如果一个算法用常数时间（）将问题的大小削减为其一部分（），则该算法就是。

典型的对数算法：

1. 二分搜索：给定整数和整数，后者已排序，求下标使得，如果不在序列之，返回。

#define NOT\_FOUND -1

template<class T>

int BinarySearch(const vector<T>& array, const T& X)

{

int low = 0;

int high = array.size() - 1;

while (low <= high)

{

int mid = (low + high) / 2;

if (array[mid] < X)

{

low = mid + 1;

}else if (array[mid] > X)

{

high = mid - 1;

}

else

{

return mid;

}

}

return NOT\_FOUND;

}

二分搜索的时间复杂度为，提供了在时间内的contains操作。在数据稳定的环境中，非常有用，将数据进行一次排序后，之后的访问就会很快。典型的例子为化学的元素周期表，表里的数据非常稳定，大于有110种，找到一个元素，最多需要存取8次。

1. 欧几里得算法gcd：计算两个数的最大公因数

// 求两个数的最大公因数

// 假设m>=n

// 如果n>m，第一次迭代将交换两个数

long gcd(long m, long n)

{

while (n != 0)

{

long rem = m % n;

m = n;

n = rem;

}

return m;

}

定理：如果，则

所以，欧几里得算法的时间复杂度为.

1. 幂运算

bool IsEven(long x)

{

if (x % 2 == 0)

{

return true;

}

return false;

}

long pow(long x, int n)

{

if (n == 0)

{

return 1;

}

if (n == 1)

{

return x;

}

if (IsEven(x))

{

return pow(x \* x, n / 2);

}

else

{

return pow(x \* x, n / 2) \* x;

}

如果为偶数，

如果为奇数，

幂运算的时间复杂度为。