时间复杂度十道练习题目

1、分析以下时间复杂度

```
void fun(int n)
{
   int i=0,s=0;
   while(s<n)
   {
      ++i;
      s=s+i;
   }
}
```

分析：

n为规模，基本操作语句是++i和s=s+i，while循环处当s>=n不符合条件停止，假设执行m次结束，i=1,2,3..依次渐加，i只影响s值，主要看s，$s_1 =1$，$s_2 =1+2=3$，$s_3 =1+2+3=6$，… $s_m =1+2+3+...+m=m(m+1)/2$,正解答案中给出，m(m+1)/2+k=n (k起修正作用的常数)，也可大致口算 $m\approx\sqrt{n}$,则时间复杂度为O($\sqrt{n}$)

2、设n为如下程序段处理的数据个数，求时间复杂度

```
for(i=1;i<n;i=2*i)
   std::cout<<"i="<<i<<std::endl;
```

分析：

主看for循环，当i>=n时结束，假设执行m次结束，$i_1 =2= 2^1$, $i_2 =2*2 = 2^2$, ...，$i_m = 2^m$,则有 $2^m =n$,大致口算m= $log_2 n$ ，则时间复杂度为O($log_2 n$)

3、计算n！的递归函数如下，分析时间复杂度

```
int func(int n)
{
   if(n<=1)
      return 1;   //①
   else
      return n*func(n-1);   //②
}
```

分析：

n！递归函数中，①的时间复杂度显然O(1)，应主要分析else后的语句②，递归调用func(n-1)的时间开销为T(n-1)，则②时间开销就是O(1)+T(n-1)。

假设求1!就是O(1)+T(n-1)=1×O(1)+T(n-1)【n=1】 ,2!就是O(1)+O(1)+T(n-2)=2×O(1)+T(n-2)【n=2】... ，n！=(n-1)×O(1)+T(n-(n-1)) =(n-1)×O(1)+T(1) = n×O(1)=O(n) ，所以时间复杂度为O(n)

4、设A是一个线性表（a_1 ....a_n）采用顺序存储结构，则在等概率的前提下，平均插入一个元素需要移动的元素个数是多少？若元素插入在a_i(1≤i≤n)所在位置处的概率为 n-i/n(n-1)/2 ,则平均插入一个元素要移动的元素个数是多少？

分析：

（1）：在a_1插入则要移动n次，a_2插入移动n-1次，...，a_n插入移动0次，总次数为0+1+2+...+n=n(n+1)/2 ,总共是0到n共1+n个 。则等概率下，平均插入一个元素移动的元素个数为[n(n+1)/2]/[1+n]=n/2

（2）：将插入在a_i处插入概率用P表示，不难发现a_i处插入一个元素则需移动元素为n-i+1 ，可以以 {1,2,3,4,5}为例，在2处插入，5-2+1=4，故2,3,4,5四个元素往后移一位。则P概率下，平均插入一个元素移动的元素个数为 $\sum P(n-i+1)$ = (2n+2)/3

5、设计一个算法，用不多于3n/2的平均比较次数，在数组A[0,...,n-1]中找出最大值和最小值的元素

```
//如果找最大值时遍历一次，最小值时遍历一次，则需要比较2n，所以尽量就遍历一次
void MaxandMin(int A[],int n,int &max,int &min)
{
   max=min=A[0];
   for(int i=0;i<n;i++)
   {
      if(A[i]>max) max=A[i];
      if(A[i]<min) min=A[i];
   }
}
```

分析：

最坏情况是A中递减次序排列，A[i]>max均不成立，比较n-1次，同样A[i]<min同样比较n-1次 ，总次数2(n-1)次，最好情况是A中递增次序排列，A[i]>max均成立，不执行A[i]<min，总次数n-1 ，所以平均比较次数为[2(n-1)+(n-1)]/2 = 3n/2-3/2 ，故符合题目条件。

6、分析以下程序时间复杂度

```
void fun()
{
  int i=1,k=0,n=10;
  while(i<=n-1)
  {
     k+=10*i;
     ++i;
  }
}
```

分析：

显然看出可以改写为for(i=1;i<=n-1;++i) ，故时间复杂度为O(n)

```
void fun(int n)
{
   int i=1,k=0;
   do
   {
    k+=10*i;
       ++i;
   }while(i==n)
}
```

分析：

显然i!=n时跳出循环，如：n=100，i=1，仅循环一次，故时间复杂度O(1)

```
void fun(n)
{
   int i=1,j=0;
   while(i+j<=n)
      if(i>j)
         ++j;
```

```
    else
        ++i;
}
```

分析:

可以写几个例子，1<=n i=1,j=1; 2<=n i=2,j=1 ;3<=n i=2,j=2 ... 显然时间复杂度O(n)

```
void fun(int n)
{
   int x=n,y=0;
   while(x>=(y+1)*(y+1))
      ++y;
}
```

分析:

同样写几个例子，x>=1×1 y=1 ; x>=2×2 y=2 ... 显然时间复杂度O($\sqrt{n}$)

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

分析:

同样可以写例子，显然时间复杂度为O($log_2 n$)

```
void fun(n)
{
   i=1;
   while(i<=n)
      i=i*3
}
```

显然时间复杂度为O($log_3 n$)

7、假设n为2的乘幂，求以下时间复杂度

```
void counter()
{
   int n,x,count;
   std::cout<<"n:";
   std::cin>>n;
   count=0;
   x=2;
   while(x<n/2)
   {
      x=2*x;
      ++count;
   }
   std::cout<<count<<std::endl;
}
```

分析:

循环处x<n/2 ,所以对x进行观察, $x= 2^2$, 2^3 ... 显然时间复杂度为O($log_2 n$)

8、某算法所需时间由以下方程表示，求出该算法时间复杂度（大O形式表达） 注意：n为求解问题规模，为简单起见，设n为2的正整数幂

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T\left(\frac{n}{2}\right) + n & n > 1 \end{cases}$$

分析:

设n= 2^m 则,

T(n)=T(2^m)=2T($2^m - 1$)+ 2^m

=2(2T($2^m - 2$)+ 2^)+ 2^m = 2^2 ×T(2^{m-2})+2× 2^m

...

= 2^m ×T(1)+m× 2^m

= (m+1) 2^m

= ($log_2 n$ +1)n

=O(n $log_2 n$)

9、分析sort函数时间复杂度

```
void sort(int j,int n)
{
   int i,temp;
   if(j<n)
   {
      for(i=j;i<=n;++i)
         if(a[i]<a[j])
            swap(a[i],a[j]);   //本函数时间复杂度O(1)
       ++j;
      sort(j,n)   //递归调用
   }
}
```

分析:

sort是一个递归排序过程，这里假设T(n) 是排序n各元素要的时间。纵观代码，主要花费时间在递归调用sort()上。如果第一次调用，处理元素个数n-1，即对剩下n-1个元素进行排序，所需时间就是T(n-1) 。又sort()在for循环中，就需要n-1次比较。

列出方程:

T(1)=0 ,n=1

T(n)=T(n-1)+n-1 ,n>1

求解:

T(n)=[T(n-2)+(n-2)]+(n-1)

=[T(n-3)+(n-3)]+(n-2)+(n-1)

...

=(T(1)+1)+2+3+...+n-1

=0+1+2+...+n-1

=n(n-1)/2

=O(n²)

10、设计下列问题算法，分析其最坏情况的时间复杂度

(1)在数组A[0,...,n-1]中查找值为k的元素，若找到，则输出其位置i(i为数组下标)；否则输出-1为坐标

```
int findK(int A[],int k)  //这里假设A中元素都是int型
{
   int i =0;
   while(i<n&&A[i]!=k)
       i++;
   return i;  //找到了
   else
       return -1;
}
```

最坏情况：就是遍历一遍后查不到k，比较了n+1次，故时间复杂度O(n)

(2)在数组A[0,...,n-1]中找出元素的最大值和次最大值

```
void mxa(int A[],int M, int m)   //M为最大值，m为次大值
{
   int i;
   M=m=MIN;  //设MIN为已定义常量，比A[]中所有元素都小
   for(i=0;i<n;i++)
   { //find M
    if(A[i]>M) M=A[i];
   }
   for(i=0;i<n;i++)
   { //find m
      if(A[i]!=M&&A[i]>m) m=A[i];
   }
}
```

最坏情况：各来一次遍历，并都是最后找到，一共比较了2n-2次，故时间复杂度O(n)

——— EOF ———