绪论

重点

时间复杂度和空间复杂度的计算

时间复杂度

常数阶$O(1)$

```
int i=1;
int j=1;
++i;
j++;
int m= i+j;
```

对数阶$O(logN)$

2^n次执行后退出循环

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

线性阶$0(n)$

```
for(int i=1;i<=n;++i)
{
        j=i;
        j++;
}
```

线性对数阶$O(NlogN)$

将对数阶循环N遍

```
for(k=1;k<n;k++)
{
    i=1;
    while(i<n)
    {
        i=i*2;
    }
}
```

$O(m * n)$

里层M次,外层N次

$O(n^k)$

n层嵌套循环

空间复杂度

$O(1)$

变量分配空间不变化

```
int i=1;
int j=1;
++i;
j++;
int m= i+j;
```

$O(n)$

```
int[] m = new int[n];

for (i = 1; i <= n; ++i)

{

  j = i;

  j++;
```

```
}
```

易错题

14.【2017 统考真题】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;
}
```

A. $O(\log n)$　　B. $O(n^{1/2})$　　C. $O(n)$　　D. $O(n\log n)$

16.【2022 统考真题】下列程序段的时间复杂度是（ ）。

```
int sum=0;
for(int i=1;i<n;i*=2)
    for(int j=0;j<i;j++)
        sum++;
```

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

归纳总结

归纳总结

本章的重点是分析程序的时间复杂度。一定要掌握分析时间复杂度的方法和步骤，很多读者在做题时一眼就能看出程序的时间复杂度，但就是无法规范地表述其推导过程。为此，编者查阅众多资料，总结出了此类题型的两种形式，供大家参考。

1．循环主体中的变量参与循环条件的判断

在用于递推实现的算法中，首先找出基本运算的执行次数 x 与问题规模 n 之间的关系式，解得 $x=f(n)$，$f(n)$的最高次幂为 k，则算法的时间复杂度为 $O(n^k)$。例如，

```
1. int i=1;
   while(i<=n)
       i=i*2;
```

```
2. int y=5;
   while((y+1)*(y+1)<n)
       y=y+1;
```

在例 1 中，设基本运算 i=i*2 的执行次数为 t，则 $2^t \leq n$，解得 $t \leq \log_2 n$，故 $T(n)=O(\log_2 n)$。

在例 2 中，设基本运算 y=y+1 的执行次数为 t，则 $t=y-5$，且$(t+5+1)(t+5+1)<n$，解得 $t<\sqrt{n}-6$，即 $T(n)=O(\sqrt{n})$。

2．循环主体中的变量与循环条件无关

此类题可采用数学归纳法或直接累计循环次数。多层循环时从内到外分析，忽略单步语句、条件判断语句，只关注主体语句的执行次数。此类问题又可分为递归程序和非递归程序：

- 递归程序一般使用公式进行递推。时间复杂度的分析如下：

$$T(n)=1+T(n-1)=1+1+T(n-2)=\cdots=n-1+T(1)$$

即 $T(n)=O(n)$。

- 非递归程序的分析比较简单，可以直接累计次数。本节前面给出了相关的习题。