НОД.
Расширенный алгоритм Евклида.

Поздняков Сергей Николаевич

запись конспекта: Пронин Валентин
дата лекции: 12.03.2018

6:00

1. Делимость целых чисел

Важно понимать, что деление и делимость - это два разных понятия: операция и отношение. В первом случае при делении чисел a и b в результате получаем число.

$$a \ / \ b \longrightarrow \text{число}$$

Во втором случае результатом будет являться не число, а логическое значение "да"или "нет"(отношение делимости описывается словами "a делится на b или "b делит a").

$$\begin{matrix} a \vdots b \\ b \mid a \end{matrix} \longrightarrow \begin{cases} \text{да} \\ \text{нет} \end{cases}$$

7:30

Определение. Целое число a делится на число b $(b \neq 0) \overset{def}{\Longleftrightarrow}$ остаток от деления a на b равен 0.

Это определение можно свести к другому:

$$a = b \cdot q + r, \quad 0 \leq r < |b|$$

, тогда

$$r = 0 \ \Leftrightarrow \ \underline{a = b \cdot q}$$

Поэтому говорят, что a делится на b, если существует такое целое число q, что a равняется $b \cdot q$

Свойства делимости: 9:50

1) $a \vdots d,\ b \vdots d \to a - k \cdot b \vdots d$

Примечание. $a, b, d, k \in \mathbb{Z}$

10:20

Доказательство.

$$\left.\begin{array}{rcl} a \vdots d & \Leftrightarrow & \text{сущ. } q_1 : \ a = \ q_1 \cdot d \\ b \vdots d & \Leftrightarrow & \text{сущ. } q_2 : \ b = \ q_2 \cdot d \end{array}\right\} \Rightarrow$$

$$a - kb = q_1 \cdot d - k \cdot q_2 \cdot d = \underbrace{(q_1 - kq_2)}_{q} \cdot d = q \cdot d \ \Leftrightarrow \ a - kb \vdots d$$

2) $a \vdots b, b \vdots c \to a \vdots c$ (транзитивность делимости)

12:30

Доказательство. Необходимо привести самим. □

3) $a \vdots b, b \vdots a \to |a| = |b|$

12:50

Доказательство. Необходимо привести самим. □

13:50

2. Наибольший общий делитель (НОД)

Допустим, что у нас есть набор целых неотрицательных чисел $\{a_1, a_2, \ldots, a_n\}$, при этом не все числа в наборе равны нулю.

Обратимся к первому свойству делимости. Назовем вычитание из одного целого числа (a) другого целого (b), умноженного на некоторый целый коэффициент (k), элементарной операцией. Первое свойство делимости говорит нам о том, что если целые числа a и b имели какой-то общий делитель, то и числа полученные в результате элементарной операции ($a - kb$ и b) также будут иметь этот общий делитель.

Вернемся к нашему набору, над которым мы будем совершать такие элементарные операции: брать два ненулевых числа и из большего вычитать меньшее, умноженное на k. Число k будем брать максимальным, при котором результат $a - k \cdot b$ не будет отрицательным, Благодаря этому числа набора будут уменьшаться максимально быстро.

Данный алгоритм закончит свою работу, когда в наборе не найдется двух ненулевых чисел, то есть все числа кроме одного будут равны нулю. Заметим, что делители чисел набора сохранятся после всех этих преобразований. Поэтому когда останется одно ненулевое число, то оно и будет наибольшим общим делителем набора.

Таким образом, следующий алгоритм находит наибольший общий делитель данного набора чисел. 18:20

Ниже представлено несколько алгоритмов, отличающихся правилом выбора пары чисел.

Algorithm 1 Алгоритм нахождение НОД

1: **while** в наборе есть два ненулевых числа **do**
2: Выбрать $a_i \geq a_j > 0$
3: $a_i := a_j - ka_j$, k - такое, что результат неотрицательный
4: **end while**
5: $d := max\{a_1, \ldots, a_n\} = \{0, \ldots, d, \ldots, 0\}$

Примечание. Если, как говорилось выше, выбирать k максимальным, при котором результат вычитания остается неотрицательным, алгоритм можно оптимизировать, заменив "$a_i := a_j - ka_j$, k - такое, что результат неотрицательный" на "a_i : заменить остатком от деления на a_j". 20:35

Инварианты цикла алгоритма нахождение НОД

1) *Все числа неотрицательны.*

Доказательство. Алгоритм не изменяет это свойство, потому что мы всегда искуственно его соблюдаем, когда заменяем одно число на другое неотрицательное. □ 21:00

2) *Множество делителей не меняется.*

$$D = D'$$

D - множество делителей исходного набора чисел до начала выполнения операций в теле цикла. D' - множество делителей набора после их выполнения. 22:00

Доказательство. Изначально у нас был набор

$$a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n$$

после одного шага цикла у нас получился модифицированный набор,

$$a'_1 = a_1, \ldots, a'_i = a_i - ka_j, \ldots, a'_j = a_j, \ldots, a'_n = a_n$$

в котором поменялся только $a'_i = a_i - ka_j$. Если у чисел исходного набора был какой-то делитель, то у всех чисел нового набора, которые не изменились, он же и останется. Необходимо доказать, что он будет и у числа a'_i.

Пусть d - делитель исходного набора, то есть

$$a_i \vdots d, a_j \vdots d \to a'_i = a_i - ka_j \vdots d$$

по первому свойству делимости. □

24:30

Так мы доказали, что любой делитель исходного набора будет также делителем измененного набора.

Далее нужно доказать что при выполнении элементарного преобразования не добавляются новые делители. 25:05

Доказательство. Пусть d - делитель модифицированного набора; докажем, что он будет делителем исходного.

Проведем рассуждения в обратной последовательности. Пусть у чисел модифицированного набора есть общий делитель d. Число a'_i модифицированного набора равно $a_i - ka_j$ в обозначениях исходного, а число a'_j модифицированного набора совпадает с числом a_j исходного. Поэтому из $a'_i = a_i - ka_j$ получаем $a'_i = a_i - ka'_j$ или, что то же самое, $a_i = a'_i + ka'_j$ которое будет иметь число d делителем по первому свойству делимости. Остальные числа у модифицированного и у исходного набора одинаковы. Поэтому любой делитель модифицированного набора будет также делителем исходного. □

32:30

Замечание. Для того, чтобы определить наибольший общий делитель чисел можно взять все общие делители чисел и выбрать из них наибольший. Однако при этом у нас кроме отношения делимости возникает потребность использования ещё одного отношения "больше-меньше". В следующем семестре мы будем изучать автоматическое доказательство теорем. Увидим, что трудоемкость этих алгоритмов велика. Поэтому при формулировке теорем нужно использовать как можно меньше различных сущностей (этот принцип достаточно общий и называется бритвой Оккама) т..к. их количество экспоненциально влияет на количество вариантов, которые программа должна будет перебрать, чтобы доказать теорему. Более выгодно будет определить наибольший общий делитель как тот, который делится на все остальные общие делители чисел, чтобы избавиться от отношения "больше-меньше".

Если вернуться к нашему алгоритму, то можно увидеть, что из всего множества делителей у нас осталось только одно число, которое делится на все остальные делители набора чисел. То есть все делители набора являются делителями наибольшего общего делителя. 35:40

Для нахождения еще одного инварианта цикла алгоритма 1 обратимся к следующей задаче. 36:00

Задача. Даны числа: 12 и 9. Какое множество получится, если выполнять над ними операции сложения и вычитания в произвольном количестве?

Ответ: числа, кратные трём.

Множество таких чисел можно записать как:

$$M = \{12x + 9y \mid x, y \in \mathbb{Z}\} = \{3d \mid d \in \mathbb{Z}\}$$

Если рассмотреть множество комбинаций линейного набора, то окажется, что оно тоже не меняется при элементарных преобразованиях. 38:50

3) Множество линейных комбинаций L набора чисел не будет меняться при элементарных преобразованиях.

   $L = \{a_1x_1, + \ldots + a_nx_n \text{ где } x_1, \ldots, x_n \in \mathbb{Z}\}$

$$L' = L$$

Примечание. L - множество линейных комбинаций. L' - аналог L, только в процессе работы алгоритма.

Доказательство. Необходимо привести самим. □ 41:15

Исходя из данного инварианта, можно сделать вывод о том, что линейная кобминация набора $(a_1, \ldots, a_n)$ будет равняться линейной комбинации, состоящей из нулей и одного наибольшего делителя.

$$L(a_1, \ldots a_n) = L(0, \ldots d, \ldots, 0)$$

Если эти множества равны, значит для каждого элемента одного из множеств можно найти соответствующее представление в другом. Если в правом множестве есть d, значит в левом множестве найдутся такие коэффициенты, что будет выполнено следующее равенство

$$a_1x_1 + \ldots + a_nx_n = d$$

43:05

Теорема (о линейном представлении НОД). *Если* $d = НОД(a_1, \ldots, a_n)$, *то найдутся такие целые коэффициенты* $x_1, \ldots, x_n$, *что* $a_1x_1 + \ldots a_nx_n = d$.

47:15

Рассмотрим детализацию работы не самого эффективного алгоритма 2 для набора из двух чисел, построенного на основе 1 для $k = 1$.

Пример.

$a = 34$	$b = 12$
34	12
22	12
10	12
10	2
8	2
6	2
4	2
2	2
0	2

48:50

Algorithm 2 Алгоритм нахождение НОД

```
1: while a ≠ 0 and b ≠ 0 do
2:     if a ≥ b then
3:         a := a − b
4:     end if
5:     if a < b then
6:         b := b − a
7:     end if
8: end while
9: d := max{a; b}
```

Данный алгоритм можно усовершенствовать, добавив циклы, в которых мы сразу получаем остаток от деления большего числа на меньшее.

Algorithm 3 Улучшенный алгоритм нахождение НОД

```
1: while a ≠ 0 and b ≠ 0 do
2:     while a ≥ b do
3:         a := a − b
4:     end while
5:     while a < b do
6:         b := b − a
7:     end while
8: end while
9: d := max{a; b}
```

Пример.

$a = 34$	$b = 12$
34	12
22	12
10	12
10	2
8	2
6	2
4	2
2	2
0	2

Заменив циклы, которые позволяют нам находить остаток от деления одного числа на другое повторяющимся вычитанием, на вычисление остатка (операция mod) мы получим алгоритм Евклида. 52:30

Algorithm 4 Алгоритм Евклида

```
1: while a ≠ 0 and b ≠ 0 do
2:     if a ≥ b then
3:         a := a mod b
4:     end if
5:     if a < b then
6:         b := b mod a
7:     end if
8: end while
9: d := max{a; b}
```

55:05

Пример. Работу данного алгоритма удобно представлять в виде таблицы.

	34	12	10	2	0
			2	1	5
1 шаг:	a	b	r		
2 шаг:		b	a	r	
3 шаг:			a	b	r
4 шаг:				b	a

Когда цикл завершается, мы выбираем наибольшее из a и b. Это число и будет НОД $(a; b)$. 1:00:10

3. Расширенный алгоритм Евклида

Задача. Как налить один литр в бочку неограниченных размеров с помощью двух вёдер объемами 17 и 12 литров, если количество воды не ограничено?

Примечание. Вливать ведра объемом 17 и 12 литров друг в друга нельзя, а зачерпывать можно только полные ведра.

Договоримся, что черпать будем полными ведрами и из ведра в ведро не переливать, хотя в дальнейшем можно заметить, что последнее условие несущественно. Заметим, что если мы налили воду, например, ведром 12 литров, а через некоторое время этим же ведром отлили 12 литров, то эти две операции компенсировали друг друга и их можно просто не делать. Таким образом, можно считать, что одним ведром мы будем только наливать воду, а другим только отливать.

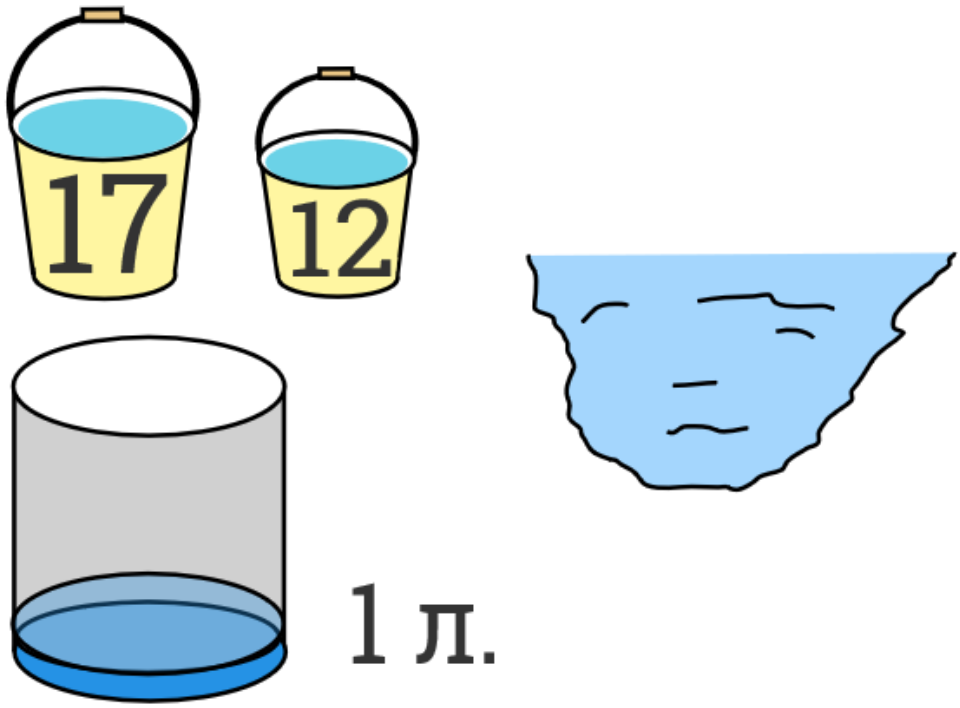
Задача сведется к уравнению вида

$$17x + 12y = 1$$

где знаки перед x и y зависят от того, наливаем мы или выливаем какое-то количество вёдер.

Вернемся к инварианту, связанному с линейными комбинациями. Левую часть уравнения можно рассматривать как линейную комбинацию 17 и 12. Нас интересуют коэффициенты, при которых она равно единице.

Отсюда может возникнуть мысль использовать алгоритм Евклида для решения данной задачи. 1:04:40



Для начала представим наши "ведра"в виде векторов:

$$A = (1, 0) = 1 \cdot 17 + 0 \cdot 12 = 17$$
$$B = (0, 1) = 0 \cdot 17 + 1 \cdot 12 = 12$$

Фактически это линейные комбинации.

Перейдем к решению нашего уравнения через алгоритм нахождение НОД. Слева решение с числами, справа же с векторами. Мысленно выполните действия алгоритма 2 сначала над числами, а потом над представляющими их векторами. Обратите внимание, что алгоритм 3 для этого не годится, так как в нем есть деление, а операция деления над векторами не определена.

17	12
5	12
5	2
1	2

(1;0)	(0;1)
(1;-1)	(0;1)
(1;-1)	(-2;3)
(5;-7)	(-2;3)

Остановимся, когда достигнем нужного нам числа 1, ему соответствует вектор $(5; -7)$. Если подставить координаты данного вектора в

изначальное уравнение вместо x и y, то получится:

$$17 \cdot 5 + (-7) \cdot 12 = 85 - 84 = 1$$

то есть чтобы получить 1 литр в бочке, нам нужно 7 раз налить в него воду ведром объемом 17 литров и 5 раз вычерпнуть из него воду ведром объемом 12 литров. 1:08:50

Выполненные выше действия удобно записывать в форме протокола в горизонтальной таблице

17	12	5	2	1	0
		1	2	2	2
1	0	1	-2	5	-12
0	1	-1	3	-7	17

Примечание. Первые два числа в первой строке - это коэфициенты перед x и y в нашем уравнении. Дальше в этой строке идут остатки после элементарных преобразований.

Во второй строке стоят неполные частные.

Третья и четвертая строка предназначены для представления чисел в виде векторов.

Вектор $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ в первом столбце таблицы обозначает, что для получения числа 17 нам необходимо 1 раз влить ведро объемом 17 литров и 0 раз влить ведро объемом 12 литров. Аналогично можно интерпретировать и другие вектора, ставшие частью столбиков таблицы.

Модифицируем алгоритм Евклида, добавив в него вектора, а также упростив его, заменяя после каждого шага результаты так, чтобы всегда делить a на b. На первый взгляд кажется, что это создаст проблему деления меньшего числа на большее, однако это добавит лишь один лишний шаг. Действительно, если $a < b$, то при делении неполное частное будет равно нулю, а остаток совпадет с делимым (a). После этого a и b обменяются местами и далее всегда большее число a будет делиться на меньшее b. 1:12:30

4. Бинарный алгоритм Евклида

В заключение, рассмотрим еще один алгоритм, который выгодно применять для нахождения НОД чисел в двоичной записи. 1:18:30

Algorithm 5 Алгоритм Евклида

Инициализация:

1: $(x_a; y_a) := (1; 0);\ (x_b; y_b) := (0; 1)$

2: **while** $a \neq 0$ и $b \neq 0$ **do**

3: $q := a$ div b

4: $r := a - qb$; $(x_r; y_r) := (x_a; y_a) - q \cdot (x_b; y_b)$

5: $a := b$; $(x_a; y_a) := (x_b; y_b)$

6: $b := r$; $(x_b; y_b) := (x_r; y_r)$

7: **end while**

Ответ:$(x_a; y_a)$.

Algorithm 6 Бинарный алгоритм Евклида

Инициализация:

1: $d := 1$ ▷ d - переменная для хранения НОД

2: **while** ($a \vdots 2$ and $b \vdots 2$) **do**

3: $b := b/2; \quad a := a/2; \quad d := d \cdot 2$

4: **end while** ▷ Данный цикл сдвигает числа a и b вправо, избавляя их от нулей в конце. В это же время он сдвигает d влево.

▷ После окончания предыдущего цикла, обязательно одно число будет делиться на 2, а другое нет. Однако если одно делится, а второе не делится, то двойки из его разложения на простые множители можно выбросить, т.к. они уже не будут общими делителями. Это достигается делением этого числа на 2 или сдвигами его двоичной записи вправо, что делается в следующем цикле.

5: **while** ($a \neq 0$ и $b \neq 0$) **do** ▷ Данное условие позволяет нам остановится, когда найдется НОД

6: **while** $a \vdots 2$ **do**

7: $a := a/2$

8: **end while**

9: **while** $b \vdots 2$ **do**

10: $b := b/2$

11: **end while**

▷ Когда останется два нечетных числа, происходит вычитание меньшего из большего, чтобы получить четное число, которое можно будет опять делить на два.

12: **if** $a \leq b$ **then**

13: $a := a - b$

14: **else**

15: $b := b - a$

16: **end if**

17: **end while**

18: $d := d \cdot max\{a; b\}$