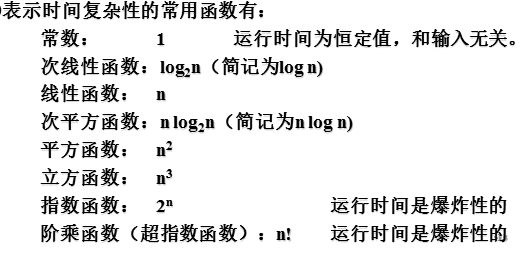
**实验内容**

**第一部分、算法的时间复杂度**

算法的执行时间依赖于具体的软硬件环境，所以，不能用执行时间的长短来衡量算法的时间复杂度，而要通过基本语句执行次数的数量级来衡量。  
  
　一、　求解算法的时间复杂度的具体步骤是：  
  
　　⑴ 找出算法中的基本语句；  
  
　　算法中执行次数最多的那条语句就是基本语句，通常是最内层循环的循环体。  
  
　　⑵ 计算基本语句的执行次数的数量级；  
  
　　只需计算基本语句执行次数的数量级，这就意味着只要保证基本语句执行次数的函数中的最高次幂正确即可，可以忽略所有低次幂和最高次幂的系数。这样能够简化算法分析，并且使注意力集中在最重要的一点上：增长率。  
  
　　⑶ 用大Ο记号表示算法的时间性能。  
  
　　将基本语句执行次数的数量级放入大Ο记号中。

二、常用的算法时间复杂度描述方法



请思考，上述函数的具体含义。

三、常见的算法时间复杂度由小到大依次为：  
  
　　Ο(1)＜Ο(log2n)＜Ο(n)＜Ο(nlog2n)＜Ο(n2)＜Ο(n3)＜…＜Ο(2n)＜Ο(n!)

四、练习

请计算以下各段代码的算法复杂度。

X=1;

for (i=1; i<=n; i++)  
　　    x++;

1. x=a;

a=b;

b=x;

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

sum=0；                 
    for(i=1;i<=n;i++)        
       for(j=1;j<=n;j++)   
         sum++；

6.

for (i=0;i<n;i++){

for (j=I;j<n;j++){

if a[i]>a[j]

{ temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}

}

7.  
  
   for (i=1;i<n;i++)  
   {   
       y=y+1;           
       for (j=0;j<=(2\*n);j++)      
          x++;                
   }           
8.  
                                         
   a=0;  
   b=1;                       
   for ( i=1;i<=n; i++)   
   {    
      s=a +b;   
      b=a;　　　　　    
      a=s;　　　　　  
   }                                                                              
9.   
   i=1;         
   while (i<=n)  
      i=i\*2;   
  
10.    
   for(i=0;i<n;i++)  
   {    
      for(j=0;j<i;j++)    
      {  
         for(k=0;k<j;k++)  
            x=x+2;    
      }  
   }

**第二部分、算法练习---枚举法**

（一）

1. 什么是枚举法？特点？
2. 枚举法的一般步骤？
3. 枚举法应用场景
4. 枚举法适用局限

针对以上问题，学习、理解枚举算法，并整理成文字。

（二）编程或描述出以下各问题的解决方法

1．一个奇异三位数

一个自然数的七进制表达式是一个三位数，而这个自然数的九进制表示也是一个三位数，且这两个三位数的数码顺序正好相反，求这个三位数。

2．

从键盘上任意输入一个正整数，如何判断该正整数是否是另一个整数的平方。

3．猜数游戏：

有一个4位的整数，并满足以下条件：

1） 数的前两位数字是相同的；

2）数的后两位数字也是相同的，但与前两位不同；

3）此数刚好是一个整数的平方。

请根据以上条件尝试找出这个数是多少。

1. 设有一个数 X，当 X 加上100后得到的数 Y 是一个正整数的平方，然后用 Y 再加上68，又是另外一个正整数的平方，请问，如何找出 X 是多少?
2. 若一个口袋中放有 12 个球，其中有 3 个红的， 3 个白的和 6 个黑的，问从中任取 8 个共有多少种不同的颜色搭配？