MAXimal

home
algo
bookz
forum
about

добавлено: 10 Jun 2008 18:15
редактировано: 26 Apr 2012 1:46

Линейные диофантовы уравнения с двумя переменными

Содержание [скрыть]

Диофантово уравнение с двумя неизвестными имеет вид:

$$a \cdot x + b \cdot y = c,$$

где a, b, c — заданные целые числа, x и y — неизвестные целые числа.

Ниже рассматриваются несколько классических задач на эти уравнения: нахождение любого решения, получение всех решений, нахождение количества решений и сами решения в определённом отрезке, нахождение решения с наименьшей суммой неизвестных.

Вырожденный случай

Один вырожденный случай мы сразу исключим из рассмотрения: когда $a = b = 0$. В этом случае, понятно, уравнение имеет либо бесконечно много произвольных решений, либо же не имеет решений вовсе (в зависимости от того, $c = 0$ или нет).

Нахождение одного решения

Найти одно из решений диофантова уравнения с двумя неизвестными можно с помощью Расширенного алгоритма Евклида. Предположим сначала, что числа a и b неотрицательны.

Расширенный алгоритм Евклида по заданным неотрицательным числам a и b находит их наибольший общий делитель g, а также такие коэффициенты x_g и y_g, что:

$$a \cdot x_g + b \cdot y_g = g.$$

Утверждается, что если c делится на $g = \gcd(a, b)$, то диофантово уравнение $a \cdot x + b \cdot y = c$ имеет решение; в противном случае диофантово уравнение решений не имеет. Доказательство следует из очевидного факта, что линейная комбинация двух чисел по-прежнему должна делиться на их общий делитель.

Предположим, что c делится на g, тогда, очевидно, выполняется:

$$a \cdot x_g \cdot (c/g) + b \cdot y_g \cdot (c/g) = c,$$

т.е. одним из решений диофантова уравнения являются числа:

$$\begin{cases} x_0 = x_g \cdot (c/g), \\ y_0 = y_g \cdot (c/g). \end{cases}$$

Мы описали решение в случае, когда числа a и b неотрицательны. Если же одно из них или они оба отрицательны, то можно поступить таким образом: взять их по модулю и применить к ним алгоритм Евклида, как было описано выше, а затем изменить знак найденных x_0 и y_0 в соответствии с настоящим знаком чисел a и b соответственно.

Реализация (напомним, здесь мы считаем, что входные данные $a = b = 0$ недопустимы):

```
int gcd (int a, int b, int & x, int & y) {
	if (a == 0) {
		x = 0; y = 1;
		return b;
	}
	int x1, y1;
	int d = gcd (b%a, a, x1, y1);
	x = y1 - (b / a) * x1;
	y = x1;
	return d;
}

bool find_any_solution (int a, int b, int c, int & x0, int & y0, int & g) {
	g = gcd (abs(a), abs(b), x0, y0);
	if (c % g != 0)
		return false;
```

```
	x0 *= c / g;
	y0 *= c / g;
	if (a < 0)   x0 *= -1;
	if (b < 0)   y0 *= -1;
	return true;
}
```

Получение всех решений

Покажем, как получить все остальные решения (а их бесконечное множество) диофантова уравнения, зная одно из решений (x_0, y_0).

Итак, пусть $g = \gcd(a, b)$, а числа x_0, y_0 удовлетворяют условию:

$$a \cdot x_0 + b \cdot y_0 = c.$$

Тогда заметим, что, прибавив к x_0 число b/g и одновременно отняв a/g от y_0, мы не нарушим равенства:

$$a \cdot (x_0 + b/g) + b \cdot (y_0 - a/g) = a \cdot x_0 + b \cdot y_0 + a \cdot b/g - b \cdot a/g = c.$$

Очевидно, что этот процесс можно повторять сколько угодно, т.е. все числа вида:

$$\begin{cases} x = x_0 + k \cdot b/g, \\ y = y_0 - k \cdot a/g, \end{cases} \quad k \in Z$$

являются решениями диофантова уравнения.

Более того, только числа такого вида и являются решениями, т.е. мы описали множество всех решений диофантова уравнения (оно получилось бесконечным, если не наложено дополнительных условий).

Нахождение количества решений и сами решения в заданном отрезке

Пусть даны два отрезка $[min_x; max_x]$ и $[min_y; max_y]$, и требуется найти количество решений (x, y) диофантова уравнения, лежащих в данных отрезках соответственно.

Заметим, что если одно из чисел a, b равно нулю, то задача имеет не больше одного решения, поэтому эти случаи мы в данном разделе исключаем из рассмотрения.

Сначала найдём решение с минимальным подходящим x, т.е. $x \ge min_x$. Для этого сначала найдём любое решение диофантова уравнения (см. пункт 1). Затем получим из него решение с наименьшим $x \ge min_x$ — для этого воспользуемся процедурой, описанной в предыдущем пункте, и будем уменьшать/увеличивать x, пока оно не окажется $\ge min_x$, и при этом минимальным. Это можно сделать за $O(1)$, посчитав, с каким коэффициентом нужно применить это преобразование, чтобы получить минимальное число, большее либо равное min_x. Обозначим найденный x через $lx1$.

Аналогичным образом можно найти и решение с максимальным подходящим $x = rx1$, т.е. $x \le max_x$.

Далее перейдём к удовлетворению ограничений на y, т.е. к рассмотрению отрезка $[min_y; max_y]$. Способом, описанным выше, найдём решение с минимальным $y \ge min_y$, а также решение с максимальным $y \le max_y$. Обозначим x-коэффициенты этих решений через $lx2$ и $rx2$ соответственно.

Пересечём отрезки $[lx1; rx1]$ и $[lx2; rx2]$; обозначим получившийся отрезок через $[lx; rx]$. Утверждается, что любое решение, у которого x-коэффициент лежит в $[lx; rx]$ — любое такое решение является подходящим. (Это верно в силу построения этого отрезка: сначала мы отдельно удовлетворили ограничения на x и y, получив два отрезка, а затем пересекли их, получив область, в которой удовлетворяются оба условия.)

Таким образом, количество решений будет равняться длине этого отрезка, делённой на $|b|$ (поскольку x-коэффициент может изменяться только на $\pm b$), и плюс один.

Приведём реализацию (она получилась достаточно сложной, поскольку требуется аккуратно рассматривать случаи положительных и отрицательных коэффициентов a и b):

```
void shift_solution (int & x, int & y, int a, int b, int cnt) {
	x += cnt * b;
	y -= cnt * a;
}

int find_all_solutions (int a, int b, int c, int minx, int maxx, int miny, int maxy) {
	int x, y, g;
	if (! find_any_solution (a, b, c, x, y, g))
		return 0;
	a /= g;  b /= g;

	int sign_a = a>0 ? +1 : -1;
	int sign_b = b>0 ? +1 : -1;
```

```
	shift_solution (x, y, a, b, (minx - x) / b);
	if (x < minx)
		shift_solution (x, y, a, b, sign_b);
	if (x > maxx)
		return 0;
	int lx1 = x;

	shift_solution (x, y, a, b, (maxx - x) / b);
	if (x > maxx)
		shift_solution (x, y, a, b, -sign_b);
	int rx1 = x;

	shift_solution (x, y, a, b, - (miny - y) / a);
	if (y < miny)
		shift_solution (x, y, a, b, -sign_a);
	if (y > maxy)
		return 0;
	int lx2 = x;

	shift_solution (x, y, a, b, - (maxy - y) / a);
	if (y > maxy)
		shift_solution (x, y, a, b, sign_a);
	int rx2 = x;

	if (lx2 > rx2)
		swap (lx2, rx2);
	int lx = max (lx1, lx2);
	int rx = min (rx1, rx2);

	return (rx - lx) / abs(b) + 1;
}
```

Также нетрудно добавить к этой реализации вывод всех найденных решений: для этого достаточно перебрать x в отрезке $[lx;rx]$ с шагом $|b|$, найдя для каждого из них соответствующий y непосредственно из уравнения $ax+by=c$.

Нахождение решения в заданном отрезке с наименьшей суммой x+y

Здесь на x и на y также должны быть наложены какие-либо ограничения, иначе ответом практически всегда будет минус бесконечность.

Идея решения такая же, как и в предыдущем пункте: сначала находим любое решение диофантова уравнения, а затем, применяя описанную в предыдущем пункте процедуру, придём к наилучшему решению.

Действительно, мы имеем право выполнить следующее преобразование (см. предыдущий пункт):

$$\begin{cases} x' = x + k \cdot (b/g), \\ y' = y - k \cdot (a/g), \end{cases} \quad k \in Z.$$

Заметим, что при этом сумма $x+y$ меняется следующим образом:

$$x' + y' = x + y + k \cdot (b/g - a/g) = x + y + k \cdot (b-a)/g.$$

Т.е. если $a<b$, то нужно выбрать как можно меньшее значение k, если $a>b$, то нужно выбрать как можно большее значение k.

Если $a=b$, то мы никак не сможем улучшить решение, — все решения будут обладать одной и той же суммой.

Задачи в online judges

Список задач, которые можно сдать на тему диофантовых уравнений с двумя неизвестными:

- SGU #106 **"The Equation"** [сложность: средняя]