Математические основы информатики

Нецелочисленная арифметика.

Сергей Леонидович Бабичев

Вещественные числа в информатике

Компьютерное представление чисел

- Компьютерное представление целых чисел формирует конечное счётное множество. Отрицательные числа кодируются в дополнительном коде.
- Все операции над целыми числами производятся в модульной арифметике по соответствующим степеням двойки.
- Вещественные числа образуют континуум. Однако нам доступно конечное множество для их представления.
- Требуется *аппроксимация* — каждое представимое вещественное число представляется либо точным образом, либо с какой-то *погрешностью*.

Представление вещественных чисел

- Перевод целой части вещественного числа в двоичное представление прост.
- Перевод дробной части в двоичное представление немного сложнее.
- Основная идея в том, что если дробь 0.1_2 равна 0.5_{10}, то все числа, большие или равные 0.5_{10} будут больше или равны 0.1_2, то есть иметь единицу в первом разряде после запятой.
- Очередная двоичная цифра правильной дроби $\frac{p}{q}$ есть $\left\lfloor \frac{2p}{q} \right\rfloor$
- Далее берётся дробная часть $\left\{ \frac{2p}{q} \right\}$ и, если она не равна нулю, процесс повторяется.
- Для q не являющегося степенью двойки, процесс продолжается вечно.

Перевод дроби в двоичное представление

0.263	$0\ (0.263 < 1)$
$0.263 \cdot 2 = 0.526$	$0\ (0.526 < 1)$
$0.526 \cdot 2 = 1.052$	$1\ (1.052 \geqslant 1)$
$0.052 \cdot 2 = 0.104$	$0\ (0.104 < 1)$
$0.104 \cdot 2 = 0.208$	$0\ (0.208 < 1)$
$0.208 \cdot 2 = 0.416$	$0\ (0.416 < 1)$
$0.416 \cdot 2 = 0.832$	$0\ (0.832 < 1)$
$0.832 \cdot 2 = 1.664$	$1\ (1.664 \geqslant 1)$
$0.664 \cdot 2 = 1.328$	$1\ (1.328 \geqslant 1)$
$0.328 \cdot 2 = 0.656$	$0\ (0.656 < 1)$
...	...

$$0.263_{10} \approx 0.010000110..._2$$

Представление вещественных чисел

- Вещественные числа имеют ограниченную информационную ёмкость и, представляются двоичными цифрами. Любая несократимая дробь, знаменатель которой не является степенью двойки, будет представляться в виде периодической двоичной дроби.

$1/3$	0
$1/3 \cdot 2 = 2/3$	0
$2/3 \cdot 2 = 4/3$	1
$1/3 \cdot 2 = 2/3$	0
...	...

$$\frac{1}{3} = 0.0(01)$$

- Вычисления с дробями в общем случае будут давать приближённые результаты, в частности,

$$\frac{1}{3} \times 3 \neq 1.$$

Это — фундаментальное свойство всех компьютерных представлений чисел: либо мы обеспечиваем абсолютную погрешность, либо относительную. Третьего не дано.

Представление вещественных чисел

- Обычно вещественные числа представляются в виде

$$R = M \times 2^e.$$

- Существуют реализации, где используются другие основания: 10, 16.
- Количество разрядов конечно.
- Поэтому общий вид представления такой:

$$R = \pm(d_0 + d_1\beta^{-1} + \cdots + d_{p-1}\beta^{-(p-1)})\beta^e,$$

где β — основание системы счисления представления, d_i — цифры , p — точность и e — экспонента.

- Число 0.1_{10} при $\beta = 10$ и $p = 3$ представляется точно как 1.00×10^{-1}, а при $\beta = 2$ и $p = 24$ приближённо как $1.10011001100110011001101 \times 2^{-4}$.
- Для каждого представления имеются наибольшая e_{max} и наименьшая e_{min} возможные экспоненты.

Приближённое представление

- Точное значение числа 0.1 лежит между двумя представимыми числами но не равно ни одному из них.
- Имеются числа, непредставимые в экспоненциальном представлении, бóльшие, чем $\beta \times \beta^{e_{max}}$ и меньшие $\beta^{e_{min}}$.
- Экспоненциальных представлений теоретически может быть бесконечное число ($5 = 5 \times 2^0 = 2.5 \times 2^1 = 1.25 \times 2^2 \dots$), поэтому используется *нормализованное* представление.
- В *нормализованном* представлении $1 \leqslant M <$ и представление оказывается единственным.
- Однако, нормализованное представление не может представить нуль.

Ошибки представления

- Ошибки округления неизбежны в вещественных числах.
- Требуется понять, ка их учитывать.
- Для $\beta = 10$ и $p = 4$ число 0.123 представляется точно.
- Число 0.123456 представляется как 1.235×10^{-1} и отличается от заданного на 0.44 от значения последней цифры.
- При $R = d.d...d \times \beta^e$ ошибка представления есть $|d.d...d - (R/b^e)| \times \beta^{p-1}$.
- Сократим термин «значение последней цифры», last digit value, как *LDV*.
- Другой способ — определить *относительную погрешность*

$$\frac{|0.123456 - 0.1235|}{0.123456} \approx 0.00035$$

.

Ошибки представления

- Относительные ошибки могут достигать $0.5LDV$
- Это соответствует числу $0.00..00\beta' \times \beta^e$, где β' есть цифра $\beta/2$.
- Это выражение равняется $((\beta/2)\beta^{-p}) \times \beta^e$.
- Все числа вида $d.dd..dd \times \beta^e$ имеют одинаковую абсолютную ошибку.
- Относительная ошибка лежит между

$$\frac{1}{2}\beta^{-p} \leqslant \frac{1}{2}LDV \leqslant \frac{\beta}{2}\beta^{-p}$$

- Обозначим $\varepsilon = \frac{\beta}{2}\beta^{-p}$ — верхий предел ошибки.

Ошибки представления

- Для числа $x = 12.35$, $\beta = 10$, $p = 3$ $\tilde{x} = 1.24 \times 10^1$
- Ошибка равна $0.5LDV$, относительная ошибка равна 0.8ε.
- При умножении $\tilde{x}$ на 8 результат $8\tilde{x} = 9.92 \times 10^1$ при точном результате $8x = 98.8$.
- Абсолютная ошибка стала $4LDV$, в 8 раз больше.
- Относительная ошибка осталась 0.8ε.

Защитные цифры

- Как найти разницу двух вещественных чисел?
- Можно вычислить точно и затем округлить?
- Для $p = 3, \beta = 10$ возьмём $x = 1.23 \times 10^5$, $y = 5.67 \times 10^{-3}$.
- Их разница равна

$$x - y = 123000 - 0.00567 = 122999.00433 \approx 1.23 \times 10^5.$$

- При выравнивании чисел по единой экспоненте

$$x = 1.23 \times 105, y = 0.00 \times 10^5, x - y = 1.23 \times 10^5.$$

- Здесь ответ совпал. Всегда ли?

Защитные цифры

- Пусть $x = 10.1, y = 9.92$.
- Тогда $x - y = 0.18$.
- Выровняем $y = 0.99 \times 10^1$, $x - y = (1.01 - 0.99) \times 10^1 = 0.02 \times 10^1$.

Theorem

При использовании экспоненциального представления с параметрами β и p при вычислении разности относительная погрешность результата может достигать $\beta - 1$.

Доказательство.

Предъявим случай, при котором это происходит. $x = 1.00.0$ и $y = 0.\gamma\gamma..\gamma$, где $\gamma = \beta - 1$. Точная разность $x - y = \beta - p$. При использовании ровно p разрядов самый правый разряд y сдвигается и исчезает. Абсолютная ошибка равна $\beta^{-p} - \beta^{-p+1} = \beta^{-p}(\beta - 1)$. Относительная ошибка равна $\beta - 1$. □

Защитные цифры

- При $\beta = 2$ относительная ошибка может достигать результата.
- Будем добавлять одну *защитную* цифру.
- Теперь меньший операнд будет содержать $p + 1$ цифру.
- $x = 10.1$, $y = 9.92$.
- $x = 1.010 \times 10^1$.
- $y = 0.992 \times 10^1$.
- $x - y = 0.018 \times 10^1$.

Защитные цифры

Theorem (о защитной цифре)

Для чисел x и y в экспоненциальном представлении с параметрами β и p относительная ошибка округления при вычитании не превосходит 2ϵ при условии использования $p+1$ цифр (одной защитной цифры).

Уничтожение цифр

- Другая проблема при вычитании — уменьшение количества значащих цифр при операции над примерно равными числами.
- Она называется *уничтожение цифр*.
- $1.2346567 - 1.2345678 = 0.0000889$.
- Уничтожение цифр может быть *жёстким* и *мягким*.

Жёсткое уничтожение цифр

- При решении квадратного уравнения нужно вычислить $b^2 - 4ac$.
- Оба операнда — результат операций умножения.
- Предположим, что они имеют точность $0.5LDV$.
- При вычитании значащие цифры могут уничтожиться и ошибка LDV возрастёт во много раз.
- Пусть $b = 3.34, a = 1.22$ и $c = 2.28$. Точное значение $b^2 - 4ac = 0.0292$. $b^2 \approx 11.2$, $4ac \approx 11.1$, $b^2 - 4ac \approx 0.1$.

Мягкое уничтожение цифр

- Мягкое уничтожение цифр возникает при вычитании точно известных значений. При отсутствии ошибок округления и использовании защитной цифры, ошибка не превосходит 2ε.
- При возможности жёсткого уничтожения цифр часто можно переделать формулу.
- Рассмотрим выражение $x^2 - y^2$. Так как операнды — результаты вычисления с округлением, это — жёсткий вариант, даже если x и y — точные.
- Как переделать формулу, чтобы перейти в мягкий вариант?

Мягкое уничтожение цифр

- Мягкое уничтожение цифр возникает при вычитании точно известных значений. При отсутствии ошибок округления и использовании защитной цифры, ошибка не превосходит 2ε.
- При возможности жёсткого уничтожения цифр часто можно переделать формулу.
- Рассмотрим выражение $x^2 - y^2$. Так как операнды — результаты вычисления с округлением, это — жёсткий вариант, даже если x и y — точные.
- Как переделать формулу, чтобы перейти в мягкий вариант?
- $(x - y)(x + y)$.
- Ошибка обоих операндов не более 2ε.

Начисление банковских процентов

- Рассмотрим выражение $(1+x)^n$, где $x \ll 1$.
- Пусть мы каждый день депонируем \$100, годовой процент равен 6. Какая сумма должна оказаться на счету через 365 дней?
- Для $n = 365$ и $i = 0.06$ общая сумма вычислится по формуле $100\dfrac{(1+\frac{i}{n})^n - 1}{\frac{i}{n}}$
- При $\beta = 2$ и $p = 24$ результат равен \$37615.45 при правильном значении \$37614.05.
- Сложение единицы с i/n приводит к тому, что много разрядов результата жёстко уничтожается.
- Многократное умножение подчёркивает это.
- Выражение $(1 + i/n)^n$ может быть переписано как $e^n \ln(1 + i/n)$.
- Количество жёстких операций уменьшается при использовании следующей теоремы.

Theorem (О вычислении натурального логарифма)

При вычислении $\ln(1+x)$ *по формуле*

$$\ln(1+x) = \begin{cases} x, \text{ если } 1 \oplus x = 1 \\ \dfrac{x}{\ln(1+x)}(1+x) - 1, \text{ если } 1 \oplus x \neq 1, \end{cases}$$

то относительная ошибка составит не более 5ε*, когда* $0 \leqslant x < \frac{3}{4}$*, вычитание производится с защитной цифрой и* $\ln$ *вычисляется с точностью* $\frac{1}{2}LDV$*.*

Небольшое резюме

1. Вычитание как разных, так и близких по порядку чисел, ведёт к уничтожению цифр результата, жёсткому или мягкому.
2. Часто формулы нужно переписывать с применением мягких форм уничтожения, но это полезно только при операциях с защитными цифрами.
3. Стоимость защитной цифры на компьютерах — один бит.

Округление

- Как округлить число 2.5 до целого?
- А как округлить число 3.5 до целого?
- Имеется две модели округления: первая округляет цифру 5 вниз, вторая — в сторону чётного числа.

Theorem (Об округлении)

Пусть x и y имеют экспоненциальное представление. Определим $x_0 = x$, $x_1 = (x_0 \ominus y) \oplus y, \ldots, x_n = (x_{n-1} \ominus y) \oplus y$.. Если операции $\oplus$ и $\ominus$ округляют к чётному, то либо $x_n = x$ для всех n или $x_n = x_1$ для всех $n \geqslant 1$.

Пусть $\beta = 10$, $p = 3$, $x = 1.00$, $y = -0.555$. При округлении вверх $x_0 \ominus y = 1.56$, $x_1 = 1.56 \ominus 0.555 = 1.01$, $x_1 \ominus y = 1.57$ и т. д. Округление к чётному x — всегда 1.00.

Немного практики

Стандарт IEEE754

- Этот стандарт положил конец войне форматов вещественных чисел.
- Много лет каждый производитель вычислительных систем использовал свой, удобный для них формат.
- Существовали системы:
  - где все целые числа представлялись в экспоненциальном формате (CDC6000 — 60 бит слово, 48 бит из них — целое представление, 10 бит — экспонента);
  - где округление производилось всегда вверх (VAX-11);
  - где $\beta = 16$ (IBM).

Стандарт IEEE754

- `IEEE754` определил три вида вещественных чисел: $\beta = 2, p = 24$ и $\beta = 2, p = 53$, $\beta = 2, p = 64$.
- Первый вид — одиночной точности. Диапазон $2^{-126}..2^{127}$.
- Второй вид — двойной точности. Диапазон $2^{-1022}..2^{1023}$.
- Третий вид — расширенной точности. Диапазон $2^{-16382}..2^{16383}$.
- Имеется так называемый скрытый бит. В записи $d.dd..ddd \times \beta^e$ первый бит перед точкой всегда принимается равным единице — обычно числа *нормализованы*.
- Имеются такие числа, как $+0$, -0, $+\infty$, $-\infty$, `NaN`.
-

NaN

- До появления *IEEE754*, для большинства машин каждый набор бит представлял конкретное вещественное число.
- Это приводило к проблемам в вычислениях.
- Например, `IBM-370` при операции `sqrt(-1)` просто печатало сообщение об ошибке и возвращало `1`.
- Сейчас сообщений об ошибках нет, возвращается `NaN`.

Специальные значения *IEEE754*

Всегда имеется один бит, представляющий знак числа.

Экспонента	Мантисса	Представляет	Примечание
$e = e_{min} - 1$	$f = 0$	± 0	Два нуля
$e = e_{min} - 1$	$f \neq 0$	$0.f \times 2^{e_min}$	Денормализованные числа
$e_{min} \leqslant e \leqslant e_{max}$	—	$1.f \times 2^e$	Обычные числа
$e = e_{max} + 1$	$f = 0$	$\pm\infty$	Бесконечность
$e = e_{max} + 1$	$\neq 0$	$\pm$NaN	Не число

Для чего потребовался NaN

- Классические процессоры при делении на ноль или при нахождении арксинуса от 4 в невоенное время печатали сообщение об ошибка и останавливались.
- Предположим, нам надо найти производную в точке непрерывности функции, имеющей точки разрыва.
- $f'(x) \approx \dfrac{f(x+dx) - f(x)}{dx}$.
- При численном вычислении функции $f(x+dx)$ мы можем попасть в точку разрыва.
- Вычисление функции может быть сложным и мы не можем заранее выделить все такие точки.
- Тогда проще всего позволить функции вернуть `NaN` для определения проблемы.

Бесконечность

- Если происходит деление неотрицательного числа на обычный ноль (положительный), то удобнее воспользоваться знаком $+\infty$.
- Бесконечность — такой же операнд в вычислениях, как и другие числа.
- $-1 \times -\infty \to +\infty$.

Некоторые специальные случаи

Операция	Результат
$+\infty + +\infty$	$+\infty$
$0 \times \infty$	`NaN`
$x/0, x \neq 0$	∞
$0/0$	`NaN`
∞/∞	`NaN`
$x \bmod 0$	`NaN`
$\infty \bmod x$	`NaN`
$\sqrt{x}, x < 0$	`NaN`

Важность ∞

- Вариант выдачи вместо бесконечности наибольшего представимого числа не корректен.
- Пусть для $\beta = 10, p = 3, e_{max} = 98$ имеются $x = 3 \times 10^{70}$ и $y = 3 \times 10^{70}$.
- Нужно вычислить $\sqrt{x^2 + y^2}$.
- В случае возвращения наибольшего представимого числа 9.99×10^{98} результат неверен и это никак не определить.
- В случае возвращения ∞ результат равен ∞.

Важность денормализованных чисел и знаковых нулей

- При суммировании рядов иногда мы готовы потерять в точности представления, но сохранить число ненулевым как можно дольше.
- Дальнейшее уменьшение ненормализованного числа приводит к нулю.
- Знак нуля сохраняет направление, откуда мы пришли.

Проблемы действительных чисел.

Неассоциативность

- Почти вся алгебра, в которой мы работали, опирается на ассоциативность основных операций.
- Арифметика вещественных чисел не ассоциативна!
- $(a + b) + c \neq a + (b + c)$.
- $(10^16 + 1) - 10^16 = 0$.
- $(10^10 - 10^16) = 1 = 1$.
- Суммирование ряда $1 + 1/2 + ... + 1/n$ даст разные результаты при суммировании слева/направо и справа/налево.

Числовые константы

- Привычные нам числовые константы часто не имеют точного представления.
- Константа 0.2 в системе $\beta = 2, p = 24$ будет примерно равна 0.200000003.
- При использовании такой константы в цикле мы будем накапливать погрешность.

Неожиданное поведение

- Вычисление минимума двух переменных x и y можно реализовать несколькими способами.
  - `если x < y то x иначе y`
  - `если x ⩽ y то x иначе y`

  При $x = +0$ и $y = -0$ результат различен.
  - `если x < y то x иначе y`
  - `если x > y то y иначе x`

  Результат различен при $x = 1$, $y =$`NaN`.

Неожиданное поведение

- Операция деления обычно исполняется дольше операции умножения.
- Эквивалентны ли операции:
  1. $y \to x/2.0$ и
  2. $y \to x \times 0.5$?

Неожиданное поведение

- Операция деления обычно исполняется дольше операции умножения.
- Эквивалентны ли операции:
  1. $y \to x/2.0$ и
  2. $y \to x \times 0.5$?

  А следующие:
  1. $y \to x/10.0$ и
  2. $y \to x \times 0.1$?

  Для $\beta = 2$ первое выражение эквивалентно, второе — нет. Для $\beta = 10$ второе выражение эквивалентно, первое — нет.

Неожиданное поведение

- Что должен делать следующий код на Си?

```
epsilon = 1;
do {
    epsilon = epsilon * 0.5;
} while (epsilon + 1 > 1);
```

Неожиданное поведение

- Что должен делать следующий код на Си?

```
epsilon = 1;
do {
    epsilon = epsilon * 0.5;
} while (epsilon + 1 > 1);
```

- Определить максимальное число, которое можно прибавить к единице, чтобы результат остался единицей.
- Запуск программы на Си, Python приводит к предсказуемому результату:

  `epsilon = 1.1102230246251565e-16`

  Компиляция Си с флагом `-ffast-math` даёт

  `epsilon = 1.1102230246251565e-16`

  Почему?

Неожиданное поведение

- Что должен делать следующий код на Си?

```
epsilon = 1;
do {
    epsilon = epsilon * 0.5;
} while (epsilon + 1 > 1);
```

- Определить максимальное число, которое можно прибавить к единице, чтобы результат остался единицей.
- Запуск программы на Си, Python приводит к предсказуемому результату:

  `epsilon = 1.1102230246251565e-16`

  Компиляция Си с флагом `-ffast-math` даёт

  `epsilon = 1.1102230246251565e-16`

  Почему?
  Этот флаг, в том числе, разрешает считать операции вещественной арифметики ассоциативными.

Сравнение чисел на равенство

- Это — запрещённая операция в арифметике вещественных чисел.
- Если при арифметических операциях должно получиться 0.6, то сравнение «если x = 0.6 то ...», вероятно, никогда не сработает.
- Первая причина — погрешность вычислений x.
- Вторая причина — принципиальная неточность константы 0.6

Сравнение чисел на равенство

- Можно ли вообще сравнивать вещественные числа на равенство?
- Сильно не рекомендуется.
- Попробовать можно, но результат будет неверным.
- Верно ли сравнение:

```
if (x == y) return;
```

Сравнение чисел на равенство

- Можно ли вообще сравнивать вещественные числа на равенство?
- Сильно не рекомендуется.
- Попробовать можно, но результат будет неверным.
- Верно ли сравнение:

```
if (x == y) return;
```

- Совершенно нет.
- А такое:

```
if (fabs(x == y) < 0.000001) return;
```

Сравнение чисел на равенство

- Можно ли вообще сравнивать вещественные числа на равенство?
- Сильно не рекомендуется.
- Попробовать можно, но результат будет неверным.
- Верно ли сравнение:

```
if (x == y) return;
```

- Совершенно нет.
- А такое:

```
if (fabs(x == y) < 0.000001) return;
```

- Совершенно нет для произвольных `x` и `y`. Что будет, например, если $x \approx y \approx 10000$?
- Для $\beta = 2, p = 24$ (float) следующее представимое число после 10000 будет 10000.000977.

Сравнение чисел на равенство

- Универсальное сравнение на равенство в Си выглядит примерно так:

```
if (fabs(x - y) <
  max(fabs(x),fabs(y)) * SOME_CONST * DBL_EPSILON)
  return x;
```

SOME_CONST выбирается обычно от 4 до 32.