Применение модулярной арифметики для вычислений над сверхдлинными неотрицательными целыми числами

Докладчик:

канд.физ.-мат.наук, Ионисян А.С.

Сверхдлинная арифметика

Современные задачи вычислительной математики требуют применения высокопроизводительных вычислений, связанных с выполнением арифметических операций над сверхдлинными целыми числами (сотни десятичных цифр).

Одним из эффективных способов ускорения арифметических операций над такими числами является применение модулярной арифметики или системы остаточных классов (СОК)

Модулярная арифметика

Модулярная арифметика (СОК) - это способ ускорения арифметических операций путем их распараллеливания по независимым вычислительным каналам.

В СОК каждое число - суть остатки от деления исходного числа в позиционной системе счисления (ПСС) на набор взаимно-простых чисел-оснований СОК.

Пример чисел в СОК

Пусть основания СОК: p1=3, p2=5, p3=7. Тогда диапазон СОК: [0;3*5*7-1] = [0;104].

Обозначим «mod» операцию нахождения остатка от деления чисел.

Если A=7, то A = (7 mod 3, 7 mod 5, 7 mod 7) = (1,2,0).

Если B=9, то B = (9 mod 3, 9 mod 5, 9 mod 7) = (0,4,2).

Сложение чисел в СОК

A = 7 =(7 mod 3, 7 mod 5, 7 mod 7) = (1,2,0).

B = 9 = (9 mod 3, 9 mod 5, 9 mod 7) = (0,4,2).

A+B=7+9=16 – в позиционной системе счисления.

A+B=(1,2,0) + (0,4,2) = (1+0 (mod 3), 2+4 (mod 5) , 0+2 (mod 7)) = (1, 1, 2).

Непосредственной проверкой убеждаемся в правильности ответа.

16=(16 mod 3, 16 mod 5, 16 mod 7) = (1, 1, 2).

Перемножение чисел в СОК

A = 7 =(7 mod 3, 7 mod 5, 7 mod 7) = (1,2,0).

B = 9 = (9 mod 3, 9 mod 5, 9 mod 7) = (0,4,2).

A*B = 7*9 = 63 – в позиционной системе счисления.

A*B = (1,2,0) * (0,4,2) =

= (1*0 (mod 3), 2*4 (mod 5) , 0*2 (mod 7)) = (0, 3, 0).

Проверка: 63=(63 mod 3, 63 mod 5, 63 mod 7) = (0, 3, 0).

А как быть с вычитанием и делением?

С вычитанием все работает отлично.

A = 8 = (8 mod 3, 8 mod 5, 8 mod 7) = (2,3,1).

B = 5 = (5 mod 3, 5 mod 5, 5 mod 7) = (2,0,5).

A-B = 8-5 = 3 – в позиционной системе счисления.

A-B = (2,3,1) - (2,0,5) =

= (2-2 (mod 3), 3-0 (mod 5) , 1-5 (mod 7)) = (0, 3, 3).

Проверка: 3 = (3 mod 3, 3 mod 5, 3 mod 7) = (0, 3, 3).

Деление?

Увы, если числа не делятся нацело, то деление, в общем случае, корректно не работает ☹

Пример достаточно длинный, требует объяснения понятия «обратный мультипликативный элемент» и поэтому здесь показан не будет.

Проблемная ситуация

Видно, что выполнение операций сложения, вычитания и умножения в СОК можно выполнять параллельно и независимо друг от друга, но при этом встает вопрос перевода чисел из позиционной системы счисления в СОК и обратно.

Переход из десятичной СС в СОК

Воспользуемся фактом, что любое число в ПСС можно представить в виде суммы произведений цифр этого числа на соответствующую степень основания системы счисления.

Например:

1979=1*1000+9*100+7*10+9

2011=2*1000+0*100+1*10+1.

Переход из десятичной СС в СОК

Мы можем заранее найти значения как каждой цифры позиционной системы счисления, так и каждой степени этой системы счисления в заданной СОК и затем "собрать" число из этих СОК-чисел, перемножая и складывая их

Например, для СОК, заданной своими основаниями p1=3, p2=5, p3=7:

0=(0,0,0), 1=(1,1,1), 2=(2,2,2), 3=(0,3,3), 4=(1,4,4), 5=(2,0,5), 6=(0,1,6), 7=(1,2,0), 8=(2,3,1), 9=(0,4,2), 10=(1,0,3), 100=(1,0,2), и т.д.

Еще быстрее (на компьютере) ☺

Если в вычислительной системе достаточно памяти, то можно заранее дополнительно вычислить числа 20,30,40,50,60,70,80,90,200,300,400,…, что сильно убыстрит процесс перевода чисел из ПСС в СОК, фактически, убрав из него все арифметические операции.

20=(2,0,6), 30=(0,0,2), 40=(1,0,5), 50=(2,0,1), 60=(0,0,4), 70=(1,0,0), 80=(2,0,3), 90=(0,0,6).

А если осуществлять поканальное модулярное сложение путем выборки из соответствующих просмотровых таблиц, то даже промежуточные операции сложения оказываются не нужны.

Переведем число 69 в СОК (3,5,7)

Предположим, что заранее вычислены константы

0=(0,0,0), 1=(1,1,1), 2=(2,2,2), 3=(0,3,3), 4=(1,4,4),

5=(2,0,5), 6=(0,1,6), 7=(1,2,0), 8=(2,3,1), 9=(0,4,2),

10=(1,0,3), 20=(2,0,6), 30=(0,0,2), 40=(1,0,5), 50=(2,0,1), 60=(0,0,4), 70=(1,0,0), 80=(2,0,3), 90=(0,0,6).

Тогда

69=60+9=(0,0,4)+(0,4,2)=(0,4,6).

И все так просто? ДА!

А как обратно из СОК в 10-ю СС?

Обратный переход из СОК в ПСС долгое время считался вычислительно сложной задачей.

Мы считаем, что нам удалось создать предельно быстрый метод перевода чисел из СОК в ПСС, основанный на модификации метода ортогональных базисов.

Ортогональные числа

Назовем ортогональными числами в СОК числа A и B, дающие нуль при нахождение произведения A*B и имеющие ровно один ненулевой разряд в СОК-представлении.

Например, ортогональными числами в СОК с основаниями p1=3, p2=5, p3=7 будут числа A=(2,0,0) и B=(0,3,0).

Всего для данного набора оснований возможно 12 ортогональных чисел:

(1,0,0), (2,0,0), (0,1,0), (0,2,0), (0,3,0), (0,4,0),

(0,0,1), (0,0,2), (0,0,3), (0,0,4), (0,0,5), (0,0,6).

Дополнительно для проведения расчетов нам потребуется тривиальное представление нуля в СОК: (0,0, …,0)=0.

А как найти эти ортогональные числа?

Значение каждого из ортогональных чисел можно предварительно найти в ПСС, например, используя перебор (для малого диапазона СОК) или более сложные методы, такие как «ОПСС» или приближенный метод профессора Н.И.Червякова.

(1,0,0)=70, (2,0,0)=35,

(0,1,0)=21, (0,2,0)=42, (0,3,0)=63, (0,4,0)=84

(0,0,1)=15, (0,0,2)=30, (0,0,3)=45, (0,0,4)=60, (0,0,5)=75, (0,0,6)=90

И зачем эти ортогональные числа нужны?

Для нахождения значения числа, заданного в СОК своими остатками X=(x1,x2, …) от деления на основания СОК (p1, p2, …) достаточно разложить число X на сумму ортогональных чисел сначала в СОК, а потом, заменив ортогональные числа их переводами в ПСС, вычислить эту сумму уже в ПСС. Результат получится с точностью до переполнения диапазона СОК

Пример:

X = 9 = (0,4,2) = (0,0,0)+(0,4,0)+(0,0,2) = 0+84+30 =

= 114= 9+1*105 (диапазон СОК переполнен 1 раз).

От переполнения нужно как-то избавиться!

Что же получается?

X = 9 = (0,4,2) = (0,0,0)+(0,4,0)+(0,0,2) = 0+84+30 = 114.

С практической точки зрения от переполнения диапазона СОК необходимо избавиться.

Для разобранного выше примера

X=(0,4,2)=114 => X=114 mod 105=9 (плохой способ)

Последняя операция нахождения остатка от деления нивелирует весь эффект использования СОК, так как вычислительно трудоемка и при аппаратной реализации требует наличия отдельного микропроцессора.

Но можно и без микропроцессора!

А именно, применяя технику бинарного сдваивания с коррекцией переполнения на каждом шаге расчета суммы ортогональных чисел в ПСС.

Действительно, каждое из ортогональных чисел в ПСС не превышает диапазона СОК, следовательно, сумма пары таких чисел не превысит удвоенного диапазона СОК и по правилам модулярной арифметики может быть тут же скорректирована путем вычитания диапазона (если сумма пары превысила диапазон)

Без примера понять сложно

Пример:

X = (0,4,2) = (0,0,0)+(0,4,0)+(0,0,2) =

= 0+84+30 = 114

0+84+30=(0+84)+(30)

0+84=84<105 (коррекция не требуется)

84+30=114>=105 (производим корректировку, вычитая диапазон 105): 114-105=9.

Ответ: 9

Более реальный пример СОК с основаниями p1=2, p1=3, p1=5, p1=7, p1=11, p1=13, p1=17, p1=19 (диапазон СОК – P=9699690)

Предположим, что мы хотим узнать значение числа (1,2,3,4,5,6,7,8) из этой СОК в ПСС.

Число (1,2,3,4,5,6,7,8) раскладывается на сумму 8 ортогональных чисел.

- (1,0,0,0,0,0,0,0)=4849845
- (0,2,0,0,0,0,0,0)=6466460
- (0,0,3,0,0,0,0,0)=1939938
- (0,0,0,4,0,0,0,0)=4157010
- (0,0,0,0,5,0,0,0)=1763580
- (0,0,0,0,0,6,0,0)=2984520
- (0,0,0,0,0,0,7,0)=5705700
- (0,0,0,0,0,0,0,8)=5615610

Более реальный пример СОК с основаниями p1=2, p1=3, p1=5, p1=7, p1=11, p1=13, p1=17, p1=19 (диапазон СОК – P=9699690)

На первом этапе бинарного сдваивания, найдем суммы чисел:

- 4849845+6466460=11316305>=P (нужна коррекция) => 1616615
- 1939938+4157010=6096948<P (коррекция не нужна)
- 1763580+2984520=4748100<P (коррекция не нужна)
- 5705700+5615610=11321310>=P (нужна коррекция) => 1621620

На втором этапе бинарного сдваивания найдем суммы чисел:

- 1616615+6096948=7713563<P (коррекция не нужна)
- 4748100+1621620=6369720<P (коррекция не нужна)

На третьем этапе бинарного сдваивания:

- 7713563+ 6369720=14083283>=P (нужна коррекция) => 4383593

Полученный ответ является окончательным.

Ответ: 4383593

Это реально эффективно?

Да, так как нигде, кроме этапа инициализации (расчет таблицы ортогональных чисел), мы не использовали ни операции умножения, ни операции деления, а только сложение и вычитание, которые выполняются быстро даже на «древних» ЭВМ.

Зачем все это школьникам?

- Хороший вопрос.
  Думаю, что лично я знаю на него ответ.
- Предлагаю обсудить ☺