1. 素数的判断

从素数的定义中可以知道，一个整数要被判断为素数，需要判断n是否能被2,3,,,n-1中的一个整除。只有2, 3, …, n-1都不能整除n，n才能判定为素数，只要有一个能整除n的数出现，n就可以判定为非素数。

1. 常规判断解法
2. 遍历所有值

了解了如何判断素数，只要从2开始一直循环到n-1，如果有一个数能整除n-1，则n不是素数。该算法的复杂度为o(n)。

//全遍历

bool isPrime(int n) {

if (n <= 1)

return false;

for (int i = 2; i <= n - 1; ++i) {

if (n % i == 0)

return false;

}

return true;

}

1. 遍历一部分

如果n不是素数，则在2~n-1中存在n的约数，设该约数为k，则n % k ==0,。那么 可以得到k \* (n / k) = n，则 n / k也是n的一个约数。因此k 和 n / k这两个n的约数定满足一个小于 sqrt(n)，一个大于sqrt(n)。因此，我们只需要判断n能否被 2, 3, …, sqrt(n)中的一个数整除即可。该算法复杂度为o(sqrt(n))。

//遍历一部分

bool isPrime(int n) {

if (n <= 1)

return false;

int sqr = (int)sqrt(1.0 \* n);

for (int i = 2; i <= sqr; ++i) {

if (n % i == 0)

return false;

}

return true;

}

1. 筛选素数解法

从2到n-1，枚举所有的数，对每个素数，筛掉它的所有倍数，那么留下来的就都是素数了。这句话听起来有点歧义，一开始并不知道素数是谁，怎么筛？

就从2开始,2是素数，筛掉所有2的倍数；然后是3，3没有被前面筛掉，则3也是素数，筛掉所有3的倍数…

原理是：当用这种方式筛到数a时，如果a没有被前面步骤筛掉，则说明a没有小于a的素因子，则a定为素数。

//筛选法

const int maxn = 101; //表长

int prime[maxn], pNum = 0; //素数表，表长

bool p[maxn] = { 0 }; //如果i为素数，则p[i]为false，初始化为false

void Find\_Prime() {

for (int i = 2; i < maxn; ++i) {

//从2开始，如果i是素数

if (p[i] == false) {

//添加i到素数表中

prime[pNum++] = i;

for (int j = i + 1; j < maxn; j += i) {

//筛掉所有i的倍数

p[j] = true;

}

}

}

}