# 求GCD

## 辗转相除法

著名的辗转相除法求两个数的GCD是被我们所熟知的，其思路是：

对于任意两个正整数a，b 假设a>b

设r = a%b

若r = 0

则GCD=b

若r!=0

则令a=b ， b=r 并重复上述操作

直到r=0 便可得到GCD

### 为了便于操作，我们用一个实例演示：

求18和30的GCD

a=30，b=18

r=a%b=12 不为0

则a=18 b=12

r = a%b = 6 不为0

则a=12 b = 6

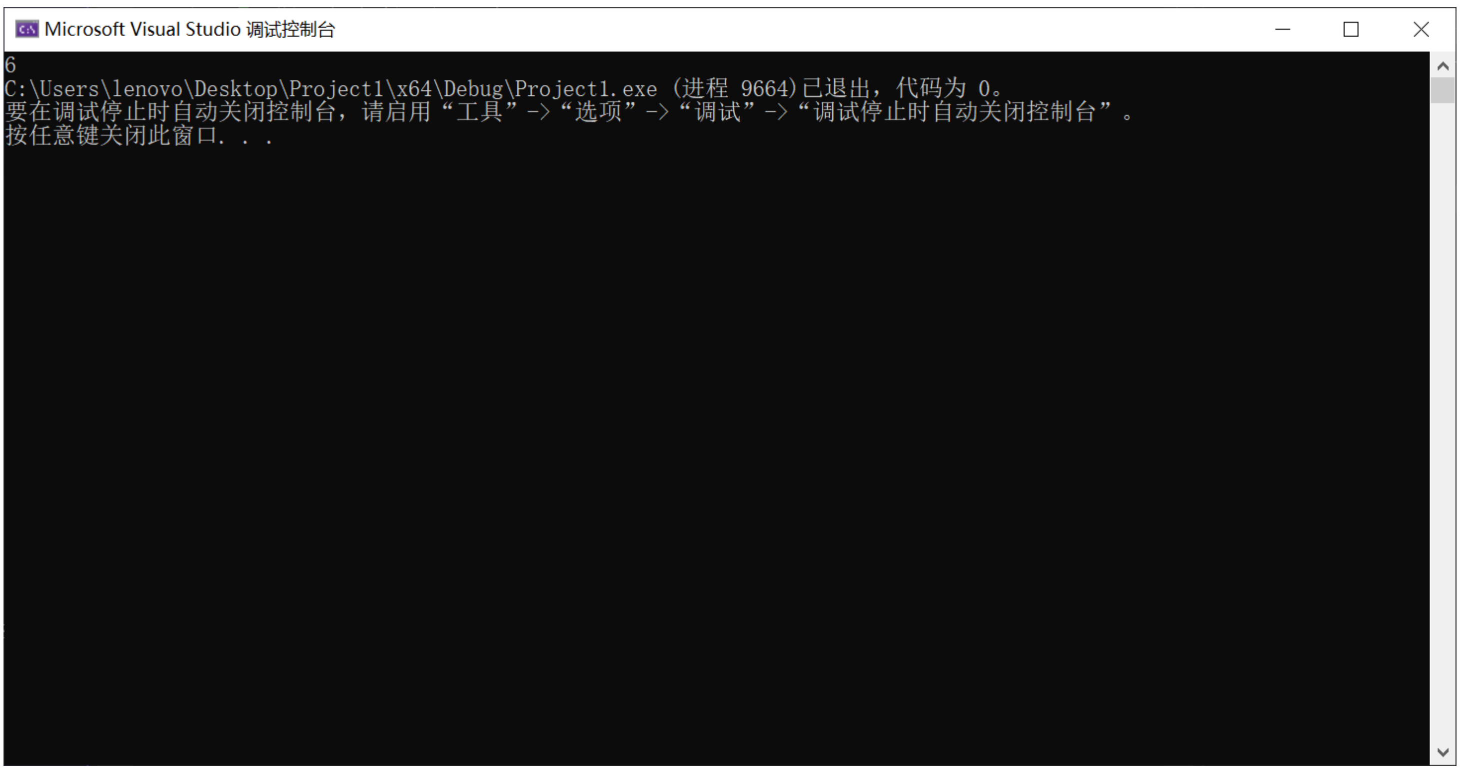
r = a%b = 0

所以GCD为6

### 代码演示：

```
#include<iostream>
using namespace std;
int GCD(int a,int b) {
	int r;
	while (b)
	{
		r = a % b;
		a = b;
		b = r;
	}
	return a;
}
void main() {
	cout << GCD(30, 18);
}
```

## 结果展示




## 算法分析：

正如课上同学所讲

%的运算速度远大于 + -

所以我们可以认为减小我们的运算量

- 如果 a，b都是偶数

肯定都有2为因子，那么很容易想到

G(a,b)=2G(a/2，b/2)

比如 G(2，8) = 2G(1，4)

- a，b 一个为奇数，一个为偶数

那么肯定没有2因子，所以对偶数除以2参与运算并不会改变最后的结果

- a，b都是奇数，a-b一定是偶数。设 d 是 a 和 b 的一个公约数，那么 d 也是 a-b 的约数。因为如果 d 能够整除 a 和 b，那么它也能够整除它们的差。设 d 是 a-b 和 b 的一个公约数，那么 d 也是 a 的约数。因为如果 d 能够整除 a-b 和 b，那么它也能够整除它们的和。所以G(a,b) = G(a,a-b)

# 更相减损法

更相减损法求GCD的思路如下：

对于任意两个正整数 a，b 假设a>b

设r = a-b

若r = 0

则GCD = a

若r!=0

则a = max(b,r)

b = min(b,r)

再重复上述操作即可以得到GCD

举例说明：

求30，18 的GCD

a = 30，  b = 18

r = a - b = 12 不为零

a = 18 b = 12

r = 6 不为零

a = 12 b = 6

r = 6 不为零

a =6 b = 6

r = 0

GCD = 6

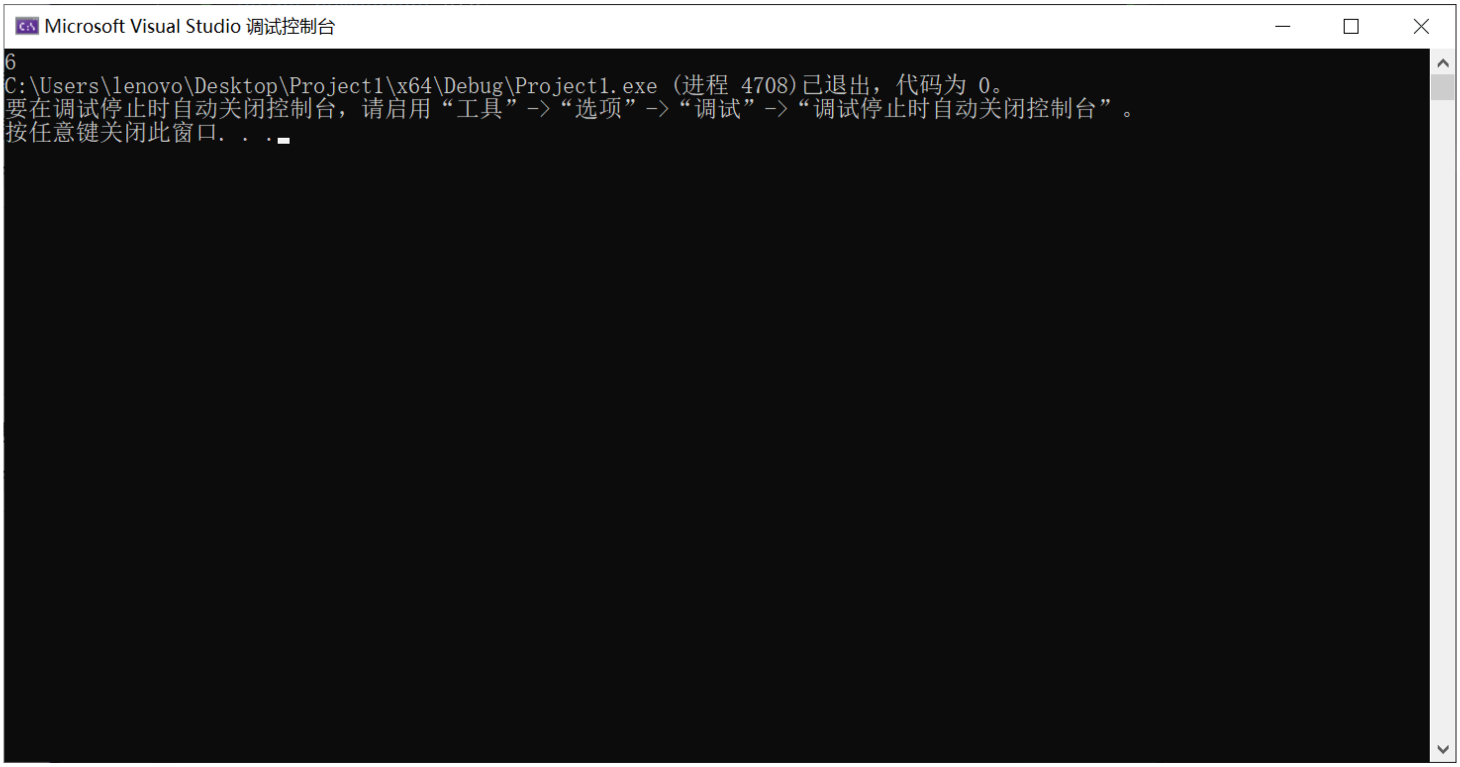
实际上上述操作可以写成递归形式

## 代码演示：

```
#include<iostream>
using namespace std;
int GCD(int a, int b) {

    if (a == b)
        return a;
    else {
        int big = max(a, b);
        int small = min(a, b);
        return GCD(big - small, small);
    }
}
void main() {
        cout << GCD(30, 18);
}
```

## 结果展示




### 算法分析：

我个人感觉更相减损法和辗转相除法几乎一致

不过一个是除法一个是减法

在时间复杂度上来讲：

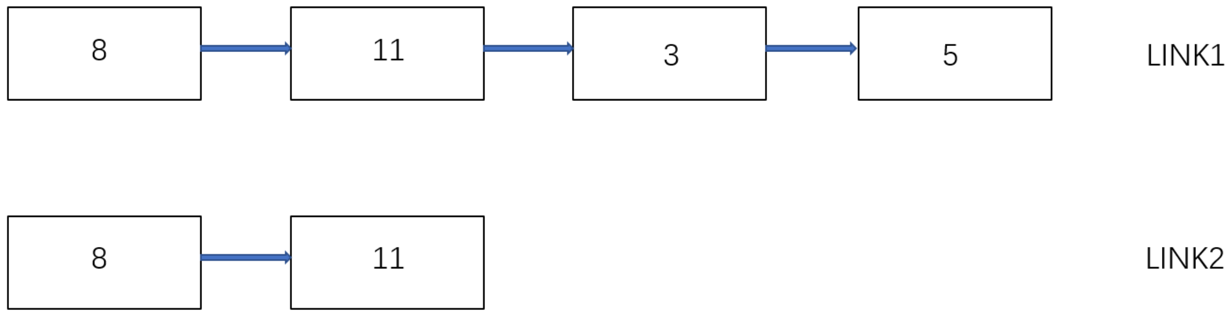
辗转相除法的时间复杂度为 O(log(min(a,b)))，而更相减损术的时间复杂度取决于 a 和 b 的差值。在最坏情况下，更相减损术的时间复杂度可能达到 O(max(a,b))。

因此，在大多数情况下，**辗转相除法比更相减损术更快。**

# 给出方案返回一可以动态变化的序列（可增可减）中的最大值，并对你的方案进行分析。

根据我们第一次的思考题，我们可以创造一链表（存放所有元素）和一链表（存放最大值）去解决这个问题

### 算法思路：




假设一个动态序列目前是8 11 3 5

则按照顺序得到的最大链表的顺序为8 11

若LINK1里面删掉11

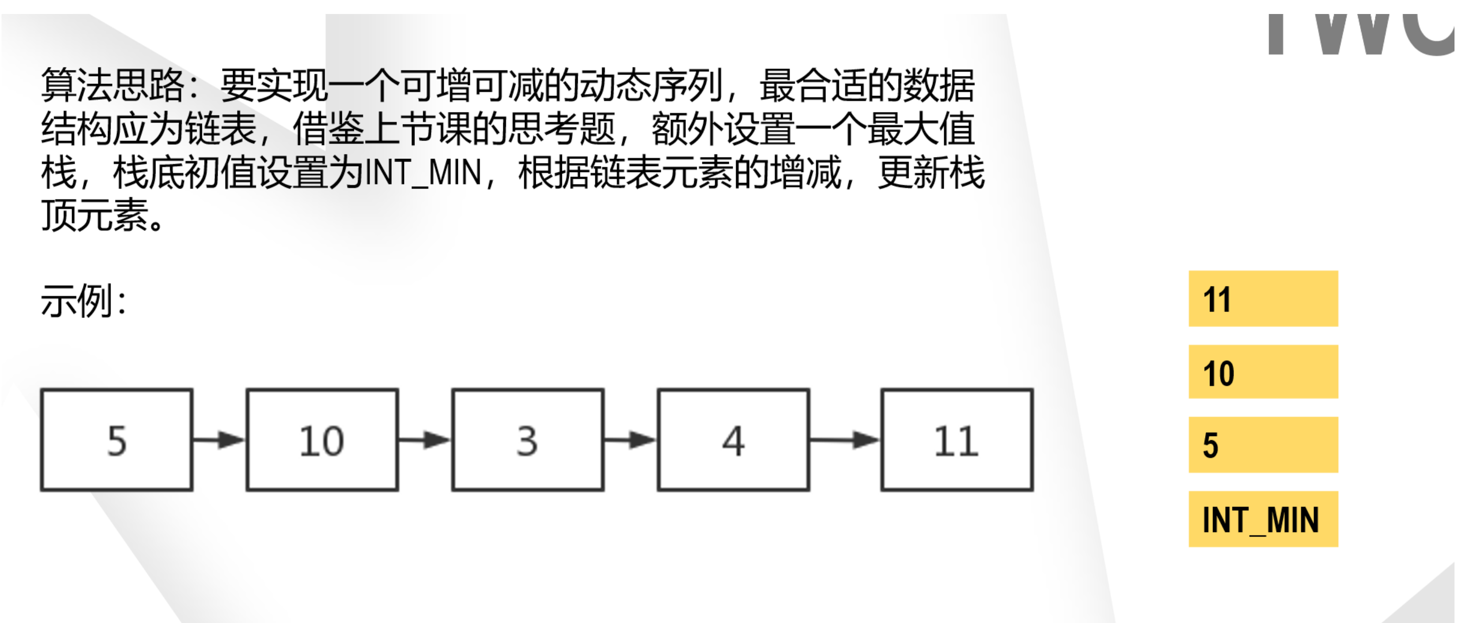
我们可以在LINK2里面删除11

若LINK1里面删掉8

则LINK2里面删除左边的8

可以实现LINK2中最右边元素始终为最大元素

## 对于课上同学问题的思考




上图是课上同学分享的方法

我有一些质疑：

假设我们在原序列中删除10

（1）右边栈中若不弹出，始终保持原状态

那么我们在原序列删除11的时候，将栈中11弹出，10被留下，10将没有意义是错误的。

（2）若右边栈中将11和10全部弹出，再压入11这种方法是十分繁琐的。因为在考虑到一个较长的序列，这种只为了删除栈中间某个元素的行为是非常费力的，不如利用链表更加方便。

鉴于该同学课上的时候，仅通过一张简单的PPT来演示这个过程，并没有细细说明具体算法，因此我在这里提出了一些质疑，并提议用链表去记录最大元素更好。

## 代码实现：

...因为我是转专业来到计算机学院的，数据结构还没有学（目前我在老师您的数据结构班上学习），所以对于一些结构的代码实现并不太了解，因此只用图的形式给出了算法思路。