Использование вычетов для разбиения числа на простые множители

Программа включает в себя ряд методик, которые используют вычеты для получения p & q из n, где n=pq. Наиболее интересны методики серии “базис A”, т. к. обладают большей эффективностью в сравнении с другими.

Дельта – Правый (базис A).

Самый простой метод основан на получении вычетов $k_i = n \bmod 2^i$, где $i = 1, 2 \ldots \log_2 n$; затем полученные вычеты складываются начиная с первого; на каждом шаге происходит проверка является ли полученная сумма делителем n; если сумма чётна, то n делится на сумму вычетов минус 1 – в обратном случае просто на сумму вычетов.

Дельта – Левый (базис A).

Его принцип работы следующий:

Шаг 1 :

$\frac{n}{k_1}$

$\frac{n}{k_1 + k_0}$

Шаг 2 :

$\frac{n}{k_2}$

$\frac{n}{k_2 + k_1}$

$\frac{n}{k_2 + k_1 + k_0}$

****

Шаг i :

$\frac{n}{k_i}$

$\frac{n}{k_i + k_{i-1}}$

. . . . . .

$\frac{n}{k_i + k_{i-1} + \ldots + k_0}$

При этом, как и в первом методе, может проводиться проверка на чётность/нечётность суммы вычетов.

Дельта – Декриз (базис A).

Принцип работы следующий:

Шаг 1 :

$\frac{n}{k_1}$

$\frac{n}{k_1 - k_0}$

Шаг 2 :

$\frac{n}{k_2}$

$\frac{n}{k_2 - k_1}$

$\frac{n}{k_2 - k_1 - k_0}$

****

Шаг i :

$\frac{n}{k_i}$

$\frac{n}{k_i - k_{i-1}}$

. . . . . .

$\frac{n}{k_i - k_{i-1} - \ldots - k_0}$

Прошу извинить меня за некоторое несоответствие: в пошаговых примерах индексация идёт с нуля, а в формулах с единицы.

Расширенные версии.

Получают $q = n \bmod \sum_i k_i$, *а затем* $p = \frac{n}{q}$.

Dee..ep (basis A).

Обрабатывает случаи, когда $qt = n \bmod qr$ $(1 \le t < r)$, где qr – сумма вычетов. Является более совершенной версией “Delta – LeFt eXt (basis A)”, но более медленной.

То, на чём методики ломают зубы.

n устойчиво ко всем методам серии “базис А”, если из полученных вычетов нельзя получить сумму равную qr, где $1 \le r < p$.

Замечания по второй версии эппа.

Добавлены два метода (BS & DF--) по эффективности сравнимые с Dee..ep - о них я расскажу немного позже, и исправлен баг в Dee..ep’е.

Два способа малость увеличить эффективность алгоритмов.

1. Возвести n в какую либо степень: например расширенная версия Декриза не “бьёт” 25 – зато без проблем дробит 625 на 5 и 125; ещё одно число 35 – при возведении в куб ломается на ура (кстати, качайте самую последнюю версию эппа, т. к. я из – за потери сна на всех этих вопросах, допускаю в проге досадные траблы, впрочем, я не жалуюсь – это того стоит☺☺).
2. Умножить n на z, правда, я, на данный момент, не обладаю быстрой методикой по подбору z, т. ч. с практической точки зрения она пока выглядит весьма сомнительной.

Suture (Шов).

Даная методика относится к разряду абсолютно эффективных. Действует следующим образом: $m = 2^t$, $\gcd(zn - im + n - m, n) = q$ - можно утверждать, что для каждого n найдётся тройка чисел (z, t, i), при которой $q > 1$. Мы имеем дело с рядом $zn - im = qk_i + r_i$, где r принимает период. значения – таким образом на i-ом шаге мы получим r_i, где $r_i + r_1 = q$ $---->$ после чего будет получена некая сумма qk, где k!=p. Использование данной методики в чистом виде на больших числах малоэффективно, так как период слишком большой.

Forced Brute Force.

Первым делом хочу заметить: в некоторых умных книжках почему – то пишут ересь, что Брут форс требует $\sqrt{pq}$ итераций – зачем при разломе нечётного числа учитывать чётные числа (???!!!), так что итераций в чистом Брут форсе $\sqrt{pq}\frac{1}{2}$. Теперь рассмотрим сумму $n + \frac{n}{2^i}; \frac{n}{2^i} = \left[\frac{n}{2^i}\right] + tail; \sum_i \frac{q}{2^i} \geq tail$. Ну и затем юзаем схему **while**($\gcd(sum, n) == 1$) sum--, либо **while**($\gcd(sum, n) == 1$) sum++ ($sum == n + \frac{n}{2^i}$) или оба варианта сразу, если у нас распред. система. Декрементировать/инкрементировать sum в форсированном Брут форсе нужно единицей – иначе есть риск разминуться с ближайшей точкой, правда, на распределёнке этот риск гораздо ниже, стоит ещё отметить, что возможность промаха взаимосвязана с направлением округления деления. Кол – во итераций лежит на отрезке $\left[\frac{\sqrt{n}}{2^i}, \sqrt{n}\right]$.

Dee..ep Ext. (Based on Suture).

Учитывая опыт старичка Дипа и Шва, имеет смысл использовать вместо схемы $i = i, i-1..1 \sum_i pq \bmod 2^i$, иной подход $i = i, i-1..1 \sum_i pq - 2^i$. Он менее прожорлив, а эффективность как минимум та же.

Length TheA.

Данная методика попытка свести поиск факторов к бинарному дереву, не могу утверждать сейчас, что подход подобный абсолютно эффективен, но абсолютно точно имеет право на существование. Итак, если мы знаем $p + q = z$ (z – известен; p, q - ??), то по бинарному

дереву можно точно отыскать p & q, правда, на практике вопрос упирается в нахождение z. Один из способов нахождение: min. t, где $t^2 > n$, тогда принимается $z = 2t$.

P. S.

Мной разрабатываются и другие методы разбиения n, но пока они довольно “сыры”, чтобы о них рассказывать.

P.P. S.

Буду благодарен за участие в проекте, как по вопросам программирования, математики – так и финансовой поддержке, к тому же, надо заметить, что работаю и в других направлениях имеющих не меньшее прикладное значение, чем вышеизложенное.

e-mail: xft_turbo@mail.ru
Web: http://xproject-all.narod.ru/prgsale.htm

Пример программы без арифметики больших чисел:
http://xproject-all.narod.ru/catcher_of_secret_key_ver2.zip

Редактирован 2007-03-03.