# Компьютерная алгебра

## (курс лекций)

Игорь Алексеевич Малышев

Computer.Algebra@yandex.ru

КСПТ

КОМПЬЮТЕРНЫЕ СИСТЕМЫ И ПРОГРАММНЫЕ ТЕХНОЛОГИИ

# АЛГОРИТМЫ ВЫЧИСЛЕНИЙ В КОМПЬЮТЕРНОЙ АЛГЕБРЕ

# Лекция 11

# Вычисление НОД целых чисел и полиномов

# Содержание лекции

- Отношение делимости и его свойства
- Алгоритмы вычисления НОД в кольце целых чисел
- Алгоритмы вычисления НОД в кольцах полиномов

# План лекции: тема подраздела

- **Отношение делимости и его свойства**
- Алгоритмы вычисления НОД в кольце целых чисел
- Алгоритмы вычисления НОД в кольцах полиномов

# Отношение делимости и его свойства

## Вводные замечания

**Замечание 1.** Метод решения задачи вычисления наибольшего общего делителя (НОД) двух целых чисел, предложенный Евклидом ( III век до н.э. ), является самым древним из существующих теоретико-числовых алгоритмов.

**Замечание 2.** Основными алгебраическими структурами, в которых решается задача вычисления НОД, являются следующие:

**Z** – кольцо целых чисел;

**F** [x] – кольцо полиномов от одной переменной над произвольным полем **F**;

**Z** [x] – кольцо полиномов от одной переменной над кольцом целых чисел **Z**.

**Замечание 3.** Рассматриваемые далее алгоритмы вычислений НОД основаны на обобщении и анализе свойств введённые ранее (см. лекцию 6) следующих понятий: отношение делимости, НОД и НОК целых чисел.

**Замечание 4.** Основное внимание уделено сущности алгоритмов и оценкам их сложности. Необходимые дополнительные алгебраические понятия имеют вид постулатов, т.е. определений (иногда сопровождаемых примерами) и утверждений (теорем) без доказательств.

# Отношение делимости и его свойства

## Делимость в коммутативных кольцах

**Определение.** Пусть $R$ – коммутативное кольцо с единицей, $a, b \in R$.
Мы говорим, что ненулевой элемент ***a* делит *b*** и пишем $a \mid b$,
если существует элемент $c \in R$, такой, что $b = a * c$;
если такого элемента не существует, то мы говорим, что $a$ не делит $b$.

**Пример.** (отношение делимости целых чисел)
$\pm 7 \mid 28$ , т.к. $28 = 7 * 4$ и $28 = ( - 7 ) * ( - 4 )$

**Замечание.** (к отношению делимости целых чисел)
Для любого ненулевого a имеем: $a \mid 0$ , $\pm 1 \mid a$ , $\pm a \mid a$

**Замечание.** Определение делимости зависит от рассматриваемого кольца.
Так, например, $2 \mid 3$ в поле рациональных чисел **Q**, но не в кольце целых чисел **Z**.

# Отношение делимости и его свойства

## Неприводимые элементы кольца

**Определение.** Ненулевой элемент $a \in R$ такой, что $a * b = 0$ для некоторого $b \neq 0$ называется делителем нуля кольца R, а элемент $\varepsilon \in R$, такой, что $\varepsilon \mid 1$ называется **обратимым** или единицей кольца R.

**Определение.** Коммутативное кольцо с единицей и без делителей нуля называется **областью целостности** или просто областью.

**Определение.** Пусть R – коммутативное кольцо с единицей. Элемент $a \in R$ называется **неприводимым**, если из представления $a = b * c$ в виде произведения двух элементов кольца R, следует, что хотя бы один из элементов $b$ и $c$ обратим в R.

# Отношение делимости и его свойства

## Кольцо главных идеалов

**Определение.** Пусть R – коммутативное кольцо с единицей.
Идеал I кольца R ( $I \subset R$ ) называется **простым**, если из $b * c \in I$
следует, что хотя бы один из элементов *b* и *c* лежит в I.

**Определение.** Мы говорим, что идеал I
**порожден** элементами $b_1, \ldots, b_n$, и пишем $I = \{ b_1, \ldots, b_n \}$,
если $b_1, \ldots, b_n, \in I$ и любой элемент $b \in I$
может быть записан в виде $b = \sum^n_{i=1} ( c_i * b_i )$, где $c_i \in R$.

**Определение.** Идеал I называется **главным**,
если $I = \{ b \}$ для некоторого элемента $b \in I$.
Кольцо R называется **кольцом главных идеалов**,
если любой идеал кольца R является главным.

**Замечание.** Главный идеал { b } является простым тогда и только тогда,
когда b – неприводимый элемент.

# Отношение делимости и его свойства

## Факториальное кольцо – 1

**Определение.** Элементы *a* и *b* кольца R называются **ассоциированными**, если a = ε * b, где ε – единица (обратимый элемент) кольца R.

**Определение.** Кольцо R называется **факториальным** или **кольцом с однозначным разложением на множители**, если любой элемент $a \in R$ можно представить в виде $a = \varepsilon * p_1 * \cdots * p_n$, где ε – единица кольца, а $p_i$ – неприводимые (не обязательно различные), причем если $a = \varepsilon_1 * q_1 * \cdots * q_m$ – другое такое разложение, то m = n и для любого индекса i существует индекс j, такой, что $p_i$ ассоциировано с $q_j$.

**Пример.**
Факториальным кольцом является кольцо целых чисел **Z**, а также любое поле **F**.

# Отношение делимости и его свойства

## Факториальное кольцо – 2

**Лемма.** Пусть R – область главных идеалов.
Если элемент a $\in$ R допускает разложение на неприводимые множители,
то это разложение однозначно в смысле определения факториального кольца.

**Теорема.** Если R – факториальное кольцо,
то кольцо многочленов R [x] также факториально.

**Замечание.** Указанная теорема позволяет получать
новые факториальные кольца.

**Пример.** Кольцо **Z** [ $\sqrt{}$ (– 5) ] – не факториально.
В частности, имеем

$9 = 3 * 3$

$9 = ( 2 + \sqrt{} (– 5) ) * ( 2 – \sqrt{} (– 5) )$

два различных разложения числа 9
на неприводимые множители в этом кольце.

# Отношение делимости и его свойства

## Наибольший общий делитель (НОД)

**Определение.** Пусть R – коммутативное кольцо с единицей, $a, b \in R$.
Элемент $d \in R$ называется **наибольшим общим делителем** элементов $a$ и $b$,
если $d \mid a$ , $d \mid b$ и для любого другого элемента $d'$, такого, что $d' \mid a$ и $d' \mid b$,
выполняется соотношение $d' \mid d$.

**Замечание 1.** В кольце целых чисел **Z** для любых целых чисел $a$ и $b$,
не равных одновременно нулю, существует наибольшее целое число,
которое делит $a$ и $b$, и это число является НОД чисел $a$ и $b$.
Однако это определяет НОД ($a, b$) неоднозначно.
Для однозначности и единственности требуется условие d > 0.

**Пример.** НОД ( 12, 30 ) = НОД ( 12, – 30 ) = НОД ( – 12, 30 ) = НОД ( – 12, – 30 ) = 6

**Замечание 2.** В кольце **F** [x] многочленов от одной переменной $x$ над полем **F**
для любых многочленов $a$ и $b$, не равных одновременно нулю,
существует многочлен наибольшей степени, который делит $a$ и $b$,
и этот многочлен является НОД $a$ и $b$.
Однако это определяет НОД ($a, b$) неоднозначно (см. Замечание 1).

# Отношение делимости и его свойства

## Свойства НОД в кольце целых чисел Z

**(1)** НОД ( a, a ) = { a, – a }

**(2)** НОД ( a, 0 ) = { a, – a }

**(3)** НОД ( a, b ) = НОД ( b, a )

**(4)** НОД ( c * a, c * b ) = c * НОД ( a, b )

**(5)** если НОД ( a, c ) = { 1, – 1 } (в частности, если c = – 1),
то НОД ( a, c * b) = НОД ( a, b )

**(6)** НОД ( a, b ) = НОД ( a – b, b )

**(7)** НОД ( a, b ) = НОД ( b, r ), где r – остаток от деления a на b.

# Отношение делимости и его свойства

## Виды алгоритмов вычисления НОД в Z

| № п/п | Краткая характеристика алгоритма | Используемые свойства НОД | Алгебраическая структура | Результат алгоритма |
|---|---|---|---|---|
| 1. | Сведение исходной задачи к задаче вычисления НОД в $\mathbf{Z}_+$ | **(3) (5)** | Множество неотрицательных целых чисел | Только НОД > 0 |
| 2. | Простейший алгоритм вычисления НОД | **(1) (3) (6)** | Кольцо целых чисел **Z** | НОД в **Z** |
| 3. | Бинарный алгоритм вычисления НОД | **(1) (4) (5)** при c = 2 | Кольцо целых чисел **Z** | НОД в **Z** |
| 4. | Алгоритм Евклида для натуральных чисел | **(2) (7)** | Множество натуральных чисел | НОД в **N** |

# Отношение делимости и его свойства

## Евклидовы кольца

**Замечание.** Свойство (7) использует понятие «остаток от деления одного числа на другое», на котором основан алгоритм Евклида (в кольце целых чисел).
Для распространения действия этого алгоритма на другие кольца необходимо обобщить свойство евклидовости (делимости).

**Определение.** Область целостности R называется **евклидовым кольцом**, если каждому ненулевому элементу $a \in R$ сопоставлено целое неотрицательное число g (a) со следующими свойствами:
(1) Если $a \neq 0$ и $b \neq 0$ , то $g(a * b) \geq g(a)$
(2) Для любых двух элементов $a, b \in R$ , где $b \neq 0$
существует представление $a = q * b + r$ , в котором $r = 0$ или $g(r) < g(b)$

**Замечание.** Евклидовыми кольцами являются следующие алгебраические структуры:
- Кольцо целых чисел **Z**;
- Кольцо многочленов **F** [x] от одной переменной над любым полем **F**;
- Любое поле **F**.

**Теорема.** Любое евклидово кольцо является кольцом главных идеалов, и, следовательно, факториальным кольцом

# План лекции: тема подраздела

- Отношение делимости и его свойства
- **Алгоритмы вычисления НОД в кольце целых чисел**
- Алгоритмы вычисления НОД в кольцах полиномов

# Алгоритмы вычисления НОД в $\mathbb{Z}$

## Вводные замечания

**Замечание 1.** Во всех изложенных ниже алгоритмах полагаем, что a и b – натуральные числа, следовательно, ненулевые.

**Замечание 2.** Результатом работы алгоритмов является следующее число:

d = НОД ( a , b )

# Алгоритмы вычисления НОД в $\mathbb{Z}$

## Простейший алгоритм вычисления НОД

**Алгоритм 1.**

**Вход:** a, b $\in$ N
**Переменные:** x, y $\in$ N
x := a
y := b
**Цикл Пока** x $\neq$ y
    **Если** x > y, **то**
        x := x − y
    **Иначе**
        y := y − x
    **Конец Если**
**Конец Цикла**
d := x

**Выход:** d $\in$ N

# Алгоритмы вычисления НОД в $\mathbb{Z}$

## Сложность простейшего алгоритма

Простейший алгоритм вычисления НОД использует следующие операции:

- Сравнение натуральных чисел;
- Вычитание натуральных чисел;
- Присваивание переменной значения натурального числа.

Оценивая сложность алгоритма,
можно рассматривать указанные операции как элементарные.

В теле цикла «пока» выполняются:

- 2 операции сравнения;
- 1 операция вычитания;
- 1 операция присваивания.

Цикл выполняется не более, чем max (a, b) раз.

Таким образом, сложность алгоритма равна **O ( max ( a, b ) )**

**Замечание.** Более естественно рассматривать в качестве элементарных битовые операции.
Тогда, если max (a, b) = n ,
то сложность вычисления НОД по данному алгоритму равна **O ( n log $_2$ ( n ) )** битовых операций.

# Алгоритмы вычисления НОД в $\mathbb{Z}$

## Алгоритм Евклида вычисления НОД

**Алгоритм 2.**

**Вход:** a, b $\in$ N

**Выход:** d $\in$ N

**Переменные:** r, x, y $\in Z_+$

x := a
y := b
**Цикл Пока** y $\neq$ 0
    r := x mod y
    x := y
    y := r
**Конец Цикла**
d := x

# Алгоритмы вычисления НОД в $\mathbb{Z}$

## Сложность алгоритма Евклида

**Замечание.** В алгоритме Евклида использовано стандартное обозначение операции вычисления остатка от деления x на y в виде x mod y .

Легко показать, что после двух делений делимое уменьшается, как минимум, в 2 раза.
Значит, количество повторений цикла равно $O ( \log_2 ( n ) )$ , где $n = \max (a, b)$ .

Учитывая, что битовая сложность операции деления квадратично зависит от количества цифр, получим оценку битовой сложности алгоритма Евклида, которая равна $\mathbf{O ( ( \log_2 ( n ) )^3 )}$

**Замечание.** Более тщательный анализ сложности алгоритма Евклида позволяет доказать оценку $\mathbf{O ( ( \log_2 ( n ) )^2 )}$

# Алгоритмы вычисления НОД в $\mathbb{Z}$

## Теорема Ламе – 1

**Замечание.** Можно показать, что наихудший (в определённом смысле) вариант для алгоритма Евклида представляют последовательные числа Фиббоначчи.

**Теорема Габриэля Ламе** (1844)
**(оценка наихудшего случая для алгоритма Евклида)**
Количество делений, которое необходимо выполнить
для вычисления НОД двух целых чисел,
не превосходит количества цифр в меньшем числе,
умноженного на 5.

Для доказательства теоремы Ламе использовал несколько замечательных свойств, которыми обладают числа, образующие последовательность Фиббоначчи.

Напомним, что последовательность Фиббоначчи строится по следующему правилу: каждый следующий член последовательности равен сумме двух предыдущих.

Таким образом, имеем следующее начало последовательности Фиббоначчи:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597 , …

# Алгоритмы вычисления НОД в $\mathbb{Z}$

## Теорема Ламе – 2

Свойства целых чисел,
образующих последовательность Фиббоначчи:

(1) Число членов в этой последовательности,
образующих интервал чисел,
имеющих одинаковое количество цифр,
не меньше четырёх и не больше пяти
(единственное исключение из этого правила – начальное число 1)

(2) Если $a = F_{n+1}$ , $b = F_n$ – соответствующие числа Фиббоначчи,
то остатки в алгоритме Евклида принимают последовательно значения
$F_{n-1}$ , … , $F_2 = 1$

(3) Если $a > b$ и $r_1$ , … , $r_n \neq 0$ – последовательные остатки,
получаемые в алгоритме Евклида,
то $a \geq F_{n+1}$ , где $F_k$ – k-е число Фиббоначчи.

# Алгоритмы вычисления НОД в $\mathbf{Z}$

## Обобщение алгоритма Евклида

**Замечание.** Алгоритм Евклида можно применить
в любом евклидовом кольце,
т.е. условия $a , b \in \mathbf{N} , d \in \mathbf{N} , r , x , y \in \mathbf{Z}_+$
можно заменить на
$a , b , d , r , x , y \in \mathbf{R}$ , где **R** – любое евклидово кольцо.

# Алгоритмы вычисления НОД в $\mathbb{Z}$

## Бинарный алгоритм вычисления НОД

**Алгоритм 3.**

**Вход:** a, b ∈ N **Выход:** d ∈ N,

**Переменные:** x, y ∈ N
x := a ; y := b ; d := 1
**Цикл Пока** ( x mod 2 = y mod 2 = 0 )
d := 2 * d, x := x / 2, y := y / 2
**Конец Цикла**

**Цикл Пока** x ≠ y
**Выбор**
**если** x mod 2 = 0 **то** x := x / 2
**если** y mod 2 = 0 **то** y := y / 2
**если** x > y **то** x := x − y
**если** x < y **то** y := y − x
**Конец Выбора**
**Конец Цикла**
d := d * x

# Алгоритмы вычисления НОД в $\mathbb{Z}$

## Сложность бинарного алгоритма – 1

**Замечание 1.** Бинарный алгоритм использует специфику машинной арифметики
в системе счисления по основанию 2,
в которой операции умножения и деления на 2 сводятся к сдвигам,
т.е. выполняются со скоростью аддитивных операций.

**Замечание 2.** В конструкции **Выбор** выполняются только действия
для первого истинного условия в операторах **если**.
Таким образом, первые два условия соответствуют случаям,
когда один из аргументов – чётный, а другой – нечётный,
а последние два условия – только случаю, когда оба аргумента – нечётные.
Учитывая, что при вычитании нечётного числа из нечётного получается чётное,
можно результат сразу разделить на 2, т.е. заменить строки:
**если** x > y **то** x := x − y
**если** x < y **то** y := y − x

на строки:
**если** x > y **то** x := ( x − y ) / 2
**если** x < y **то** y := ( y − x ) / 2

# Алгоритмы вычисления НОД в $\mathbb{Z}$

## Сложность бинарного алгоритма – 2

При каждом повторении тела цикла
хотя бы один аргумент уменьшается в 2 раза.
Значит, количество повторений цикла равно
$O ( \log_2 ( n ) )$ , где $n = \max ( a , b )$ .

Учитывая, что битовая сложность выполнения операций
в теле цикла линейно зависит от количества цифр,
получим, что битовая сложность бинарного алгоритма
вычисления НОД равна $\mathbf{O ( ( \log_2 ( n ) )^2 )}$

# Алгоритмы вычисления НОД в $\mathbb{Z}$

## Ещё один алгоритм вычисления НОД

**Алгоритм 4.** (вычисление НОД с помощью разложения на простые числа)

**Вход:** a, b ∈ N **Выход:** d ∈ N,

**Переменные:** x, y, p ∈ N, p — простое число
x := a ; y := b ; p := 2 ; d := 1
**Цикл Пока** x ≠ 1 или y ≠ 1
**Цикл Пока** x mod p = y mod p = 0
d = d * p ; x = x / p ; y = y / p
**Конец Цикла**
**Цикл Пока** x mod p = 0
x = x / p
**Конец Цикла**
**Цикл Пока** y mod p = 0
y = y / p
**Конец Цикла**
p := следующее простое число
**Конец Цикла**

**Замечание.** Данный алгоритм основан на свойстве факториальности кольца **Z**.

# Алгоритмы вычисления НОД в $\mathbb{Z}$

## Теорема существования НОД

Наибольший общий делитель двух целых чисел
обладает важным свойством, которому посвящена следующая

**Теорема.** Если a и b одновременно не равны нулю,
то существуют целые числа u и v , такие, что НОД ( a , b ) = a * u + b * v

**Замечание 1.** Наличие у НОД в кольце целых чисел **Z** свойства,
указанного в теореме, легко доказать в силу того, что **Z** – кольцо главных идеалов.

**Замечание 2.** Теорема не утверждает, что числа u и v определены однозначно.
Она говорит лишь о том, что НОД может быть выражен в таком виде.

**Пример.** a = 12 , b = – 30 , d = НОД ( a , b ) = 6

6 = ( 12 , – 30 ) = 12 * ( 3 ) + ( – 30 ) * 1 ( u = 3 , v = 1 )
= 12 * ( – 2 ) + ( – 30 ) * ( – 1 ) ( u = – 2 , v = – 1 )

# Алгоритмы вычисления НОД в $\mathbb{Z}$

## Расширенный алгоритм Евклида

**Алгоритм 5.** (вычисление целых чисел u и v )

**Вход:** $a, b \in N$ **Выход:** $d \in N, u, v \in Z$

**Переменные:** R, X, Y — векторы элементов типа Z с индексами 0, 1, 2
$q \in Z_+$
**Обозначения:** $r \equiv R[0]$, $x \equiv X[0]$, $y \equiv Y[0]$

```
X := (a, 1, 0)
Y := (b, 0, 1)

Цикл Пока y ≠ 0
        q := [ x / y ]          // [ x / y ] – обозначение целой части дроби x / y
        R := X − q * Y
        X := Y
        Y := R
Конец Цикла
(d, u, v) := X
```

# Алгоритмы вычисления НОД в $\mathbb{Z}$

## Сложность расширенного алгоритма Евклида

**Замечание 1.** Три компонента вектора X связаны соотношением:

X [ 0 ] = X [ 1 ] * a + X [ 2 ] * b

Такие же соотношения справедливы для Y и R.

**Замечание 2.** Чтобы распространить расширенный алгоритм Евклида на отрицательные и нулевые целые числа, достаточно заменить две первые строки:

X := (a, 1, 0)

Y := (b, 0, 1)

на следующие строки:

**Если** a ≥ 0 **то** X := (a, 1, 0) **иначе** X := (- a, - 1, 0)

**Если** b ≥ 0 **то** Y := (b, 0, 1) **иначе** Y := (- b, 0, - 1)

Битовая сложность расширенного алгоритма Евклида равна $\mathbf{O\left(\left(\log_2(n)\right)^2\right)}$, где n = max ( | a | , | b | )

# Алгоритмы вычисления НОД в $\mathbb{Z}$

## Заключительные замечания к алгоритмам

**Замечание 1.** Кроме расширенной версии алгоритма Евклида существуют расширенные версии для:

- простейшего алгоритма вычисления НОД (см. **Алгоритм 1**);
- бинарного алгоритма вычисления НОД (см. **Алгоритм 3**).

**Замечание 2.** Алгоритм Евклида и бинарный алгоритм вычисления НОД являются достаточно эффективными для большинства приложений. Однако при проектировании высокопроизводительных систем могут быть использованы различные методы уменьшения сложности алгоритмов вычисления НОД.
К таким методам относится, в частности, **алгоритм Деррика Генри Лемера**.
Он наиболее эффективен при вычислении НОД длинных целых чисел, которое частично можно выполнить над их старшими разрядами (т.е. числами однократной точности).
При этом количество циклов вычислений с многократной точностью, которые можно заменить в каждой итерации, пропорционально log (n) , где n – разрядность числа однократной точности.

# План лекции: тема подраздела

- Отношение делимости и его свойства
- Алгоритмы вычисления НОД в кольце целых чисел
- **Алгоритмы вычисления НОД в кольцах полиномов**

# Алгоритмы вычисления НОД в F/Z[x]

## Евклидовы области

**Определение**. **Евклидова область** – это область целостности J
вместе с функцией «степени» (или «порядка») $d : J \setminus 0 \to \mathbf{N}$
такой, что :

(1) $d(p_1 * p_2) \geq d(p_1)$, $p_1 \neq 0$, $p_2 \neq 0$

(2) для любых элементов $p_1$ и $p_2$ из J ($p_2 \neq 0$) в J существуют элементы q и r, обладающие свойством евклидовости :
$p_1 = p_2 q + r$ , $d(r) < d(p_2)$ или $r = 0$ .

**Пример (евклидовы области) :**

1) J = **Z** при d (p) = | p | – это евклидова область, т.к. для $p_1, p_2 \in \mathbf{Z}$, $p_2 \neq 0$ существуют q и r такие, что $p_1 = p_2 q + r$ , $0 \leq r < | p_2 |$

2) Если J – поле, то J [x] при d (p (x)) = deg (p (x)) – это евклидова область, т.к. для любых $p_1(x), p_2(x) \in J[x]$, $p_2(x) \neq 0$, в J [x] существуют единственные полиномы q (x) , r (x) , такие, что
$p_1(x) = p_2(x) * q(x) + r(x)$ , $\deg(r(x)) < \deg(p_2(x))$
( deg (p (x)) – обозначает степень полинома p (x) )

**Пример (неевклидовы области) :** (1) J = **Q** – поле рациональных чисел, при d (p) = | p | ;
(2) J = **Z** [x] – кольцо полиномов, при d (p (x)) = deg (p (x))

# Алгоритмы вычисления НОД в $\mathbf{F/Z[x]}$

## НОД полиномов

**Определение**. Пусть J – область целостности и $p_1(x), p_2(x) \in J[x], p_2(x) \neq 0$
Полином $p_h(x)$ из J [x] называется **наибольшим общим делителем** (**НОД**) полиномов $p_1(x)$ и $p_2(x)$, если выполнены следующие условия :

(1) $p_h(x) \mid p_1(x)$ и $p_h(x) \mid p_2(x)$

(2) если $q(x) \mid p_1(x)$ и $q(x) \mid p_2(x)$, то $\deg(q(x)) \leq \deg(p_h(x))$ и $q(x) \mid p_h(x)$

**Замечание.** В общем случае, бессмысленно говорить о «единственном» НОД двух полиномов, т.к. в алгебраической системе J может быть много обратимых элементов, т.е. если $p_h(x) = \text{НОД}(p_1(x), p_2(x))$,
то произведение $a * p_h(x)$, где a – обратимый элемент,
также является $\text{НОД}(p_1(x), p_2(x))$.

**Замечание**. Два полинома в J [x] называются **взаимно простыми**,
если любой их НОД – обратимая константа из J .

**Замечание**. Алгоритм Евклида для полиномов над полем – простая процедура.
Вычисление НОД в **Z** [x] значительно сложнее, т.к. **Z** [x] – неевклидова область.

# Алгоритмы вычисления НОД в $\mathbf{F/Z[x]}$

## Расширенный алгоритм Евклида для полиномов над полем

**Алгоритм 6.** ( **XEA-P –** E**x**tended **E**uclidean **A**lgorithm for **P**olynomials over a Field )

**Вход :**

$p_1(x)$ , $p_2(x)$ – полиномы в J [x], J – поле
$p_2(x) \neq 0$
$\deg(p_1(x)) = n_1$
$\deg(p_2(x)) = n_2$
$n_1 \geq n_2$

**Выход :**

$p_h(x)$ , $f(x)$ , $g(x) \in J[x]$
такие , что
$\deg(f(x)) < \deg(p_1(x)) - \deg(p_h(x))$
$\deg(g(x)) < \deg(p_2(x)) - \deg(p_h(x))$
$p_h(x) = НОД(p_1(x), p_2(x)) =$
$= p_1(x) * g(x) + p_2(x) * f(x)$

**[ Инициализация ]**
$[p_0(x), p_1(x)] := [p_1(x), p_2(x)]$
$[g_0(x), g_1(x)] := (1, 0)$
$[f_0(x), f_1(x)] := (0, 1)$
**[ Основной цикл ]**
**Цикл Пока** $p_1(x) \neq 0$
$q(x) :=$ **PDF** $(p_0(x), p_1(x))$ // **PDF** - алгоритм полиномиального деления над полем
$[p_0(x), p_1(x)] := [p_1(x), p_0(x) - p_1(x) * q(x)]$
$[g_0(x), g_1(x)] := [g_1(x), g_0(x) - g_1(x) * q(x)]$
$[f_0(x), f_1(x)] := [f_1(x), f_0(x) - f_1(x) * q(x)]$
**Конец Цикла Пока**
**[ Формирование результата ]** $[p_h(x), g(x), f(x)] := [p_0(x), g_0(x), f_0(x)]$

# Алгоритмы вычисления НОД в $F/Z[x]$

## Сложность алгоритма XEA-P

**Замечание**. Очевидно, что временная сложность алгоритма **XEA-P** определяется длительностью выполнения шага **[ Основной цикл ]** , которое равно $\mathbf{O [ n_2 (n_1 - n_2 + 1) ]}$ .

Т.к. количество выполнений шага **[ Основной цикл ]** не превышает $n_2$ , то временная сложность расширенного алгоритма Евклида для полиномов над полем равна $\mathbf{O [ (n_2)^2 (n_1 - n_2 + 1) ]}$ .

# Алгоритмы вычисления НОД в $\mathbf{F/Z[x]}$

## Последовательность полиномиальных остатков (PRS)

**Замечание. Алгоритм 2** и **Алгоритм 5** без изменений применимы
в любом евклидовом кольце,
в частности, в кольце полиномов **F** [x] от одной переменной
над произвольным полем **F**.

**Определение.** Последовательность остатков полиномов,
полученная при выполнении алгоритма Евклида,
называется **последовательность полиномиальных остатков**
( **PRS** ).

**Замечание.** Безусловно,
для вычисления последовательности полиномиальных остатков
можно использовать арифметику рациональных чисел.
Но при этом возникает серьёзная проблема –
обеспечение баланса емкостной (рост коэффициентов)
и временной (рост количества операций вычисления НОД целых чисел) сложности.

Наглядно покажем сущность проблемы на следующем примере.

# Алгоритмы вычисления НОД в F/Z[x]

## Проблема вычисления PRS – 1

**Пример.**

Рассмотрим следующие полиномы как элементы евклидова кольца **Q** [x]:

$p_1(x) = x^3 - 7x + 7$

$p_2(x) = 3x^2 - 7$

Применяя алгоритм Евклида, получим следующие последовательности:

$p_1(x) = x^3 - 7x + 7$

$p_2(x) = 3x^2 - 7$ $\qquad$ $q_1(x) = (1/3)x$

$p_3(x) = (-14/3)x + 7$ $\qquad$ $q_2(x) = (-9/14)x - 27/28$

$p_4(x) = -1/4$ $\qquad$ $q_3(x) = (56/3)x - 28$

$p_5(x) = 0$

Т.к. $p_4(x) = -1/4$ , то НОД ( $p_1(x)$ , $p_2(x)$ ) = 1

Рост коэффициентов PRS может быть минимизирован,
если производить нормировку каждого вновь вычисленного члена PRS.

# Алгоритмы вычисления НОД в $\mathbf{F/Z[x]}$

## Проблема вычисления PRS – 2

Таким образом, получим:

$p_1(x) = x^3 - 7x + 7$

$p_2(x) = x^2 - 7/3$ $\qquad$ $q_1(x) = x$

$p_3(x) = x - 3/2$ $\qquad$ $q_2(x) = x + 3/2$

$p_4(x) = 1$ $\qquad$ $q_3(x) = x - 3/2$

$p_5(x) = 0$

Очевидно, что коэффициенты растут медленнее,
но это происходит за счёт редуцирования дробей
– вычисления на каждом шаге НОД целых чисел.

**Замечание 1.** Фактически,
баланс емкостной и временной сложности недостижим,
т.к. возможность его получения существенно зависит от исходных полиномов.

**Замечание 2.** Необходим поиск более эффективных алгоритмов.

# Алгоритмы вычисления НОД в $\mathbf{F/Z[x]}$

## Лемма Гаусса

В приведенном выше примере коэффициенты полиномов являются целыми числами,
т. е. мы можем рассматривать эти полиномы как элементы кольца **Z** [x].

Поскольку это кольцо факториально (с однозначным разложением на неприводимые множители),
для любых двух элементов этого кольца, не равных одновременно нулю,
определен их наибольший общий делитель.

С другой стороны, из вложения **Z** [x] в **Q** [x] следует,
что мы можем рассматривать наибольший общий делитель этих же полиномов в кольце **Q** [x].

Как связаны между собой эти наибольшие общие делители ?

Известно очевидное различие этих НОД:
НОД в кольце **Z** [x] определен с точностью до знака,
а в кольце **Q** [x] – с точностью до умножения на любое ненулевое рациональное число.
Следующая лемма показывает, что иных отличий нет.

**Лемма (Гаусс).** Если коэффициенты многочлена $f \in \mathbf{Z}[x]$
взаимно просты в совокупности и $f = g * h$, где $g, h \in \mathbf{Q}[x]$
и НОД числителей коэффициентов каждого из многочленов g и h равен 1, то $g, h \in \mathbf{Z}[x]$.

# Алгоритмы вычисления НОД в $\mathbf{F/Z[x]}$

## Этапы вычисления НОД в кольце Z [x]

Пользуясь леммой Гаусса,
можно разбить вычисление НОД ( f (x) , g (x) ) в кольце **Z** [x]
на следующие этапы:

(1) Найти НОД $d_c \in \mathbf{Z}$ коэффициентов полиномов f (x) и g (x)

(2) Найти d (x) = НОД ( f (x) , g (x) ) в кольце **Q** [x] ,
нормированный таким образом,
что d (x) $\in$ **Z** [x] и коэффициенты полинома d (x) взаимно просты

(3) Теперь НОД ( f (x) , g (x) ) = $d_c$ d (x) в кольце **Z** [x]

# Алгоритмы вычисления НОД в $F/Z[x]$

## Содержание и примитивная часть полинома

**Определение.** НОД коэффициентов полинома $f(x) \in \mathbf{Z}[x]$
называется **содержанием** этого полинома
и обозначается cont ( f ) ( cont – сокращение от content ).
Полином f (x) / cont ( f ) называется **примитивной частью** полинома f (x)
и обозначается p.p. ( f (x) ) ( p.p. – сокращение от primitive part).

**Замечание.** Далее будем рассматривать задачу вычисления НОД
двух полиномов $p_1(x)$, $p_2(x)$ в кольце **Z** [x],
при условии, что все арифметические операции
над коэффициентами выполняются не в поле **Q**, а в кольце **Z**,
(являющимся не полем, а только областью
с однозначным разложением на множители) .

Из приведенных выше рассуждений получим следующие важные соотношения :

$$\text{cont}(\text{НОД}(p_1(x), p_2(x))) = \text{НОД}(\text{cont}(p_1(x)), \text{cont}(p_2(x)))$$
$$\text{p.p.}(\text{НОД}(p_1(x), p_2(x))) = \text{НОД}(\text{p.p.}(p_1(x)), \text{p.p.}(p_2(x)))$$

Таким образом, задача вычисления НОД произвольных полиномов
сводится к задаче вычисления НОД примитивных полиномов.

# Алгоритмы вычисления НОД в F/Z[x]

## Псевдо – деление, (– частное), (– остаток)

Рассмотрим два примитивных ненулевых полинома $p_1(x)$ и $p_2(x)$ в Z [x],
степени которых равны соответственно :

$\deg(p_1(x)) = m$ и $\deg(p_2(x)) = n$, $m > n$

Т.к. алгоритм деления полиномов с остатком
требует точной делимости старшего коэффициента делимого
на старший коэффициент делителя,
обычно этот процесс невозможно выполнить для полиномов $p_1(x)$ и $p_2(x)$
над целыми числами, не ослабляя требования делимости.

Поэтому введём процесс **псевдоделения**,
который всегда даёт **псевдочастное** и **псевдоостаток**,
коэффициенты которых являются целыми числами.

Псевдоделение означает предварительное умножение полинома $p_1(x)$ на $(lc(p_2(x)))^{m-n+1}$,
а затем применение алгоритма деления полиномов,
когда известно, что все частные существуют, т.е. :

$$(lc(p_2(x)))^{m-n+1} * p_1(x) = p_2(x) * q(x) + r(x) \qquad \deg(r(x)) < \deg(p_2(x))$$

где $lc(p(x))$ – это старший коэффициент ( leading coefficient ) полинома $p(x)$,
$q(x)$ – это псевдочастное , $r(x)$ – это псевдоостаток

# Алгоритмы вычисления НОД в $F/Z[x]$

## Пример применения псевдоделения в Z [x]

**Пример.**
Разделим полином $p_1(x) = x^4 - 7x + 7$ на полином $p_2(x) = 3x^2 - 7$
Очевидно, что $\deg(p_1(x)) = m = 4$ и $\deg(p_2(x)) = n = 2$
Далее, чтобы вычислить q (x) и r (x) ,
требуется предварительно умножить $p_1(x)$ на
$(lc(p_2(x)))^{m-n+1} = 3^{m-n+1} = 3^{4-2+1} = 3^3 = 27$
Теперь, применяя алгоритм деления полиномов, получим:
$q(x) = 9x^2 + 21$ $r(x) = -189x + 336$

**Замечание.** Легко убедиться в том, что алгоритм деления полиномов
не будет работать, если $p_1(x)$ предварительно умножить только на 3 ,
т.е. на старший коэффициент полинома $p_2(x)$ .
Таким образом, для вычисления НОД полиномов $p_1(x)$ и $p_2(x)$
НЕОБХОДИМО предварительно убедиться в том,
что ВСЕ ОПЕРАЦИИ ДЕЛЕНИЯ, встречающиеся в этом процессе ВЫПОЛНИМЫ
(иными словами, используя псевдоделения,
необходимо сформировать последовательность полиномиальных остатков).

# Алгоритмы вычисления НОД в $\mathbf{F/Z[x]}$

## Обобщённый алгоритм Евклида для полиномов над целыми числами

**Алгоритм 7.** ( **GEA-P – G**eneralized **E**uclidean **A**lgorithm for **P**olynomials over the Integers )

**Вход :**

$p_1(x)$ и $p_2(x)$ – ненулевые полиномы в **Z** [x],

$\deg(p_1(x)) = n_1$

$\deg(p_2(x)) = n_2$

$n_1 \geq n_2$

**Выход :**

НОД ( $p_1(x)$ , $p_2(x)$ )

**[ Вычисление НОД содержаний полиномов ]**

// Используя алгоритм Евклида для вычисления НОД двух целых чисел

c := НОД ( cont ( $p_1(x)$ ) , cont ( $p_2(x)$ ) )

**[ Вычисление примитивных частей ]**

$p'_1(x) := p_1(x) / \text{cont}(p_1(x))$

$p'_2(x) := p_2(x) / \text{cont}(p_2(x))$

**[ Построение PRS ]**

// Вычислить полиномиальные остатки : $p_3(x)$ , $p_4(x)$ , … , $p_h(x)$

**[ Формирование результата ]**

**Если** $\deg(p_h(x)) = 0$ **то** НОД ( $p_1(x)$ , $p_2(x)$ ) := c

**иначе** НОД ( $p_1(x)$ , $p_2(x)$ ) := c * p.p. ( $p_h(x)$ )

# Алгоритмы вычисления НОД в $F/Z[x]$

## Сложность обобщённого алгоритма Евклида

Временная сложность алгоритма **GEA-P** зависит от того,
насколько эффективно вычисляется последовательность полиномиальных остатков,
т.е. $p'_1(x)$ , $p'_2(x)$ , $p_3(x)$ , $p_4(x)$ , … , $p_h(x)$

**Замечание.** Если $n_i = \deg(p_i(x))$ , то в общем случае можно утверждать,
что члены PRS удовлетворяют следующим соотношениям:

$$(lc(p_{i+1}(x)))^{n_i - n_{i+1} + 1} * p_i(x) = p_{i+1}(x) * q_i(x) + \beta_i * p_{i+2}(x)$$

$$\deg(p_{i+2}(x)) < \deg(p_{i+1}(x))$$

где i = 1 , 2 , … , h-1 для некоторого h .

**Замечание.** Очевидно, что в указанных выше соотношениях :
$p_i(x) \equiv p'_i(x)$ при i = 1 и i = 2
(в свою очередь, $p'_i(x)$ при i = 1 и i = 2
вычислены на шаге **[ Вычисление примитивных частей ]** алгоритма **GEA-P**).

**Замечание.** Если определён метод выбора коэффициентов $\beta_i$ ,
то указанные выше соотношения дают алгоритм построения PRS .
Условием завершения такого алгоритма является равенство нулю псевдоостатка.

**Замечание.** Далее рассматриваются различные алгоритмы вычисления PRS ,
полученные для разных значений коэффициентов $\beta_i$ .

# Алгоритмы вычисления НОД в $\mathbf{F/Z[x]}$

## Евклидов алгоритм вычисления PRS

Пусть $\beta_i = 1$ для всех $i = 1, 2, \ldots, h-1$
( т.е. каждый псевдоостаток используется в том виде, в котором он получен ).
В результате получим один из худших алгоритмов вычисления PRS ,
который приводит к экспоненциальному росту коэффициентов.

**Пример.**
Рассмотрим полиномы $p_1(x) = x^3 - 7x + 7$ и $p_2(x) = 3x^2 - 7$ в $\mathbf{Z}[x]$ .
Очевидно, что $cont(p_1(x)) = cont(p_2(x)) = 1$

Имеем следующую последовательность :

$p_1(x) = x^3 - 7x + 7$
$p_2(x) = 3x^2 - 7$ $\qquad$ $q_1(x) = 3x$
$p_3(x) = -42x + 63$ $\qquad$ $q_2(x) = -126x - 189$
$p_4(x) = -441$ $\qquad$ $q_3(x) = 18522x - 27783$

полученную при выполнении следующих псевдоделений :

$(3)^2 * p_1(x) = p_2(x) * (3x) + (-42x + 63)$
$(-42)^2 * p_2(x) = p_3(x) * (-126x - 189) + (-441)$
$(-441)^2 * p_3(x) = p_4(x) * (18522x - 27783) + 0$

# Алгоритмы вычисления НОД в F/Z[x]

## Анализ примера евклидова алгоритма

Проанализируем некоторые важные особенности примера,
приведённого на предыдущем слайде.

Нетрудно видеть, что :

(1) т.к. cont ( $p_1(x)$ ) = cont ( $p_2(x)$ ) = 1 ,
то по алгоритму **GEA-P** c := НОД ( cont ( $p_1(x)$ ) , cont ( $p_2(x)$ ) ) = 1
и на последнем шаге ( при deg ($p_4(x)$ = – 441) = 0 ) имеем
НОД ( $p_1(x)$ , $p_2(x)$ ) = 1

(2) псевдоделение, выполняемое на последнем шаге,
наглядно показывает рост коэффициентов :
$(-441)^2 * p_3(x) = -8168202 * x + 12252303$
(коэффициент монома при $x^0$ имеет 8 десятичных цифр,
а коэффициенты в исходных полиномах – только 1 десятичную цифру).

(3) экспоненциальный рост коэффициентов членов PRS обусловлен тем,
что полиномы этой последовательности не являются примитивными
(иными словами, в процессе вычисления мы не избавляемся от их содержания,
что даёт вредный эффект).

# Алгоритмы вычисления НОД в $\mathbf{F/Z[x]}$

## Полные и неполные PRS

Последовательность полиномиальных остатков,
полученная в рассмотренном выше примере, называется **полной**,
т.к. степень каждого её члена
на единицу меньше степени предыдущего
(конечно, при этом два первых члена
могут иметь одинаковые степени).

В противном случае последовательность называется **неполной**.

**Замечание**. Не существует метода априорной оценки,
позволяющего определить, будет ли PRS полной или неполной.

# Алгоритмы вычисления НОД в $F/Z[x]$

## Алгоритм примитивных PRS

Пусть $\beta_i$ = cont ( prem ( $p_i(x)$ , $p_{i+1}(x)$ ) ) 1 для всех i = 1 , 2 , … , h-1
где prem обозначает псевдоостаток.
( т.е. мы удаляем содержание (i + 1) - го члена PRS до того, как используем его ).

Напомним, что для данного p (x) удобно определять p.p. ( p (x) ) так,
чтобы старший коэффициент был положительным.

**Пример.** (исходные данные те же, что и в предыдущем примере)
Рассмотрим полиномы $p_1(x) = x^3 - 7x + 7$ и $p_2(x) = 3x^2 - 7$ в **Z** [x] .
Имеем следующую последовательность :

$p_1(x) = x^3 - 7x + 7$
$p_2(x) = 3x^2 - 7$ $\quad q_1(x) = 3x$
$p_3(x) = 2x - 3$ $\quad q_2(x) = 6x + 9$ $\quad \beta_1 = -21$
$p_4(x) = 1$ $\quad q_2(x) = 2x - 3$ $\quad \beta_2 = -1$

полученную при выполнении следующих псевдоделений :

$(3)^2 * p_1(x) = p_2(x) * (3x) + (-21) * (2x - 3)$
$(2)^2 * p_2(x) = p_3(x) * (6x + 9) + (-1)$
$(1)^2 * p_3(x) = p_4(x) * (2x - 3) + 0$

# Алгоритмы вычисления НОД в $\mathbf{F/Z[x]}$

## Анализ примера алгоритма примитивных PRS

Из рассмотренного выше примера видно,
что алгоритм примитивных PRS эффективен настолько,
насколько этого можно добиться в отношении роста коэффициентов членов PRS.

Однако на каждом шаге требуется вычислять один или более НОД коэффициентов
и эти вычисления становятся всё более сложными по мере роста коэффициентов.

Таким образом, необходимо :

с одной стороны, найти способ избежать большей части сложных вычислений НОД;

с другой стороны, существенно уменьшить рост коэффициентов по сравнению с тем, который происходит в евклидовом алгоритме PRS.

**Замечание.** В настоящее время поиск эффективных алгоритмов вычисления PRS сосредоточен в двух (иногда взаимодополняющих) направлениях :
(1) Методы выбора коэффициентов $\beta_i$ .
(2) Методы псевдоделения редуцированных (субрезультантных) PRS.

# Алгоритмы вычисления НОД в F/Z[x]

## Матрица Сильвестра

Пусть даны два полинома $p_1(x)$ , $p_2(x) \in \mathbf{Z}[x]$ :
$p_1(x) = \sum ( a_i x^i ) , i = 0 .. n$
$p_2(x) = \sum ( b_i x^i ) , i = 0 .. m$

**Определение**. Матрицей **Джеймса Сильвестра** полиномов $p_1(x)$ и $p_2(x)$ называется следующая матрица, в которой имеется m строк, образованных из коэффициентов $a_i$
и n строк, образованных из коэффициентов $b_i$ :

$$\begin{pmatrix}
a_n & a_{n-1} & \dots & a_1 & a_0 & 0 & 0 & \dots & 0 \\
0 & a_n & a_{n-1} & \dots & a_1 & a_0 & 0 & \dots & 0 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
0 & \dots & 0 & 0 & a_n & a_{n-1} & \dots & a_1 & a_0 \\
b_m & b_{m-1} & \dots & b_1 & b_0 & 0 & 0 & \dots & 0 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
0 & \dots & 0 & 0 & b_m & b_{m-1} & \dots & b_1 & b_0
\end{pmatrix}$$

# Алгоритмы вычисления НОД в $\mathbf{F/Z[x]}$

## Результант полиномов

**Определение**. Определитель матрицы Сильвестра называется **результантом** полиномов $p_1(x)$ и $p_2(x)$, обозначаемым $\mathrm{Res}(p_1, p_2)$ или $\mathrm{Res}_x(p_1, p_2)$.

В силу известных свойств определителей :

(1) результант принадлежит кольцу $\mathbf{Z}[x]$

(2) $\mathrm{Res}(p_1, p_2)$ и $\mathrm{Res}(p_2, p_1)$ равны с точностью до знака

**Замечание**. Хотя результант задаётся как определитель,
это не лучший способ его вычисления, т.к. в кольцах целых чисел и полиномов операция деления может оказаться невыполнимой
или, даже если деление возможно, могут потребоваться вычисления с дробями.
**Эрвин Барейс** предложил такую модификацию метода исключения Гаусса (который для матрицы Сильвестра аналогичен алгоритму Евклида),
что каждое деление в кольце даёт результат из того же кольца, а не дробь.

# Алгоритмы вычисления НОД в $F/Z[x]$

## Алгоритм вычисления результанта

**Замечание**. Ниже приведён вариант алгоритма Евклида
для вычисления результанта. Алгоритм представлен в рекурсивной форме.
В алгоритме приняты следующие обозначения :

lc (p) – старший коэффициент полинома p (x);
deg (p) – степень полинома p (x);
rem (p , q) – остаток от деления полинома p (x) на q (x) .

**Алгоритм 8.**

**Вход** : $p_1(x)$ , $p_2(x)$ **Выход** : Res ( $p_2$ , $p_1$ )

n := deg ($p_1(x)$)
m := deg ($p_2(x)$)
**Если** n > m **то** r := $(-1)^{nm}$ * Res ( $p_2$ , $p_1$ )
**иначе** $a_n$ := lc ($p_1(x)$)
**Если** n = 0 **то** r := $(a_n)^m$
**иначе** h := rem ($p_2$ , $p_1$)
**Если** h = 0 **то** Res ( $p_2$ , $p_1$ ) := 0
**иначе** s := deg (h)
Res ( $p_2$ , $p_1$ ) := $(a_n)^{m-s}$ * Res ($p_1$ , h)

# Алгоритмы вычисления НОД в $\mathbf{F/Z[x]}$

## Субрезультант полиномов

**Определение**. Определитель матрицы, получающейся из матрицы Сильвестра вычёркиванием k первых и k последних столбцов, k первых и k последних строк, называется k-м **субрезультантом** полиномов $p_1(x)$ и $p_2(x)$, обозначаемым $\mathrm{Res}^{(k)}(p_1, p_2)$.

**Замечание.** В определённом смысле можно считать, что результант является нулевым субрезультантом.

**Замечание**. В настоящее время в компьютерной алгебре разработано несколько различных методов вычисления последовательностей полиномиальных остатков на основе субрезультантов. В частности, к ним относятся метод псевдоделения субрезультантных PRS, предложенный Сильвестром (1853) и дополненный **Вальтером Габихтом** (1948), а также метод матричной триангуляризации субрезультантных PRS, предложенный **Алкивиадисом Акритасом** (1986). Метод Акритаса существенно отличается от остальных методов тем, что он не зависит от выбора коэффициентов $\beta_i$.

# Алгоритмы вычисления НОД в $\mathbf{F/Z[x]}$

## Лучший метод выбора коэффициентов $\beta_i$

Пусть даны два полинома $p_1(x)$ , $p_2(x) \in \mathbf{Z}[x]$ .
Для вычисления их НОД построим последовательность полиномиальных остатков :
$p_3(x)$ , $p_4(x)$ , … , $p_s(x)$ , 0
Введём следующие обозначения :

$c_i$ – старший коэффициент полинома $p_i(x)$

$\delta_i$ – разность степеней полиномов $p_i(x)$ и $p_{i+1}(x)$

PRS строим по ранее рассмотренным формулам :

$$(lc(p_{i+1}(x)))^{n_i - n_{i+1} + 1} * p_i(x) = p_{i+1}(x) * q_i(x) + \beta_i * p_{i+2}(x)$$

$$\deg(p_{i+2}(x)) < \deg(p_{i+1}(x))$$

в которых полагаем :

$$\beta_1 = (-1)^{\delta_1 + 1}$$

$$\beta_i = -c_i * \psi_i^{\delta_i} , i > 1$$

$$\psi_1 = -1$$

$$\psi_i = (-c_i)^{\delta_{i-1}} * i * \psi_{i-1}^{1 - \delta_{i-1}} , i > 1$$

Тогда по теореме Лооса о субрезультантах (**Rüdiger Loos**, 1982)
все $p_i(x)$ являются полиномами с целыми коэффициентами,
и разрядность коэффициентов возрастает не более, чем линейно.

# Алгоритмы вычисления НОД в $F/Z[x]$

## Пример вычисления НОД – 1
## ( на основе п.п.с. – последовательности полиномиальных субрезультантов )

**Пример** (Стэнли Браун, 1971).
Пусть даны два полинома :

$p_1(x) = x^8 + x^6 - 3x^4 - 3x^3 + 8x^2 + 2x - 5$

$p_2(x) = 3x^6 + 5x^4 - 4x^2 - 9x + 21$

которые являются элементами кольца **Q** [x] .
Применяя алгоритм Евклида, получим следующую последовательность :

$p_3(x) = (-5/9)x^4 + (1/9)x^2 - (1/3)$

$p_4(x) = (-117/25)x^2 - 9x + (441/25)$

$p_5(x) = (233150/6591)x - (102500/2197)$

$p_6(x) = 1288744821/543589225$

**Замечание**. Все вышеуказанные дроби являются несократимыми.

# Алгоритмы вычисления НОД в $\mathbf{F/Z[x]}$

## Пример вычисления НОД – 2 ( на основе п.п.с. )

Применение нормализации делителя позволяет уменьшить коэффициенты, но не слишком сильно. В результате получим следующую евклидову последовательность полиномиальных остатков :

$p_3(x) = -15x^4 + 3x^2 - 9$ (итого 2 разряда)
$p_4(x) = 15795x^2 + 30375x - 59535$ (итого 5 разрядов)
$p_5(x) = 1254542875143750x - 1654608338437500$ (итого 16 разрядов)
$p_6(x) = 1259333879550074310093115199218750$ (итого 35 разрядов)

Наконец, применяя формулы лучшего метода выбора коэффициентов $\beta_i$, получим следующую последовательность :

$p_3(x) = 15x^4 - 3x^2 + 9$ (итого 2 разряда)
$p_4(x) = 65x^2 + 125x - 245$ (итого 3 разряда)
$p_5(x) = 9326x - 12300$ (итого 5 разрядов)
$p_6(x) = 260708$ (итого 6 разрядов)

**Замечание**. Очевидно, что рост коэффициентов PRS существенно замедлился.

# Спасибо за внимание !

Вопросы ?