



# 程序设计与算法(一)

## C语言程序设计

郭 炜

微博：<http://weibo.com/guoweiofpku>

**学会程序和算法，走遍天下都不怕！**

讲义照片均为郭炜拍摄



北京大学  
PEKING UNIVERSITY

信息科学技术学院

指定教材：

# 《新标准C++程序设计教程》

郭炜 编著

清华大学出版社

重点大学计算机专业系列教材

## 新标准C++程序设计教程

郭炜 编著



清华大学出版社



# 二分算法



北京大学  
PEKING UNIVERSITY

信息科学技术学院

## 程序或算法的 时间复杂度



美国加州1号公路

# 程序或算法的时间复杂度

- 一个程序或算法的时间效率，也称“时间复杂度”，有时简称“复杂度”

# 程序或算法的时间复杂度

- 一个程序或算法的时间效率，也称“时间复杂度”，有时简称“复杂度”
- 复杂度常用大的字母O和小写字母n来表示，比如 $O(n)$ ,  $O(n^2)$ 等。n代表问题的规模

# 程序或算法的时间复杂度

- 一个程序或算法的时间效率，也称“时间复杂度”，有时简称“复杂度”
- 复杂度常用大的字母O和小写字母n来表示，比如 $O(n)$ ,  $O(n^2)$ 等。n代表问题的规模
- 时间复杂度是用算法运行过程中，某种时间固定的操作需要被执行的次数和n的关系来度量的。在无序数列中查找某个数，复杂度是 $O(n)$

# 程序或算法的时间复杂度

- 一个程序或算法的时间效率，也称“时间复杂度”，有时简称“复杂度”
- 复杂度常用大的字母O和小写字母n来表示，比如 $O(n)$ ,  $O(n^2)$ 等。n代表问题的规模
- 时间复杂度是用算法运行过程中，某种时间固定的操作需要被执行的次数和n的关系来度量的。在无序数列中查找某个数，复杂度是 $O(n)$
- 计算复杂度的时候，只统计执行次数最多的(n足够大时)那种固定操作的次数。比如某个算法需要执行加法 $n^2$ 次，除法n次，那么就记其复杂度是 $O(n^2)$ 的。



# 插入排序

```
void InsertionSort(int a[] ,int size)
{
    for(int i = 1;i < size; ++i ) {
        //a[i]是最左的无序元素，每次循环将a[i]放到合适位置
        for(int j = 0; j < i; ++j)
            if( a[j]>a[i]) {
                //要把a[i]放到位置j，原下标j到 i-1的元素都往后移一个位子
                int tmp = a[i];
                for(int k = i; k > j; --k)
                    a[k] = a[k-1];
                a[j] = tmp;
                break;
            }
    }
} //复杂度O(n2)
```

# 程序或算法的时间复杂度

- 如果复杂度是多个n的函数之和，则只关心随n的增长增长得最快的那个函数

$$O(n^3+n^2) \Rightarrow O(n^3)$$

$$O(2^n+n^3) \Rightarrow O(2^n)$$

$$O(n! + 3^n) \Rightarrow O(n!)$$

# 程序或算法的时间复杂度

- 如果复杂度是多个n的函数之和，则只关心随n的增长增长得最快的那个函数

$$O(n^3+n^2) \Rightarrow O(n^3)$$

$$O(2^n+n^3) \Rightarrow O(2^n)$$

$$O(n! + 3^n) \Rightarrow O(n!)$$

- 常数复杂度： $O(1)$       时间(操作次数)和问题的规模无关
- 对数复杂度： $O(\log(n))$
- 线性复杂度： $O(n)$
- 多项式复杂度： $O(n^k)$
- 指数复杂度： $O(a^n)$
- 阶乘复杂度： $O(n!)$

# 程序或算法的时间复杂度

- 复杂度有“平均复杂度”和“最坏复杂度”两种。  
两者可能一致，也可能不一致

# 程序或算法的时间复杂度

- 在无序数列中查找某个数(顺序查找)  $O(n)$
- 平面上有 $n$ 个点，要求出任意两点之间的距离  $O(n^2)$
- 插入排序、选择排序、冒泡排序  $O(n^2)$
- 快速排序  $O(n \cdot \log(n))$
- 二分查找  $O(\log(n))$



北京大学  
PEKING UNIVERSITY

信息科学技术学院

## 二分查找



甘肃张掖平山湖大峡谷

# 二分查找

●A心里想一个1-1000之间的数，B来猜，可以问问题，A只能回答是或否。怎么猜才能问的问题次数最少？

是1吗？是2吗？.....是999吗？ 平均要问500次

大于500吗？大于750吗？大于625吗？ .....每次缩小猜测范围到上次的一半，只需要 10次

## 二分查找函数

- 写一个函数BinarySearch，在包含size个元素的、从小到大排序的int数组a里查找元素p,如果找到，则返回元素下标，如果找不到，则返回-1。要求复杂度 $O(\log(n))$

```
int BinarySearch(int a[],int size,int p)
{
    int L = 0; //查找区间的左端点
    int R = size - 1; //查找区间的右端点
    while( L <= R) { //如果查找区间不为空就继续查找
        int mid = L+(R-L)/2; //取查找区间正中元素的下标
        if( p == a[mid] )
            return mid;
        else if( p > a[mid])
            L = mid + 1; //设置新的查找区间的左端点
        else
            R = mid - 1; //设置新的查找区间的右端点
    }
    return -1;
} //复杂度 $O(\log(n))$ 
```



## 二分查找函数

●写一个函数LowerBound，在包含size个元素的、从小到大排序的int数组a里查找比给定整数p小的，下标最大的元素。找到则返回其下标，找不到则返回-1

```
int LowerBound(int a[],int size,int p)  //复杂度 $O(\log(n))$ 
{
    int L = 0; //查找区间的左端点
    int R = size - 1; //查找区间的右端点
    int lastPos = -1; //到目前为止找到的最优解
    while( L <= R) { //如果查找区间不为空就继续查找
        int mid = L+(R-L)/2; //取查找区间正中元素的下标
        if(a[mid]>= p)
            R = mid - 1;
        else {
            lastPos = mid;
            L = mid+1;
        }
    }
    return lastPos;
}
```

# 二分查找函数

- 注意：

```
int mid = (L+R) / 2; //取查找区间正中元素的下标
```

- 为了防止 (L+R)过大溢出:

```
int mid = L + (R-L) / 2;
```



北京大学  
PEKING UNIVERSITY

信息科学技术学院

## 二分法求方程的根



祁连山风光

## 二分法求方程的根

求下面方程的一个根： $f(x) = x^3 - 5x^2 + 10x - 80 = 0$

若求出的根是 $a$ ，则要求  $|f(a)| \leq 10^{-6}$

●解法：对 $f(x)$ 求导，得 $f'(x) = 3x^2 - 10x + 10$ 。由一元二次方程求根公式知方程 $f'(x) = 0$ 无解，因此 $f'(x)$ 恒大于0。故 $f(x)$ 是单调递增的。易知 $f(0) < 0$ 且 $f(100) > 0$ ，所以区间 $[0, 100]$ 内必然有且只有一个根。由于 $f(x)$ 在 $[0, 100]$ 内是单调的，所以可以用二分的办法在区间 $[0, 100]$ 中寻找根。

# 二分法求方程的根

```
#include <stdio>
#include <iostream>
#include <cmath>
using namespace std;
```

```
double EPS = 1e-6;
double f(double x) { return x*x*x - 5*x*x + 10*x - 80; }
int main() {
    double root, x1 = 0, x2 = 100, y;
    root = x1+(x2-x1)/2;
    int triedTimes = 1; //记录一共尝试多少次，对求根来说不是必须的
    y = f(root);
    while( fabs(y) > EPS) {
        if( y > 0 )    x2 = root;
        else          x1 = root;
        root = x1+(x2 - x1)/2;
        y = f(root);
        triedTimes ++;
    }
    printf("%.8f\n", root);
    printf("%d", triedTimes);
    return 0;
}
```

5.70508593  
32



北京大学  
PEKING UNIVERSITY

信息科学技术学院

## 例题 寻找指定和的整数对



青海湖

## 例题 寻找指定和的整数对

输入 $n$  ( $n \leq 100,000$ )个整数，找出其中的两个数，它们之和等于整数 $m$ (假定肯定有解)。题中所有整数都能用 `int` 表示



## 例题：寻找指定和的整数对

输入 $n$  ( $n \leq 100,000$ )个整数，找出其中的两个数，它们之和等于整数 $m$ (假定肯定有解)。题中所有整数都能用 `int` 表示

解法1：用两重循环，枚举所有的取数方法，复杂度是 $O(n^2)$ 的。

```
for(int i = 0; i < n-1; ++i)
    for(int j = i + 1; j < n; ++j)
        if( a[i]+a[j] == m)
            break;
```

$100,000^2 = 100$ 亿，在各种OJ上提交或参加各种程序设计竞赛，这样的复杂度都会超时！



## 例题：寻找指定和的整数对

输入 $n$  ( $n \leq 100,000$ )个整数，找出其中的两个数，它们之和等于整数 $m$ (假定肯定有解)。题中所有整数都能用 `int` 表示

解法2：

- 1) 将数组排序，复杂度是 $O(n \times \log(n))$
- 2) 对数组中的每个元素 $a[i]$ ，在数组中二分查找 $m - a[i]$ ，看能否找到。复杂度 $\log(n)$ ，最坏要查找 $n-2$ 次，所以查找这部分的复杂度也是 $O(n \times \log(n))$

这种解法总的复杂度是 $O(n \times \log(n))$ 的。

## 例题：寻找指定和的整数对

输入 $n$  ( $n \leq 100,000$ )个整数，找出其中的两个数，它们之和等于整数 $m$ (假定肯定有解)。题中所有整数都能用 `int` 表示

解法3：

- 1) 将数组排序，复杂度是 $O(n \times \log(n))$
- 2) 查找的时候，设置两个变量 $i$ 和 $j$ ,  $i$ 初值是0,  $j$ 初值是 $n-1$ . 看 $a[i] + a[j]$ , 如果大于 $m$ ，就让 $j$ 减1，如果小于 $m$ , 就让 $i$ 加1，直至 $a[i] + a[j] = m$ 。

这种解法总的复杂度是 $O(n \times \log(n))$ 的。



北京大学  
PEKING UNIVERSITY

信息科学技术学院

# 例题 Aggressive cows



航拍青海湖

## 例题2 百练 2456: Aggressive cows

<http://bailian.openjudge.cn/practice/2456>

农夫 John 建造了一座很长的畜栏，它包括  $N$  ( $2 \leq N \leq 100,000$ ) 个隔间，这些小隔间的位置为  $x_0, \dots, x_{N-1}$  ( $0 \leq x_i \leq 1,000,000,000$ , 均为整数, 各不相同)。

John 的  $C$  ( $2 \leq C \leq N$ ) 头牛每头分到一个隔间。牛都希望互相离得远点省得互相打扰。怎样才能使任意两头牛之间的最小距离尽可能的大，这个最大的最小距离是多少呢？

## 例题2

### ●解法1：

先得到排序后的隔间坐标  $x_0, \dots, x_{N-1}$

从1,000,000,000到1依次尝试这个“最大的最近距离”  $D$ ，找到的第一个可行的就是答案。

尝试方法：

- 1) 第1头牛放在 $x_0$
- 2) 若第 $k$ 头牛放在 $x_i$ ，则找到 $x_{i+1}$ 到 $x_{N-1}$ 中第一个位于 $[x_i + D, 1,000,000,000]$ 中的 $x_j$ ，第 $k+1$ 头牛放在 $x_j$ 。找不到这样的 $x_j$ ，则  $D = D - 1$ ，转 1)再试

若所有牛都能放下，则 $D$ 即答案

## 例题2

### ●解法1：

先得到排序后的隔间坐标  $x_0, \dots, x_{N-1}$

从  $1,000,000,000/C$  到 1 依次尝试这个“最大的最近距离”  $D$ ，找到的第一个可行的就是答案。

尝试方法：

- 1) 第1头牛放在  $x_0$
- 2) 若第  $k$  头牛放在  $x_i$ ，则找到  $x_{i+1}$  到  $x_{N-1}$  中第一个位于  $[x_i + D, 1,000,000,000]$  中的  $x_j$ ，第  $k+1$  头牛放在  $x_j$ 。找不到这样的  $x_j$ ，则  $D = D - 1$ ，转 1) 再试

若所有牛都能放下，则  $D$  即答案

复杂度  $1,000,000,000/C * N$ ，即  $1,000,000,000$ ，超时！

## 例题2

### ●解法2：

先得到排序后的隔间坐标  $x_0, \dots, x_{N-1}$

在 $[L, R]$ 内用二分法尝试“最大最近距离”  $D = (L+R)/2$  ( $L, R$ 初值为 $[1, 1,000,000,000/C]$ )

若 $D$ 可行，则记住该 $D$ ，然后在新 $[L, R]$ 中继续尝试( $L = D + 1$ )

若 $D$ 不可行，则在新 $[L, R]$ 中继续尝试( $R = D - 1$ )

复杂度  $\log(1,000,000,000/C) * N$