PTA 循环结构 E

rogeryoungh

2021 年 04 月 13 日

目录

PTA 循环结构 EASY 部分，PDF。

1 约定

随着程序逐渐复杂，我决定介绍一些算法竞赛中常用的定义，来简化我们的程序。

```
typedef long long ll;
#define _fora(i,a,n) for(int i=(a);i<=(n);i++)
#define _forz(i,a,n) for(int i=(a);i>=(n);i--)
```

即 `ll` 是 `long long` 类型的简写，能够带来比 `int` 更大的范围。
同样，`_fora` 和 `_forz` 是 `for` 的简写。
我尽量避免在讲解中使用它们，感兴趣的可以尝试。

2 时间复杂度

可能已经有同学察觉到，尽管很多写法都可以通过测试，可它们并不是一样快的。
比如，求

$$\sum_{i=1}^{n}\sum_{j=1}^{i} j = \sum_{i=1}^{n} \frac{i(i+1)}{2} = \frac{1}{6}n(n+1)(n+2)$$

我们可以写出四种写法

```
int sum0(int n) { // 不会真有人这么写吧
    int sum = 0;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= i; j++)
```

```
            for (int k = 1; k <= j; k++)
                sum++;
    return sum;
}

int sum1(int n) {
    int sum = 0;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= i; j++)
            sum += j;
    return sum;
}

int sum2(int n) {
    int sum = 0;
    for (int i = 1; i <= n; i++)
        sum += i * (i + 1) / 2;
    return sum;
}

int sum3(int n) {
    return n * (n + 1) * (n + 2) / 6;
}
```

可以得出，sum0 累加执行的次数是 $\frac{1}{6}(n^3+3n^2+2n)$，sum1 累加执行的次数是 $\frac{1}{2}(n^2+n)$，而 sum2 需要 n 次运算，sum3 只需要 1 次。

尽管电脑的运行速度很快，可并不是无限快，不同的算法所需时间差别可能会很大。

当 n 较大时，比如 $n=10000$，sum0 需要接近 10^{12} 次，sum1 大致 10^8 次，sum2 大致 10^4，而 sum3 还是 1 次，差距非常明显。

为了凸显算法的运行时间和 n 的关系，在表示算法的时间复杂度时常常略去常数和低阶项，系数也会被忽略。比如

$$\frac{1}{6}(n^3+3n^2+2n) \sim \frac{1}{6}n^3 \sim n^3$$

我们可将上述四种算法的时间复杂度记为 $O(n^3)$，$O(n^2)$，$O(n)$ 和 $O(1)$。时间复杂度不一定是全是多项式，还可能是 $O(2^n)$，$O(\log n)$，$O(n\log n)$ 等等。

在着手编写之前先估算算法的复杂度，一些显然过不去的算法就没必要写了，一般把计算机的执行速度定为 10^9 次，我们可以列出算法的最大规模

运算量	n	$n!$	2^n	n^2	n^3	$n\log_2 n$
数据范围	10^9	13	29	31622	1000	3.9×10^7
二倍速度	2×10^9	13	30	44721	1259	7.6×10^7

时间复杂度本身是很复杂的概念，有兴趣的可以查阅资料。因为时间很容易测不准，不能简单的以运行时间评判程序的优劣，尤其运行时间极其短时；同样也不能以上界分析的结果简单的断定程序的速度。在不少情况下，算法实际能解决的问题规模与上表有着较大差异。

尽管如此，此表还是有一定借鉴意义的，比如一个指明 $n \leqslant 20$ 的题目可能 2^n 的算法已经足够，而 $n \leqslant 10^4$ 的题目可能需要 n^2 的算法，$n \leqslant 10^6$ 则可能至少要 $n \log n$ 的算法了。

相对于充斥着各种奇妙优化的算法，朴素的算法常称为暴力。

我们应尽量思考，尝试发现更优的算法，以更佳的方法解决问题。

3 简单循环

大部分题还是常规的。