

## The Euclidean algorithm

The first known appearance of the Euclidean algorithm was in book VII of Euclid's elements, where it is used to find the gcd of two numbers. The proof shows implicitly that if  $a$  and  $b$  are any two positive integers then there exist integers  $k$  and  $\ell$  such that

$$ka + \ell b = c$$

where  $c$  is the gcd of  $a$  and  $b$ . It also allows you to find  $k$  and  $\ell$  explicitly, but only by going to the end of the construction and then backing up. It is in fact not necessary to back up in this inefficient way, if only some extra data is carried along as the calculation proceeds.

The basic algorithm proceeds by carrying out a sequence of integer divisions until the remainder becomes 0. Here is a Java procedure that does this:

```
int gcd(int a, int b) {
    while (b > 0) {
        // assert: b > 0, gcd(a,b) does not change in the loop
        int q = a/b;
        int r = a - q*b;
        a = b;
        b = r;
    }
    // the loop exits with (a, b) = (a, 0) and a = gcd of original a, b
    return(a);
}
```

In order to recover  $k$  and  $\ell$  at the end, we carry along a  $2 \times 2$  matrix  $E$  with the property that at the start of the loop, we have

$$\begin{pmatrix} a \\ b \end{pmatrix} = E \begin{pmatrix} a_0 \\ b_0 \end{pmatrix}$$

where  $a_0$  and  $b_0$  are the original  $a$  and  $b$ . At the start  $E = E_0$  is the identity matrix. In each loop, say in step  $n$ , we set

$$\begin{aligned} a_{n+1} &= b_n \\ b_{n+1} &= a_n - q_n b_n \end{aligned}$$

which means that

$$E_{n+1} = \begin{bmatrix} 0 & 1 \\ 1 & -q \end{bmatrix} E_n,$$

At the end,  $(k, \ell)$  is the top row of  $E$ . So we rewrite the method above to return the  $(k, \ell)$  vector:

```
int[] euclid(int a, int b) {
    // represent E by rows
    int[][] E = {{1, 0}, {0, 1}};
    while (b > 0) {
        // assert: b > 0, gcd(a,b) does not change in the loop
        // E * (a_0, b_0) = (a, b)
        int q = a/b;
        int r = a - q*b;
        a = b;
        b = r;
        // E = ( 0 1)/(1 -q) E
        int tmp = E[1][0];
        E[1][0] = E[0][0] - q*tmp;
        E[0][0] = tmp;
    }
}
```

---

```
    tmp = E[1][1];
    E[1][1] = E[0][1] - q*tmp;
    E[0][1] = tmp;
}
// the loop exits with (a, b) = (a, 0) and a = gcd of original a, b
int[] e = {E[0][0], E[0][1]};
return(e);
}
```