Algorithm for Polynomial Division

Input: Two polynomials $a(x)$ and $b(x)$.

Output: Quotient polynomial $q(x)$ and remainder polynomial $r(x)$.

Step 1. $q := [\ ];\ a' := a;\ c := lc(b);\ m := deg(a');\ n := deg(b)$.
Step 2. While $m \geq n$ do
    Step 2.1 $d := lc(a')/c;\ q := CONS(d, q);\ a' := a' - d \cdot x^{m-n} \cdot b$.
    Step 2.2 For $i = 1$ to min $\{m - deg(a') - 1, m - n\}$ do
        $q := CONS(0, q);\quad m := deg(a')$.
Step 3. $q := INV(q);\ r := a';$ return.

Algorithm for Polynomial Pseudo-Division

Input: Two polynomials $a(x)$ and $b(x)$.

Output: Pseudoquotient polynomial $q(x)$ and pseudoremainder polynomial $r(x)$.

Step1. $q := [\ ];\ a' := a;\ c := lc(b);\ m := deg(a');\ n := deg(b)$.
Step 2. $c^{(1)} := c;$ for $i = 2$ to $m - n$ do $c^{(i)} := c \cdot c^{(i-1)}$.
Step 3. While $m \geq n$ do
    Step 3.1 $d := lc(a')c^{(m-n)};\ q := CONS(d, q);\quad a' := c \cdot a' - lc(a') \cdot x^{m-n} \cdot b$.
    Step 3.2 For $i = 1$ to min $\{m - deg(a') - 1, m - n\}$ do
        $q := CONS(0, q);$
        $a' := c \cdot a'$.
    Step 3.3 $m := deg(a')$.
Step 4. $q := INV(q);\ r := a';$ return.

Algorithm for Polynomial GCD Computation

Input: Two polynomials $a(x)$ and $b(x)$ in $I[x]^*$.

Output: The GCD g of $a(x)$ and $b(x)$.

Step 1. If $deg(a) \geq deg(b)$
    then $f_1 = pp(a);\ f_2 = pp(b);$
    else $f_1 = pp(b);\ f_2 = pp(a)$.
Step 2. $d := GCD(cont(a), cont(b))$.
Step 3. Compute a prs sequence:$f_1, \cdots, f_k, f_{k+1} = 0$.
Step 4. $g := d \cdot pp(f_k)$, return.

Algorithm for Subresultant PRS

Input: Two polynomials $f_1(x)$ and $f_2(x)$.

Output: The subresultant prs $F := [f_1, f_2, \cdots, f_k, 0]$ up to sign.

Step 1. $F := [f_2, f_1]$;
$g := 1;\ h := 1, f' := f_2;\ i := 3.$

Step 2. While $f' \neq 0$ and $deg(f') > 0$ do

Step 2.1 $\delta := deg(f_{i-2}) - deg(f_{i-1})$;
$f' := prem(f_{i-2}, f_{i-1})$;

Step 2.2 If $f' \neq 0$ then
$f_i := f'/(g \cdot h^{\delta});\ F := CONS(f_i, F)$;
$g := lc(f_{(i-1)};\ h := h^{1-\delta} \cdot g^{\delta}$;
$i := i + 1$.

Step 3. $F := INV(F)$; return.

Algorithm for Modular gcd

Input: $a, b \in Z[x]^*$ primitive.

Output: $g = \gcd(a, b)$.

Step 1. $d := \gcd(\mathrm{lc}(a), \mathrm{lc}(b)); M := 2 \cdot d \cdot$ (Landau-Mignotte bound for a, b).

Step 2. $p :=$ a new prime not dividing d;
$c_{(p)} := \gcd(a_{(p)}, b_{(p)})$; (with $\mathrm{lc}(c_{(p)} = 1)$
$g_{(p)} := (d \mod p) \cdot c_{(p)}$.

Step 3. If $\deg(g_{(p)}) = 0$ then return $g := 1$.
$P := p$;
$g := g_{(p)}$.

Step 4. While $P \leq M$ do

1. $p :=$ a new prime not dividing d;
2. $c_{(p)} := \gcd(a_{(p)}, b_{(p)})$; [with $\mathrm{lc}(c_{(p)} = 1]$
3. $g_{(p)} := (d \mod p) \cdot c_{(p)}$;
4. if $\deg g_{(p)} < \deg(g)$ then goto Step 3;
5. if $\deg g_{(p)} = \deg(g)$ then
   $g := \mathrm{CRA}_2(g, g_{(p)}, P, p)$;
   (actually CRA_2 is applied to the coefficients of g and $g_{(p)}$)
   $P := P \cdot p$.

Step 5. $g := \mathrm{pp}(g)$;
if $g|a$ and $g|b$ then return;
goto Step 2.