第二~五章习题课

第二章作业：绪论

3、**以下说法正确的是（ ）。**

错误选项：

顺序存储方式只能用于线性结构，不能用于非线性结构（例如完全二叉树）

数据的逻辑结构是指数据的各数据项之间的逻辑关系（数据元素间）

正确选项：

基于某种逻辑结构上的运算，其实现与其存储结构有关

8、

下列函数的时间复杂度是（ ）。

```
int func(int n)
{
int i=0, sum=0;
while(sum<n) sum+=++i;
return i;
}
```

错选答案：　O(n)

正确答案：　$O(n^{1/2})$

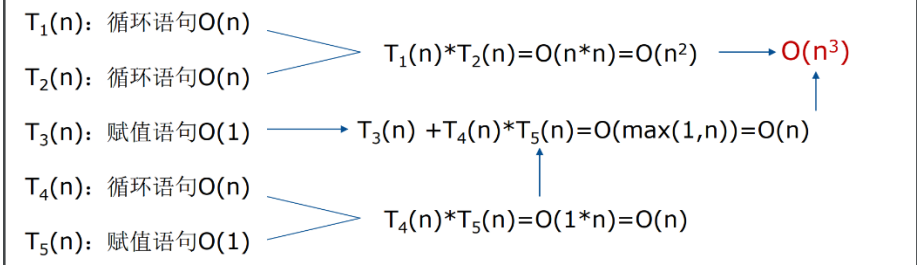
时间复杂度概念：假设算法的问题规模为 n，使用函数 f(n)表示操作单元数量，记算法运行时间 T：

$$T(n) = O(f(n))$$

随着 n 的增大趋于∞，算法执行时间 T 的增长率和 f(n)的增长率渐进同阶，$T(n)$称作为算法的渐近时间复杂度，简称时间复杂度。时间复杂度是算法时间复杂度的宏观定性评价，可以方便开发者估算出程序的运行时间。

```
以下算法的时间复杂度为 O(log₂n)

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


/*基本运算（执行频率最高的语句）i=i*2（每次执行一次i*2），又2^i≤n;
 设执行次数为t，则判断条件可理解为2^t=n,
 即t=log₂n,则T(n)=O(log₂n)*/
```

```
1     for (i=1; i<=n; i++)                                    //T1(n)
2         for (j=1; j<=n; j++)                                //T2(n)
3             c[i][j] = 0;                                    //T3(n)
4             for (k=1; k<=n; k++)                            //T4(n)
5              c[i][j] = c[i][j] + a[i][k] * b[k][j];//T5(n)
```



10、**以下说法正确的是（ ）。**

错选答案：　算法的可行性是指算法的指令不能有二义性（确定性、每一步都能通过有限次数完成）

正确答案：　同一个算法，实现语言的级别越高，执行效率就越低

15、**评价算法的优劣，主要考虑算法的____和____这两方面。**

错误答案：间接性、复杂性

正确答案：时间、空间复杂度

18、19 题目要求算法实现和时间复杂度分析，题目没要求考虑算法的效率

18、公元前 5 世纪，我国古代数学家张丘建在《算经》一书中提出了“百鸡问题”：鸡翁一值钱五，鸡母一值钱三，鸡雏三值钱一，百钱买百鸡，问鸡翁、鸡母、鸡雏各几何？请设计一个算法求解该问题，并分析算法的时间复杂度。

时间复杂度：$O(n^3)$

```cpp
#include<iostream>
using namespace std;

int main(){
    int x,y,z;
    for(x=0;x<=20;x++)//公鸡不会超过20只
        for(y=0;y<=34;y++)//母鸡不会超过34只
            for(z=0;z<=100;z++)//小鸡不会超过100只
                if((x+y+z)==100 && (5*x+3*y+z/3)==100&&z%3==0)//同时要满足z是3的整数
                    cout<<"x="<<x<<" "<<"y="<<y<<" "<<"z="<<z<<endl;
    return 0;
}
```

```
x=0 y=25 z=75
x=4 y=18 z=78
x=8 y=11 z=81
x=12 y=4 z=84
Press any key to continue
```

时间复杂度：$O(n^2)$

```cpp
#include<iostream>
using namespace std;
int main(){
    int x,y,z;
    for(x=0;x<=20;x++)//公鸡不会超过20只
        for(y=0;y<=34;y++)//母鸡不会超过34只
        {
            z=100-x-y;//根据x,y可求出z
            if(z==(300-15*x-9*y)&&z%3==0)
                cout<<"x="<<x<<" "<<"y="<<y<<" "<<"z="<<z<<endl;
        }
    return 0;
}
```

时间复杂度：$O(n)$

```cpp
#include<iostream>
using namespace std;
int main(){
    int x,y,z;
    for(x=0;x<=20;x++)//公鸡不会超过20只
    {
        y=25-1.75*x;//先求y
        z=100-x-y;//再求z
        if(z==(300-15*x-9*y)&&z%3==0&&y>0&&&z>0)
            cout<<"x="<<x<<" "<<"y="<<y<<" "<<"z="<<z<<endl;
    }
    return 0;
}
```

第三章作业：线性表

5、单链表中，增加一个头结点的目的是（ ）。

错误选项:　A.标识表中首结点的位置

正确选项：C.方便运算的实现

9、设一个链表最常用的操作是在末尾插入结点和删除结点，则选用（ ）最节省时间

错选答案:B.带尾指针的单循环链表

　　D.单链表

正确答案:A.带头结点的双循环链表

B:插入容易操作，但是删除不好操作，需要找到末尾节点的前一个节点

10、线性表的静态链表存储结构与顺序存储结构相比，优点是（ ）。

错选答案: A.便于利用零散的存储器空间

正确答案:D.便于插入和删除

11、**顺序存储结构是通过存储位置相邻表示元素之间的关系的，链式存储结构是通过指针表示元素之间的关系的。**

12、对一个线性表分别进行遍历和逆置运算，其最好的时间复杂度分别是 O(n)和 **O(1)**。
绝大多数同学逆置复杂度都写 O(n)，实际上线性表包含双向链表，对双向链表做逆置，认为只需要修改头结点后指针的指向为末尾结点即可。中间的结点前后指针不需要改变。

17、在单链表、双链表和单循环链表中，若仅知道指针 p 指向某结点，而不知道头指针，是否可以将结点 p 从相应的链表中删除，并说明其时间复杂度。
单链表：可以。交换结点 p 和结点 p 下一节点内容，后续依次循环至尾结点，然后删除最后一结点。时间复杂度为 O(n)。
双链表：可以。其时间复杂度为 O(1)。
单循环链表：可以。其时间复杂度为 O(n)。

18、分析线性表顺序存储结构和链式存储结构的优缺点，说明何时应该利用哪种结构。
顺序存储结构：1、可以任意存取数据元素，访问时间开销 O(1)；存储密度高，数据结构逻辑关系表达不占用额外存储空间
2、插入删除需移动大量数据元素，平均时间开销 O(n)；静态固定存储空间分配，可能导致存储空间浪费
链式存储结构：1、每个结点存储空间在物理位置上不要求地址连续；插入删除操作方便，时间复杂度为 O(1)；不必事先预知表长，可动态利用计算机内存空间
2、只能顺序存取，不具备随机存取；链表结点增加指示元素间关系的指针域，存储密度相对低一些
应在存取任意指定序号的元素和在最后插入与删除操作时使用线性表顺序存储结构;在连续存储空间不足、表长随程序运行不断变化或需要多次插入删除操作时候使用线性表链式存储结构。

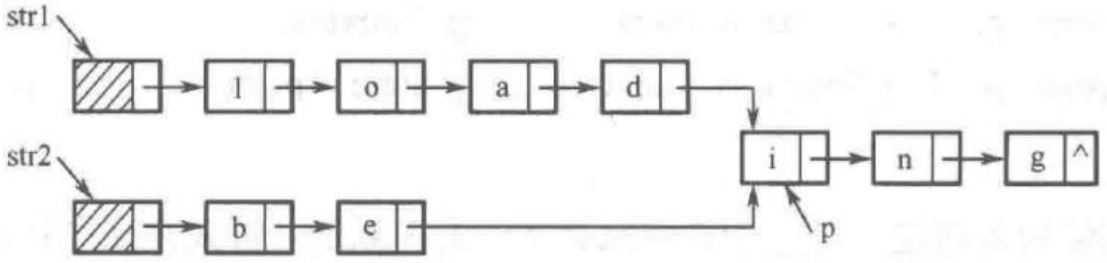
19、设将 n(n>1)个整数存放到一维数组 R 中。试设计一个在时间和空间两方面都尽量高效的算法，将 R 中保存的序列循环左移 p(n>p>0)个位置，即将 R 中的数据由（x0,x1,…xp-1,xp,xp+1,…xn-1）变换为(xp,xp+1,…xn-1,x0,x1,…xp-1)。

```
void leftslidingk(vector<int> nums,int k){
   reverse(nums,0,nums.size()-1);
   reverse(nums,0,k);
   reverse(nums,k+1,nums.size()-1);
}
//反序，in place的解决方案就是swap
void reverse(vector<int> nums,int left,int right){

       for(;left<right;++left,--right)

            swap(nums[left],nums[right]);

}
```

时间复杂度为O(N)，空间为O（1）

有些同学使用循环链表，遍历到 p 位置后将头节点指向该节点也可以。本题中要求时间和空间复杂度都尽量高效，因此若时间空间复杂度不满足最优都酌情扣分。

20、假设采用带头结点的单链表保存单词（每个数据元素仅为一个字符），当两个单词有相同的后缀时，可共享相同的后缀存储空间。设 str1 和 str2 分别指向两个单词所在单链表的头结点，链表结点结构为（data，next），请设计一个在时间方面尽量高效的算法，找出由 str1 和 str2 所指的两个链表共同后缀的起始位置 p。例如 loading 和 being 的存储映像中，字符"i"、"n"和"g"所在 3 个结点空间共享，p 指向字符"i"所在结点。



1.对于链表 str1 和 str2，分别求出两个链表的长度 m 和 n；
2.p 指向 str1 的头节点，q 指向 str2 的头节点，当 m 大于等于 n，p 向后移动 m-n+1 个结点，当 n 大于 m，q 向后移 n-m+1，总之，**要保证 p 和 q 所指的节点到链表尾的长度相等；**
3.反复将指针 p 和 q 同步后移，并判断它们是否指向同一个节点，若指向同一点，则返回共同后缀的起始位置。

```
LinkNode *Find_1st_Common(LinkList str1,LinkList str2){
    int len1=Length(str1),len2=Length(str2);
    LinkNode *p,*q;
    for(p=str1;len1>len2;len1--)   //使 p 指向的链表与 q 指向的链表等长
        p=p->next;
    for(q=str2;len1<len2;len2--)   //使 q 指向的链表与 p 指向的链表等长
        q=q->next;
    while(p->next!=NULL&&p->next!=q->next){// 查找共同后缀起始点
        p=p->next;                 //两个指针同步向后移动
        q=q->next;
    }
    return p->next;                //返回共同后缀的起始点
}
```

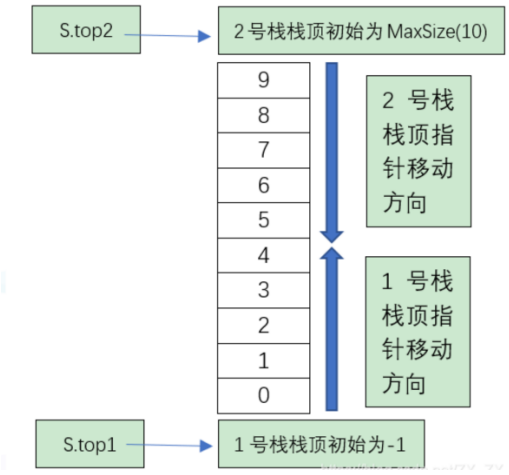
时间复杂度是 O(m+n)

注：将链表存入数组中再从后向前比较元素是否相同可能会出错：

loving 和 giving，两个 v 在链表中存储位置应不同，放入数组中元素相同。应比较指针指向的结构是否相同，而不是比较结构中存储的元素。

第四章作业：栈和队列

14、为了增加内存空间的利用率和减少溢出的可能性，可由两个栈共享长度为 n 的一片连续存储空间。若其中第一个栈的栈顶指针 top1 的初值设为-1，则第二个栈的栈顶指针 top2 的初值应设为 n 较为合理，且判断该共享栈满的条件应该是 top1+1==top2。



15、引入循环队列的目的是为了假溢出（时移动大量数据元素）克服现象。对于循环向量长度为 M 的循环队列 Q，则使用队头指针 Q.front 和队尾指针 Q.rear 求队列长度的公式为 （Q.rear-Q.front+M)%M 。

16、什么是递归？递归程序有什么优缺点？

递归：在定义（函数）内直接或间接调用自身的定义（函数），称为递归定义（函数），一般由递归规则和终止条件两部分构成。其中，递归规则是将原问题拆解为性质完全相同

的若干子问题，且合并子问题的解即可获得原问题的解；终止条件是指存在最小子问题，对其求解可不必再分解。
递归程序的优点：程序简洁易读，实现代码量大大减少；系统可通过自主管理一个递归工作栈，完成递归调用。
递归程序的缺点：
1）空时复杂度高：使用递归工作栈需消耗大量参数入栈出栈及额外的内存空间。每一次函数调用，都需要在内存栈中分配空间以保存参数、返回地址以及临时变量，而往栈中压入数据和弹出数据都需要时间。
2）相对难以想到递归函数的逻辑，有时也很难调试递归代码。
3）可能存在重复计算，如求斐波那契数列的递归实现。

18、设长度为 n 的链队列用单循环链表表示，若只设置头指针，则入队、出队操作的时间复杂度为多少？若只设置尾指针呢？

只设置头指针：

入队操作：时间复杂度为 O（n）

出队操作：如果带头结点，直接删除头结点的后继结点，时间复杂度为 O（1）。如果不带头结点，需要先找到尾结点，再删除首元结点，头指针指向下一结点，再让尾结点的 next 指针指向新的首元结点使之首尾闭合，所以时间复杂度为 O（n）

只设置尾指针： 入队操作时间复杂度为 O（1）,出队操作时间复杂度为 O（1）。

20、从长度为 n（n>0）的数组 A 中选出 k 个元素（1<=k<=n），已知 A 中元素的值各不相同，请编写一个算法输出其全排列。

```
1.  void print(char array[], int n, int k) {
2.      CreateStack(S);
3.      m = 0;
4.      do {
5.          while (S.size < k && m < n){
6.              Push(S, m);
7.              m++;
8.          }
9.          if (S.size = k)
10.             TraverseStack(S, array);
11.             //S 中存的是 array 的位序，此处应打印出位序对应的 array 中的元素
12.         m = Pop(S);
13.         m++;
14.     } while (!IsStackEmpty(S) && m <= n)
15. }
```

本题原意是从 n 个数中选择 k 个元素输出所有可能的组合即可，属于题意不清，很多同学理解为输出全部可能的全排列也可以。一般实现思路是递归得到所有的组合可能，然后对每种组合方式全排列。只要代码意思表达清楚都给了满分。

第五章作业：串与数组

2、**设主串 s=”acbcabcacbab”，模式串 t=”abcac”，利用 BF 算法进行模式匹配的过程中，进行字符串比较的次数总和为（）。**
正确答案：A.10
错选答案：B.8

6、**二维数组 a[14][9]采用行优先的存储方法，若每个元素占两个存储单元，第一个元素的首地址为 1，则 a[3][3]的地址为（ ）。**
正确答案：A.61
涉及计算的题目中矩阵下标从第 1 位开始，数组从 0 位开始

8、**以下说法正确的是（B）。**
正确选项：B.稀疏矩阵压缩存储后，必会失去随机存取功能
错误选项：A.采用三元组存储稀疏矩阵，实现转置运算很简单，原则上只需将每个三元组的行标和列标交换即可（交换行列后还要重排三元组间的次序）
C.三元组存储比十字链表法更高效(十字链表法便于实现增加或减少元素操作)

9、**若对 n 阶对称矩阵 A 以行序为主序方式将其下三角形的元素依次存放于一维数组 B 中，则在 B 中确定 aij (i<j) 的位置 k 的公式为（A）。**
正确答案：A. j*(j-1)/2+i
错选答案：B. i*(i-1)/2+j
C. j*(j+1)/2+i
矩阵首个元素为 a11

12、下列函数的功能是实现两个字符串的比较，试根据字符串比较运算的定义，完善该函数：

```
int strcmp(char s[], char t[]) {
    int i;
    for (i=0; s[i]&&t[i]; i++)
        if (s[i]!=t[i])    return s[i]-t[i];
    return s[i]-t[i];
}
```

逐位比较，按照 ASCII 码值比较

15、数组 M 中每个元素长度为 3 个字节，行下标 i 从 0-7，列下标 j 从 0-9，从首地址 EA 开始连续存放在存储器中。若按行优先方式存放，元素 M[7][5]的起始地址为 EA+225（0x1CB）；若按列优先方式存储，则起始地址为 EA+141（0x177）。

16、假设有如下的串说明：
char s1[30]=”Stocktom,CA”, s2[30]=”March 5,1999”, s3[30];
则分别调用函数 strcmp(s1,s2)、strcmp(&S1[5],”ton”)、strlength(strconcat(s1,s2))的返回值是什么？
6; -1; 23

&s1[5]表示指向以 s1[5]为首地址的字符串

17、设有 n 阶上三角矩阵 A，将其上三角元素 aij 逐行存于数组 B[m]中(m 足够大)，使得 B[k]=aij，且 k=f(i)+g(j)+c。试推导函数 f、g 和常数 c。

f(i)= -1/2i^2+(1/2+n)i , g(j)=j, c=-n-1。

上三角矩阵第 k 行需要被存储的元素个数为：$n-k+1$

第 1 到 i-1 行总共需被存储元素个数为：$n+(n-1)+\cdots+(n-(i-1)+1)=\frac{(2n-i+2)(i-1)}{2}$

第 i 行，第 j 个元素前需要被储存元素个数为：$(n-i+1)-(n-j+1)=j-i$

因此：$k=\frac{(2n-i-2)(i-1)}{2}+j-i=-\frac{1}{2}i^2+\frac{1}{2}i+j-n-1$

BF 和 KMP 算法

①KMP 算法、BF 算法在遍历过程中不同之处

BF 算法思路直观简明，但匹配失败主串指针 i 回溯到 i-j+2，子串指针会输到首字符 j=1，因此算法时间复杂度高。

KMP 算法是对 BF 算法的改进，改进在于每当一趟匹配过程中出现字符比较不等时，不需回溯主串 i 指针，而是利用已经得到的“部分匹配”结果将模式向右滑动尽可能远一段距离，继续进行比较。KMP 算法最大特点是主串指针不需回溯，整个匹配过程对主串仅需从头到尾扫描一遍。

②时间复杂度比较（13 题）

假设主串长度为 n，子串长度为 m。

BF 在最好情况下平均时间复杂度是$O(n+m)$，最坏情况下平均时间复杂度是$O(n\times m)$。实际执行时间近似于$O(n+m)$，这是由于最坏情况特殊性较大。

KMP 只有当主串和子串间存在许多“部分匹配”情况下才比 BP 算法快得多，遍历匹配过程和 next 数组求解过程时间复杂度分别是$O(n)$和$O(m)$，因此整体执行时间复杂度为$O(n+m)$，空间复杂度为$O(m)$。