# 一、时间复杂度

1.

$f_1(n) = log(log(n))$ -- $O(log(log(n)))$
$f_2(n) = 2^{log(n)}$ -- $O(n)$
$f_3(n) = n^8$ -- $O(n^8)$
$f_4(n) = n^8 + n!$ -- $O(n!)$
$f_5(n) = n^{16} + nlog(n)$ -- $O(n^{16})$
$f_6(n) = e^n + e^{16}$ -- $O(e^n)$

从高到低：$f_4 > f_6 > f_5 > f_3 > f_2 > f_1$

2.

$O(sqrt(n))$，j 每次加1，j 方超过 n 时跳出循环。

3.

$O(n^2 * log(n))$，内层 for 循环执行 n 方次，外层执行 $log(n^2)$ 次，即 $2log(n)$ 次。

4.

$O(log_3(n))$，判断是否存在整数 k，使得 $n = 3^k$。

# 二、OnlineJudge系统与测试

1.没有通过全部测试点。
2.使用 freopen() 函数。

# 三、实验 E1

## G.Rally

**题目大意**

找一个整数 p 使得 $\sum_1^n (x_i - p)^2$ 最小。

**解法**

数据较小，直接暴力。

**时间复杂度**

$O(100 * n)$

### 代码

```
int main (){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;
    vector<int> a(n);
    ll ans = 1e6;

    for (int i = 0; i < n; i++) cin >> a[i];

    for (int i = 1; i <= 100; i++) {
        ll b = 0;
        for (int j = 0; j < n; j++) {
            b += (a[j] - i) * (a[j] - i);
        }
        ans = min(ans, b);
    }
    cout << ans << '\n';

    return 0;
}
```

## I.Kaprekar Number

### 题目大意

定义运算。

### 解法

根据定义，模拟即可。

### 时间复杂度

$O(k)$

### 代码

```
ll f(ll n) {
    ll g1 = 0, g2 = 0;
    vector<int> a(10, 0);

    while (n) {
        a[n % 10]++;
        n /= 10;
    }
    for (int i = 9; i >= 0; i--) {
        for (int j = 0; j < a[i]; j++) {
            g1 *= 10;
            g1 += i;
        }
    }
    for (int i = 1; i < 10; i++) {
        for (int j = 0; j < a[i]; j++) {
            g2 *= 10;
            g2 += i;
        }
    }
    return g1 - g2;
}

int main (){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    ll n, k;
    cin >> n >> k;

    for (int i = 0; i < k; i++) {
        n = f(n);
    }
    cout << n << "\n";
    return 0;
}
```