
   广度优先搜索序列：0, 4, 3, 1, 7, 5, 6, 2, 8
3. 深度优先搜索序列：0, 2, 3, 5, 6, 1, 4
   广度优先搜索序列：0, 2, 3, 5, 6, 1, 4
4. 深度优先搜索序列：0, 3, 6, 4, 1, 5, 2
   广度优先搜索序列：0, 3, 2, 6, 5, 4, 1

5. 过程如图 6-16 所示

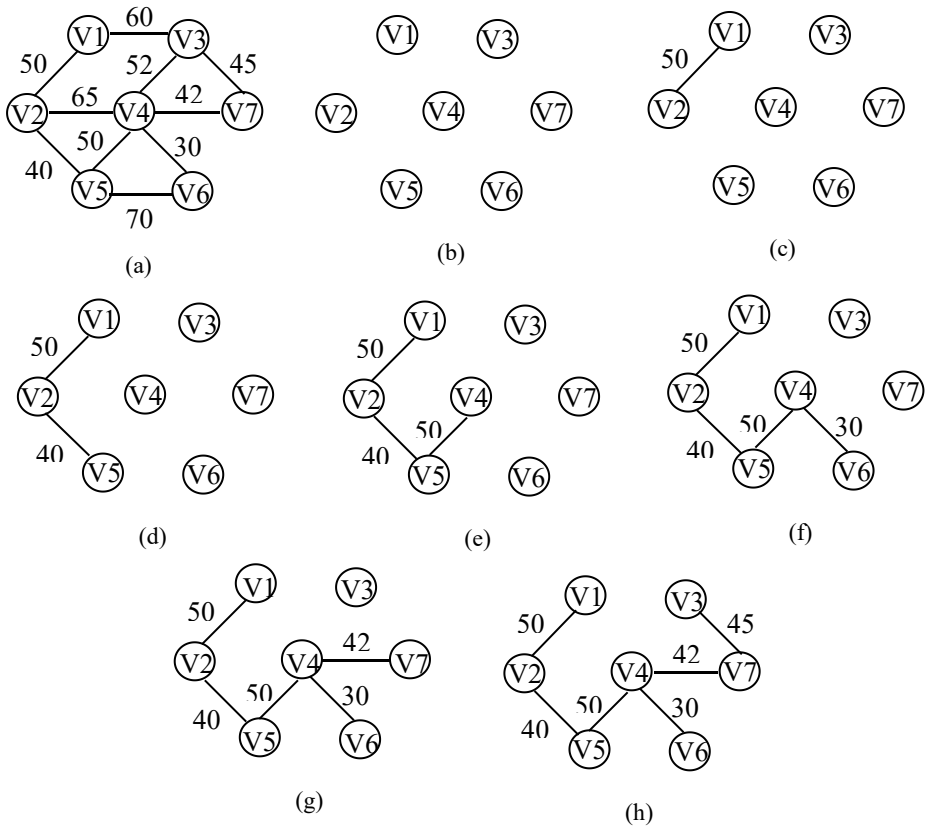


图 6-16

6. 求解过程如图 6-17 所示。

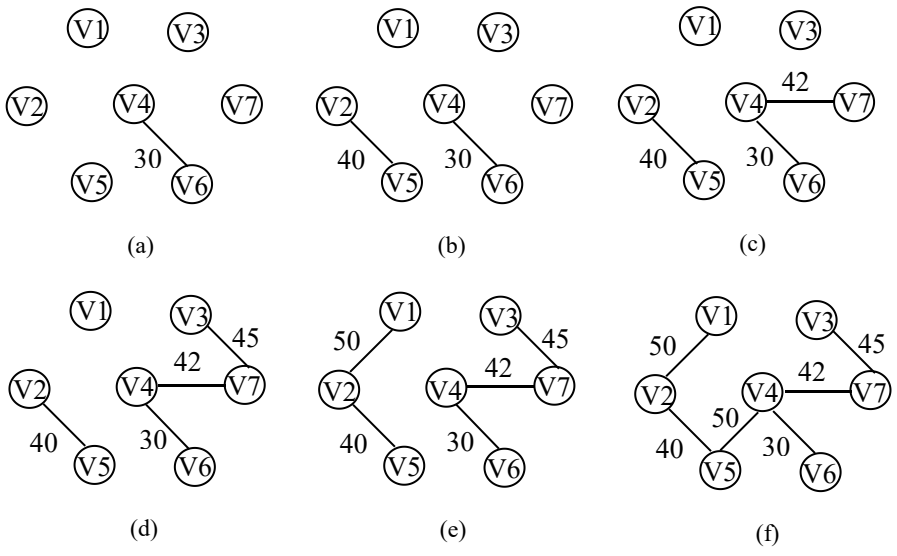


图 6-17

7. 求解过程如下表所示。

终点	从 v0 到各终点的 D 值和最短路径的求解过程				
	i=1	i=2	i=3	i=4	i=5
V1	∞	∞	∞	∞	∞ 无
V2	10 (v0,v2)				
V3	∞	60 (v0,v2,v3)	50 (v0,v4,v3)		
V4	30 (v0,v4)	30 (v0,v4)			
V5	100 (v0,v5)	100 (v0,v5)	90 (v0,v4,v5)	60 (v0,v4,v3,v5)	
Vj	V2	V4	V3	V5	
S	{v0,v2}	{v0,v2,v4}	{v0,v2,v3,v4}	{v0,v2,v3,v4,v5}	

8. 求解过程如下：

①事件的最早发生时间 ve[k]。

ve (1)=0

ve (2)=3

ve (3)=4

ve (4)=ve(2)+2=5

ve (5)=max{ve(2)+1,ve(3)+3}=7

ve (6)=ve(3)+5=9

ve (7)=max{ve(4)+6,ve(5)+8}=15

ve (8)=ve(5)+4=11

ve (9)=max{ve(8)+10,ve(6)+2}=21

ve (10)=max{ve(8)+4,ve(9)+1}=22

ve (11)=max{ve(7)+7,ve(10)+6}=28

②事件的最迟发生时间 vl[k]。

vl (11)= ve (11)=28

vl (10)= vl (11)-6=22

vl (9)= vl (10)-1=21

vl (8)=min{ vl (10)-4, vl (9)-10}=11

vl (7)= vl (11)-7=21

vl (6)= vl (9)-2=19

vl (5)=min{ vl (7)-8,vl (8)-4}=7

vl (4)= vl (7)-6=15

vl (3)=min{ vl (5)-3, vl (6)-5}=4

vl (2)=min{ vl (4)-2, vl (5)-1}=6

vl (1)=min{vl (2)-3, vl (3)-4}=0

③活动 ai 的最早开始时间 e[i]和最晚开始时间 l[i]。

活动 a1	e (1)=ve (1)=0	l (1)=vl (2) -3 =3
活动 a2	e (2)=ve (1)=0	l (2)=vl (3) - 4=0
活动 a3	e (3)=ve (2)=3	l (3)=vl (4) - 2=13
活动 a4	e (4)=ve (2)=3	l (4)=vl (5) - 1=6
活动 a5	e (5)=ve (3)=4	l (5)=vl (5) - 3=4
活动 a6	e (6)=ve (3)=4	l (6)=vl (6) - 5=14
活动 a7	e (7)=ve (4)=5	l (7)=vl (7) - 6=15
活动 a8	e (8)=ve (5)=7	l (8)=vl (7) - 8=13
活动 a9	e (9)=ve (5)=7	l (9)=vl (8) - 4=7
活动 a10	e (10)=ve (6)=9	l (10)=vl (9) - 2=19
活动 a11	e (11)=ve (7)=15	l (11)=vl (11) - 7=21
活动 a12	e (12)=ve (8)=11	l (12)=vl (10) - 4=18
活动 a13	e (13)=ve (8)=11	l (13)=vl (9) - 10=11
活动 a14	e (14)=ve (9)=21	l (14)=vl (10) -1=21
活动 a15	e (15)=ve (10)=22	l (15)=vl (11) -6 =22

④最后，比较 e[i]和 l[i]的值可判断出 a2, a5, a9, a13, a14, a15 是关键活动，关键路径如图 6-18 所示。

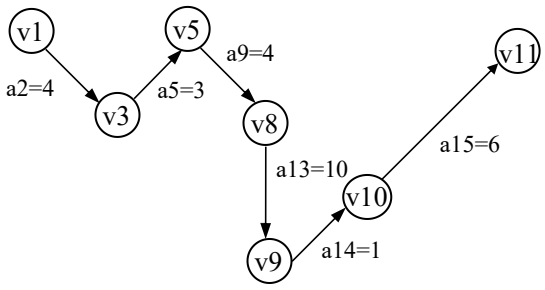


图 6-18

四、算法设计题

1.
```
int degree1(Graph & ga, int numb)
{   //根据无向图的邻接矩阵求出序号为 numb 的顶点的度数
    int j,d=0;
    for(j=0; j<ga.vexnum; j++)
        if (ga.cost[numb][j]!=0 && ga.cost[numb][j]!=MAXINT)
            d++;
```

```
    return (d);
}
2.  int degree2(Graph & ga, int numb)
//根据有向图的邻接矩阵求出序号为 numb 的顶点的度数
{   int i,j,d=0;
    //求出顶点 numb 的出度
    for(j=0; j<ga.vexnum; j++)
        if(ga.cost[numb][j]!=0 && ga.cost[numb][j]!=MAXINT)
            d++;
    //求出顶点 numb 的入度
    for(i=0; i<ga.vexnum; i++)
        if(ga.cost[i][numb]!=0 && ga.cost[i][numb]!=MAXINT)
            d++;
    //返回顶点 numb 的度
    return (d);
}
3.  int degree3(GraphL & gl, int numb)
//根据无向图的邻接表求出序号为 numb 的顶点的度数
{   int d=0;
    vexnode   * p=gl.adjlist[numb];
    while(p!=NULL)
    {   d++;
        p=p->next;
    }
    return (d);
}
4.  int degree4(GraphL & gl, int numb)
//根据有向图的邻接表求出序号为 numb 的顶点的度数
{   int d=0, i;
    vexnode   * p=gl.adjlist[numb];
    while (p!=NULL)
    {   d++;
        p=p->next;
    }   //求出顶点 numb 的出度
    for(i=0; i<gl.vexnum; i++)
    {   p=gl.adjlist[i];
        while(p!=NULL)
        {   if(p->vertex= =numb) d++;
```

```
                p=p->next;
            }
        }//求出顶点 numb 的入度
        return (d);     //返回顶点 numb 的度数
}
```

第七章 查找

一、单项选择题

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

2. 对于长度为 9 的顺序存储的有序表，若采用折半查找，在等概率情况下的平均查找长度为(　　)的 9 分之一。

A. 20　　B. 18　　C. 25　　D. 22

3. 对于长度为 18 的顺序存储的有序表，若采用折半查找，则查找第 15 个元素的比较次数为(　　)。

A. 3　　B. 4　　C. 5　　D. 6

4. 对于顺序存储的有序表(5, 12, 20, 26, 37, 42, 46, 50, 64)，若采用折半查找，则查找元素 26 的比较次数为(　　)。

A. 2　　B. 3　　C. 4　　D. 5

5. 对具有 n 个元素的有序表采用折半查找，则算法的时间复杂度为(　　)。

A. O(n)　　B. $O(n^2)$　　C. O(1)　　D. $O(\log_2 n)$

6. 在索引查找中，若用于保存数据元素的主表的长度为 n，它被均分为 k 个子表，每个子表的长度均为 n/k，则索引查找的平均查找长度为(　　)。

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

7. 在索引查找中，若用于保存数据元素的主表的长度为 144，它被均分为 12 子表，每个子表的长度均为 12，则索引查找的平均查找长度为(　　)。

A. 13　　B. 24　　C. 12　　D. 79

8. 从具有 n 个结点的二叉排序树中查找一个元素时，在平均情况下的时间复杂度大致为(　　)。

A. O(n)　　B. O(1)　　C. $O(\log_2 n)$　　D. $O(n^2)$

9. 从具有 n 个结点的二叉排序树中查找一个元素时，在最坏情况下的时间复杂度为(　　)。

A. O(n)　　B. O(1)　　C. $O(\log_2 n)$　　D. $O(n^2)$

10. 在一棵平衡二叉排序树中，每个结点的平衡因子的取值范围是(　　)。

A. -1~1　　B. -2~2　　C. 1~2　　D. 0~1

11. 若根据查找表(23, 44, 36, 48, 52, 73, 64, 58)建立哈希表，采用 h(K)=K%13 计算哈希地址，则元素 64 的哈希地址为(　　)。

A. 4　　B. 8　　C. 12　　D. 13

12. 若根据查找表(23, 44, 36, 48, 52, 73, 64, 58)建立哈希表，采用 h(K)=K%7 计算哈希

地址，则哈希地址等于 3 的元素个数(　　)。

A. 1　　B. 2　　C. 3　　D. 4

13. 若根据查找表建立长度为 m 的哈希表，采用线性探测法处理冲突，假定对一个元素第一次计算的哈希地址为 d，则下一次的哈希地址为(　　)。

A. d　　B. d+1　　C. (d+1)/m　　D. (d+1)%m

二、填空题

1. 以顺序查找方法从长度为 n 的顺序表或单链表中查找一个元素时，平均查找长度为_____，时间复杂度为_____。

2. 对长度为 n 的查找表进行查找时，假定查找第 i 个元素的概率为 p_i，查找长度（即在查找过程中依次同有关元素比较的总次数）为 c_i，则在查找成功情况下的平均查找长度的计算公式为_____。

3. 假定一个顺序表的长度为 40，并假定查找每个元素的概率都相同，则在查找成功情况下的平均查找长度_____，在查找不成功情况下的平均查找长度_____。

4. 以折半查找方法从长度为 n 的有序表中查找一个元素时，平均查找长度约等于_____的向上取整减 1，时间复杂度为_____。

5. 以折半查找方法在一个查找表上进行查找时，该查找表必须组织成_____存储的_____表。

6. 从有序表(12, 18, 30, 43, 56, 78, 82, 95)中分别折半查找 43 和 56 元素时，其比较次数分别为_____和_____。

7. 假定对长度 n=50 的有序表进行折半查找，则对应的判定树高度为_____，最后一层的结点数为_____。

8. 假定在索引查找中，查找表长度为 n，每个子表的长度相等，设为 s，则进行成功查找的平均查找长度为_____。

9. 在索引查找中，假定查找表（即主表）的长度为 96，被等分为 8 个子表，则进行索引查找的平均查找长度为_____。

10. 在一棵二叉排序树中，每个分支结点的左子树上所有结点的值一定_____该结点的值，右子树上所有结点的值一定_____该结点的值。

11. 对一棵二叉排序树进行中序遍历时，得到的结点序列是一个_____。

12. 从一棵二叉排序树中查找一个元素时，若元素的值等于根结点的值，则表明_____，若元素的值小于根结点的值，则继续向_____查找，若元素的值大于根结点的值，则继续向_____查找。

13. 向一棵二叉排序树中插入一个元素时，若元素的值小于根结点的值，则接着向根结点的_____插入，若元素的值大于根结点的值，则接着向根结点的_____插入。

14. 根据 n 个元素建立一棵二叉排序树的时间复杂度大致为_____。

15. 在一棵平衡二叉排序树中，每个结点的左子树高度与右子树高度之差的绝对值不超过_____。

16. 假定对线性表(38, 25, 74, 52, 48)进行哈希存储，采用 H(K)=K % 7 作为哈希函数，采用线性探测法处理冲突，则在建立哈希表的过程中，将会碰到_____次存储冲突。

17. 假定对线性表(38, 25, 74, 52, 48)进行哈希存储，采用 H(K)=K % 7 作为哈希函数，采用线性探测法处理冲突，则平均查找长度为_____。

18. 在线性表的哈希存储中，装填因子α又称为装填系数，若用 m 表示哈希表的长度，n 表示线性表中的元素的个数，则α等于_____。

19. 对线性表(18, 25, 63, 50, 42, 32, 90)进行哈希存储时，若选用 H(K)=K % 9 作为哈希函数，则哈希地址为 0 的元素有_____个，哈希地址为 5 的元素有_____个。

三、应用题

1. 已知一个顺序存储的有序表为(15, 26, 34, 39, 45, 56, 58, 63, 74, 76)，试画出对应的折半查找判定树，求出其平均查找长度。

2. 假定一个线性表为(38, 52, 25, 74, 68, 16, 30, 54, 90, 72)，画出按线性表中元素的次序生成的一棵二叉排序树，求出其平均查找长度。

3. 假定一个待哈希存储的线性表为(32, 75, 29, 63, 48, 94, 25, 46, 18, 70)，哈希地址空间为 HT[13]，若采用除留余数法构造哈希函数和线性探测法处理冲突，试求出每一元素在哈希表中的初始哈希地址和最终哈希地址，画出最后得到的哈希表，求出平均查找长度。

元素	32	75	29	63	48	94	25	46	18	70
初始哈希地址										
最终哈希地址										

	0	1	2	3	4	5	6	7	8	9	10	11	12
哈希表													

4. 假定一个待哈希存储的线性表为(32, 75, 29, 63, 48, 94, 25, 36, 18, 70, 49, 80)，哈希地址空间为 HT[12]，若采用除留余数法构造哈希函数和拉链法处理冲突，试画出最后得到的哈希表，并求出平均查找长度。

四、算法设计题

1. 试写一个判别给定二叉树是否为二叉排序树的算法，设此二叉树以二叉链表作为存储结构，且树中结点的关键字均不同。

2. 试将折半查找的算法改写成递归算法。

第七章参考答案

一、单项选择题

1. D　2. C　3. B　4. C　5. D　6. D　7. A　8. C　9. A　10. A　11. C　12. B　13. D

二、填空题

1.　(n+1)/2,　O(n)

2.　$\sum_{i=1}^{n} p_i c_i$

3.　20.5,　41

4.$\lceil \log_2(n+1) \rceil$，O($\log_2 n$)

5.　顺序　有序

6. 1，3

7.　6，　19

8. (n/s+s)/2+1

9.　11

10. 小于，大于

11. 有序序列

12. 查找成功，左子树，右子树

13. 左子树，右子树

14. O($n\log_2 n$)

15.　1

16. 5

17.　2

18. n/m

19.　3，　2

三、应用题

1. 折半查找判定树如图 7-3 所示，平均查找长度等于 29/10。图 7-3 中的结点与有序表中元素的对应关系如下表所示。

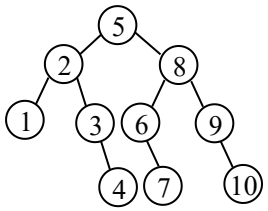


图 7-3

1	2	3	4	5	6	7	8	9	10
15	26	34	39	45	56	58	63	74	76

2. 二叉排序树如图 7-4 所示，平均查找长度等于 32/10。

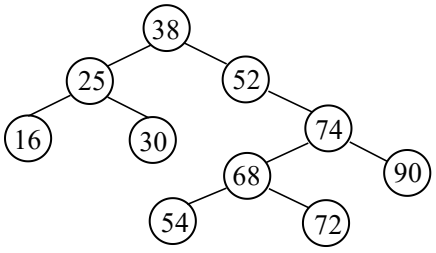


图 7-4

3. H(K)=K % 13 平均查找长度为 14/10，其余解答如下。

元素	32	75	29	63	48	94	25	46	18	70
初始哈希地址	6	10	3	11	9	3	12	7	5	5
最终哈希地址	6	10	3	11	9	4	12	7	5	8

	0	1	2	3	4	5	6	7	8	9	10	11	12
哈希表				29	94	18	32	46	70	48	75	63	25

4. H(K)=K % 11，哈希表如图 7-5 所示，平均查找长度 17/12。

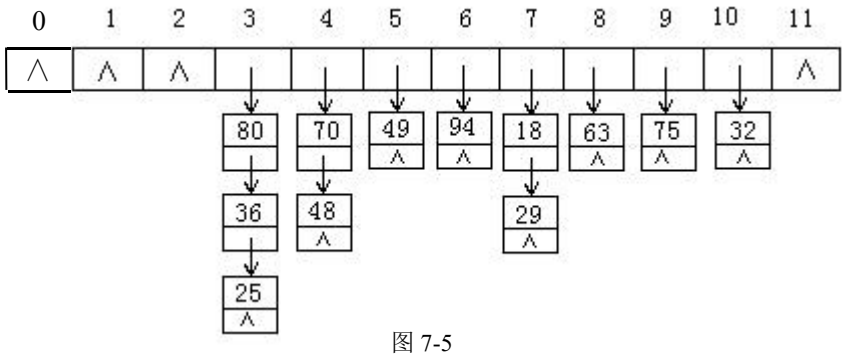


图 7-5

四、算法设计题

1. 设计思路：进入判别算法之前，pre 取初值为 min(小于树中任一结点值)，fail=FALSE, 即认为 bt 是二叉排序树。按中序遍历 bt，并在沿向根结点，与前趋比较，若逆序，则 fail 为 TRUE,则 bt 不是二叉排序树。

```
void bisorttree(bitree bt,keytype pre, bool &fail)
{   //fail 初值为 FALSE，若非二叉序树，则 fail 值 TRUE
    if (!fail)
    {   if (bt)
        {   bisosrttree(bt->lchild,pre,fail);   //判断左子树
            if (bt->data_key<pre)    fail=TRUE;
            else
            {   pre=bt->data_key;
                bisorttree(bt->rchild,pre,fail);   //判断右子树
            }
        }
    }
}   //bisorttree
```

说明：较为直观的方法，可套用中序遍历非递归算法。

2. int search_bin(SeqTable st , keytype k , int low , int high)

```
{   if (low<high)    return (0);   //不成功
    else
    {   mid=(low+high)/2;
        if (k= =st.elem[mid].key)    return (mid) ;   //成功
        else
            if (k<st.elem[mid].key)    return (st,k,low,mid-1);
            else    return(st,k,mid+1,high);
     }
}   //search-bin
```

第八章 排序

一、单项选择题

1. 若对 n 个元素进行直接插入排序，在进行第 i 趟排序时，假定元素 r[i+1]的插入位置为 r[j]，则需要移动元素的次数为（　　）。

A. j-i　　B. i-j-1　　C. i-j　　D. i-j+1

2. 若对 n 个元素进行直接插入排序，则进行任一趟排序的过程中，为寻找插入位置而需要的时间复杂度为（　　）。

A. O(1)　　B. O(n)　　C. $O(n^2)$　　D. $O(\log_2 n)$

3. 在对 n 个元素进行直接插入排序的过程中，共需要进行（　　）趟。

A. n　　B. n+1　　C. n-1　　D. 2n

4. 对 n 个元素进行直接插入排序时间复杂度为（　　）。

A. O(1)　　B. O(n)　　C. $O(n^2)$　　D. $O(\log_2 n)$

5. 在对 n 个元素进行冒泡排序的过程中，第一趟排序至多需要进行（　　）对相邻元素之间的交换。

A. n　　B. n-1　　C. n+1　　D. n/2

6. 在对 n 个元素进行冒泡排序的过程中，最好情况下的时间复杂度为（　　）。

A. O(1)　　B. $O(\log_2 n)$　　C. $O(n^2)$　　D. O(n)

7. 在对 n 个元素进行冒泡排序的过程中，至少需要（　　）趟完成。

A. 1　　B. n　　C. n-1　　D. n/2

8. 在对 n 个元素进行快速排序的过程中，若每次划分得到的左、右两个子区间中元素的个数相等或只差一个，则整个排序过程得到的含两个或两个元素的区间个数大致为（　　）。

A. n　　B. n/2　　C. $\log_2 n$　　D. 2n

9. 在对 n 个元素进行快速排序的过程中，第一次划分最多需要移动（　　）次元素，包括开始把支点元素移动到临时变量的一次在内。

A. n/2　　B. n-1　　C. n　　D. n+1

10. 在对 n 个元素进行快速排序的过程中，最好情况下需要进行（　　）躺。

A. n　　B. n/2　　C. $\log_2 n$　　D. 2n

11. 在对 n 个元素进行快速排序的过程中，最坏情况下需要进行（　　）躺。

A. n　　B. n-1　　C. n/2　　D. $\log_2 n$

12. 在对 n 个元素进行快速排序的过程中，平均情况下的时间复杂度为（　　）。

A. O(1)　　B. $O(\log_2 n)$　　C. $O(n^2)$　　D. $O(n\log_2 n)$

13. 在对 n 个元素进行快速排序的过程中，最坏情况下的时间复杂度为（　　）。

A. O(1)　　B. $O(\log_2 n)$　　C. $O(n^2)$　　D. $O(n\log_2 n)$

14. 在对 n 个元素进行快速排序的过程中，平均情况下的空间复杂度为（　　）。

A. $O(1)$　　B. $O(\log_2 n)$　　C. $O(n^2)$　　D. $O(n\log_2 n)$

15. 在对 n 个元素进行直接插入排序的过程中，算法的空间复杂度为（　　）。

A. $O(1)$　　B. $O(\log_2 n)$　　C. $O(n^2)$　　D. $O(n\log_2 n)$

16. 对下列四个序列进行快速排序，各以第一个元素为基准进行第一次划分，则在该次划分过程中需要移动元素次数最多的序列为（　　）。

A. 1, 3, 5, 7, 9　　B. 9, 7, 5, 3, 1

C. 5, 3, 1, 7, 9　　D. 5, 7, 9, 1, 3

17. 假定对元素序列（7, 3, 5, 9, 1, 12, 8, 15）进行快速排序，则进行第一次划分后，得到的左区间中元素的个数为（　　）。

A. 2　　B. 3　　C. 4　　D. 5

18. 在对 n 个元素进行简单选择排序的过程中，需要进行（　　）趟选择和交换。

A. n　　B. n+1　　C. n-1　　D. n/2

19. 在对 n 个元素进行堆排序的过程中，时间复杂度为（　　）。

A. $O(1)$　　B. $O(\log_2 n)$　　C. $O(n^2)$　　D. $O(n\log_2 n)$

20. 在对 n 个元素进行堆排序的过程中，空间复杂度为（　　）。

A. $O(1)$　　B. $O(\log_2 n)$　　C. $O(n^2)$　　D. $O(n\log_2 n)$

21. 假定对元素序列（7, 3, 5, 9, 1, 12）进行堆排序，并且采用小根堆，则由初始数据构成的初始堆为（　　）。

A. 1, 3, 5, 7, 9, 12　　B. 1, 3, 5, 9, 7, 12

C. 1, 5, 3, 7, 9, 12　　D. 1, 5, 3, 9, 12, 7

22. 假定一个初始堆为(1, 5, 3, 9, 12, 7, 15, 10)，则进行第一趟堆排序后得到的结果为（　　）。

A. 3, 5, 7, 9, 12, 10, 15, 1　　B. 3, 5, 9, 7, 12, 10, 15, 1

C. 3, 7, 5, 9, 12, 10, 15, 1　　D. 3, 5, 7, 12, 9, 10, 15, 1

23. 若对 n 个元素进行归并排序，则进行归并的趟数为（　　）。

A. n　　B. n-1　　C. n/2　　D. $\lceil \log_2 n \rceil$

24. 若一个元素序列基本有序，则选用（　　）方法较快。

A. 直接插入排序　　B. 简单选择排序

C. 堆排序　　D. 快速排序

25. 若要从 1000 个元素中得到 10 个最小值元素，最好采用（　　）方法。

A. 直接插入排序　　B. 简单选择排序

C. 堆排序　　D. 快速排序

26. 若要对 1000 个元素排序，要求既快又稳定，则最好采用（　　）方法。

A. 直接插入排序　　B. 归并排序　　C. 堆排序　　D. 快速排序

27. 若要对 1000 个元素排序，要求既快又节省存储空间，则最好采用（　　）方法。

A. 直接插入排序　　B. 归并排序　　C. 堆排序　　D. 快速排序

28. 在平均情况下速度最快的排序方法为（　　）。

A. 简单选择排序　　B. 归并排序　　C. 堆排序　　D. 快速排序

二、填空题

1. 每次从无序子表中取出一个元素，把它插入到有序子表中的适当位置，此种排序方法叫做_____排序；每次从无序子表中挑选出一个最小或最大元素，把它交换到有序表的一端，此种排序方法叫做_____排序。

2.每次直接或通过支点元素间接比较两个元素，若出现逆序排列时就交换它们的位置，此种排序方法叫做_____排序；每次使两个相邻的有序表合并成一个有序表的排序方法叫做_____排序。

3. 在简单选择排序中，记录比较次数的时间复杂度为_____，记录移动次数的时间复杂度为_____。

4. 对 n 个记录进行冒泡排序时，最少的比较次数为_____，最少的趟数为_____。

5. 快速排序在平均情况下的时间复杂度为_____，在最坏情况下的时间复杂度为_____。

6. 若对一组记录(46,79,56,38,40,80,35,50,74)进行直接插入排序，当把第 8 个记录插入到前面已排序的有序表时，为寻找插入位置需比较_____次。

7. 假定一组记录为(46,79,56,38,40,84)，则利用堆排序方法建立的初始小根堆为_____。

8. 假定一组记录为(46,79,56,38,40,84)，在冒泡排序的过程中进行第一趟排序后的结果为_____。

9. 假定一组记录为(46,79,56,64,38,40,84,43)，在冒泡排序的过程中进行第一趟排序时，元素 79 将最终下沉到其后第_____个元素的位置。

10. 假定一组记录为(46,79,56,38,40,80)，对其进行快速排序的过程中，共需要_____趟排序。

11. 假定一组记录为(46,79,56,38,40,80)，对其进行快速排序的过程中，含有两个或两个以上元素的排序区间的个数为_____个。

12. 假定一组记录为(46,79,56,25,76,38,40,80)，对其进行快速排序的第一次划分后，右区间内元素的个数为_____。

13. 假定一组记录为(46,79,56,38,40,80)，对其进行快速排序的第一次划分后的结果为_____。

14. 假定一组记录为(46,79,56,38,40,80,46,75,28,46)，对其进行归并排序的过程中，第二趟归并后的子表个数为_____。

15. 假定一组记录为(46,79,56,38,40,80,46,75,28,46)，对其进行归并排序的过程中，第三趟归并后的第 2 个子表为_____。

16. 假定一组记录为(46,79,56,38,40,80,46,75,28,46)，对其进行归并排序的过程中，供需要_____趟完成。

17. 在时间复杂度为 $O(n\log_2 n)$ 的所有排序方法中，_____排序方法是稳定的。

18. 在时间复杂度为 $O(n^2)$ 的所有排序方法中，_____排序方法是不稳定的。

19. 在所有排序方法中，_____排序方法采用的是二分法的思想。

20. 在所有排序方法中，_____方法使数据的组织采用的是完全二叉树的结构。

21. 在所有排序方法中，_____方法采用的是两两有序表合并的思想。

22. _____排序方法使键值大的记录逐渐下沉，使键值小的记录逐渐上浮。

23. _____排序方法能够每次使无序表中的第一个记录插入到有序表中。

24. _____排序方法能够每次从无序表中顺序查找出一个最小值。

三、应用题

1. 已知一组记录为(46,74,53,14,26,38,86,65,27,34)，给出采用直接插入排序法进行排序时每一趟的排序结果。

2. 已知一组记录为(46,74,53,14,26,38,86,65,27,34)，给出采用冒泡排序法进行排序时每一趟的排序结果。

3. 已知一组记录为(46,74,53,14,26,38,86,65,27,34)，给出采用快速排序法进行排序时每一趟的排序结果。

4. 已知一组记录为(46,74,53,14,26,38,86,65,27,34)，给出采用简单选择排序法进行排序时每一趟的排序结果。

5. 已知一组记录为(46,74,53,14,26,38,86,65,27,34)，给出采用堆排序法进行排序时每一趟的排序结果。

6. 已知一组记录为(46,74,53,14,26,38,86,65,27,34)，给出采用归并排序法进行排序时每一趟的排序结果。

四、算法设计题

1. 编写一个双向起泡的排序算法，即相邻两趟向相反方向起泡。

2. 试以单链表为存储结构实现简单选择排序的算法。

第八章参考答案

一、单项选择题

1. D　2. B　3. C　4. C　5. B　6. D　7. A　8. B　9. D　10. C　11. B　12. D　13. C　14. B
15. A　16. D　17. B　18. C　19. D　20. A　21. B　22. A　23. D　24. A　25. C　26. B　27. C
28. D

二、填空题

1. 插入，选择
2. 快速，归并
3. $O(n^2)$，$O(n)$
4. n−1，1
5. $O(n\log_2 n)$，$O(n^2)$
6. 4
7. (38,40,56,79,46,84)
8. (46,56,38,40,79,84)
9. 4
10. 3
11. 4
12. 4
13. [40 38] 46 [56 79 80]
14. 3
15. [28 46]
16. 4
17. 归并
18. 直接选择
19. 快速
20. 堆排序
21. 归并排序
22. 冒泡
23. 直接插入
24. 直接选择

三、应用题

1.

(0) [46] 74 53 14 26 38 86 65 27 34
(1) [46 74] 53 14 26 38 86 65 27 34
(2) [46 53 74] 14 26 38 86 65 27 34
(3) [14 46 53 74] 26 38 86 65 27 34
(4) [14 26 46 53 74] 38 86 65 27 34
(5) [14 26 38 46 53 74] 86 65 27 34
(6) [14 26 38 46 53 74 86] 65 27 34
(7) [14 26 38 46 53 65 74 86] 27 34
(8) [14 26 27 38 46 53 65 74 86] 34
(9) [14 26 27 34 38 46 53 65 74 86]

2.

(0) [46 74 53 14 26 38 86 65 27 34]
(1) [46 53 14 26 38 74 65 27 34] 86
(2) [46 14 26 38 53 65 27 34] 74 86
(3) [14 26 38 46 53 27 34] 65 74 86
(4) [14 26 38 46 27 34] 53 65 74 86
(5) [14 26 38 27 34] 46 53 65 74 86
(6) [14 26 27 34] 38 46 53 65 74 86
(7) [14 26 27 34] 38 46 53 65 74 86

3.

(0) [46 74 53 14 26 38 86 65 27 34]
(1) [34 27 38 14 26] 46 [86 65 53 74]
(2) [26 27 14] 34 38 46 [74 65 53] 86
(3) 14 26 27 34 38 46 [53 65] 74 86
(4) 14 26 27 34 38 46 53 65 74 86

4.

(0) [46 74 53 14 26 38 86 65 27 34]
(1) 14 [74 53 46 26 38 86 65 27 34]
(2) 14 26 [53 46 74 38 86 65 27 34]
(3) 14 26 27 [46 74 38 86 65 53 34]
(4) 14 26 27 34 [74 38 86 65 53 46]
(5) 14 26 27 34 38 [74 86 65 53 46]
(6) 14 26 27 34 38 46 [86 65 53 74]
(7) 14 26 27 34 38 46 53 [65 86 74]
(8) 14 26 27 34 38 46 53 65 [86 74]
(9) 14 26 27 34 38 46 53 65 74 [86]

5. 构成初始堆（即建堆）的过程：

	1	2	3	4	5	6	7	8	9	10
(0)	46	74	53	14	26	38	86	65	27	34
(1)	46	74	53	14	26	38	86	65	27	34
(2)	46	74	53	14	26	38	86	65	27	34
(3)	46	74	38	14	26	53	86	65	27	34
(4)	46	14	38	27	26	53	86	65	74	34
(5)	14	26	38	27	34	53	86	65	74	46

进行堆排序的过程：

(0) 14 26 38 27 34 53 86 65 74 46
(1) 26 27 38 46 34 53 86 65 74 [14]
(2) 27 34 38 46 74 53 86 65 [26 14]
(3) 34 46 38 65 74 53 86 [27 26 14]
(4) 38 46 53 65 74 86 [34 27 26 14]
(5) 46 65 53 86 74 [38 34 27 26 14]
(6) 53 65 74 86 [46 38 34 27 26 14]
(7) 65 86 74 [53 46 38 34 27 26 14]
(8) 74 86 [65 53 46 38 34 27 26 14]
(9) 86 [74 65 53 46 38 34 27 26 14]

6.

(0) [46] [74] [53] [14] [26] [38] [86] [65] [27] [34]
(1) [46 74] [14 53] [26 38] [65 86] [27 34]
(2) [14 46 53 74] [26 38 65 86] [27 34]
(3) [14 26 38 46 53 65 74 86] [27 34]
(3) [14 26 27 34 38 46 53 65 74 86]

四、算法设计题

1.

```
void Bubble_Sort2(int a[ ],int n)   //相邻两趟是反方向起泡的冒泡排序算法
{   low=0;high=n-1;   //冒泡的上下界
    change=1;
    while (low<high&&change)
    {   change=0;
        for(i=low;i<high;i++)   //从上向下起泡
        if (a[i]>a[i+1])
        {   a[i]<->a[i+1];
            change=1;
         }
        high--;   //修改上界
        for (i=high;i>low;i--)   //从下向上起泡
            if (a[i]<a[i-1])
            {   a[i]<->a[i-1];
                change=1;
            }
        low++;    //修改下界
    }//while
}//Bubble_Sort2
```

2.

```
void LinkList_Select_Sort(LinkList &L)   //单链表上的简单选择排序算法
{   for (p=L;p->next->next;p=p->next)
    {  q=p->next;   x=q->data;
       for (r=q,s=q;r->next;r=r->next)   //在 q 后面寻找元素值最小的结点
           if (r->next->data<x)
           {  x=r->next->data;
              s=r;
           }
       if (s!=q)   //找到了值比 q->data 更小的最小结点 s->next
       {   p->next=s->next;   s->next=q;
           t=q->next;   q->next=p->next->next;
           p->next->next=t;
       }   //交换 q 和 s->next 两个结点
    }//for
 }//LinkList_Select_Sort
```