АНАЛИЗ ПОСЛЕДОВАТЕЛЬНОСТЕЙ И ТАБЛИЦ ДАННЫХ
(Вычислительные системы)
1994 год Выпуск 150

УДК 8.74

ХЕШИРОВАНИЕ СИМВОЛЬНЫХ ЦЕПОЧЕК В РЕЖИМЕ СКОЛЬЗЯЩЕГО ОКНА

В.Д.Гусев, Т.Н.Титкова

В в е д е н и е

Хеширование - эффективный метод поиска элементов в информационном массиве [1],характеризующийся константной в среднем трудоемкостью (время поиска не зависит от числа элементов в массиве). В качестве элементов массива часто фигурируют сим - вольные цепочки (обычно небольшой длины). Метод хеширования хорошо зарекомендовал себя при отыскании повторов произвольной длины в текстовых базах данных большого объема [2]. Задача отыскания всевозможных повторов в тексте является актуальной для таких прикладных областей как информатика, компьютерная гене - тика, музыкометрика.

Полное множество повторов в тексте можно разбить на под - множества, содержащие лишь повторы фиксированной длины (l = 1, $2,\ldots,l_{max}$, где l_{max} - длина максимального повтора в тексте). Подмножество, содержащее лишь повторы длины l, будем называть частотной характеристикой 1-го порядка и обозначать $\Phi_1(T)$, где T - анализируемый текст. Для получения $\Phi_1(T)$ в [2] предложен линейный в среднем алгоритм, использующий идеи хеширования.Аргументами хеш-функции в этом алгоритме являются l-граммы - цепочки из l подряд следующих символов текста T. Они вырезаются из текста скользящим окном ширины l, которое на каждом шаге смещается на один символ вправо.

Параметр $l (1 \leq l \leq l_{max})$ может быть достаточно большим. К примеру, в приложениях, связанных с компьютерной генетикой, длина максимального повтора может составлять сотни и даже тысячи символов. Точное вычисление хеш-функций с аргументами такой длины становится нетривиальной вычислительной задачей. Трудоемкость поиска начинает существенным образом зависеть от длины аргумента. Используемый в этих случаях стандартный прием состоит во введении промежуточного отображения, которое редуцирует исходный (длинный) аргумент x в короткий - x', соответствующий размеру машинного слова. В силу неинъективности отображения возможно "склеивание" некоторых аргументов, что приводит к увеличению числа наложений и соответственно повышению трудоемкости процедуры хеширования. Зависимость трудоемкости поиска от l слегка нивелируется, но сохраняется.

Целью данной работы является получение рекуррентных соотношений, в которых устраняется зависимость трудоемкости вычисления хеш-функций от длины аргумента l для случая хеширования в скользящем окне. Такая возможность обеспечивается "зацепленностью" соседних хешируемых l-грамм. Дополнительным преимуществом метода является то, что все хеш-функции вычисляются точно, т.е. устраняются дополнительные наложения, возникавшие из-за использования редуцирующего преобразования. Для вычисления хеш-функции от начальной (самой левой) l-граммы текста выводится отдельное рекуррентное соотношение. Оно представляет самостоятельный интерес в качестве основы для стандартной машинной процедуры вычисления модульной функции от длинного аргумента.

1. Вывод рекуррентных соотношений

Одной из наиболее эффективных и широко используемых хеш-функций является модульная:

$$h(x) = K(x) \bmod M = K(x) - M\lfloor K(x)/M \rfloor, \qquad (1)$$

где $K(x)$ - числовой код элемента информационного массива x; M - положительное целое число, соответствующее числу квантов памяти, отводимой под размещение информационного массива; $h(x)$ - номер кванта памяти, в который заносится информация об элементе x $(0 \le h(x) \le M-1)$; $\lfloor Z \rfloor$ - означает целое, ближайшее снизу к Z. Выбор функции (1) в качестве базовой продиктован следующими соображениями:

1) для больших информационных массивов довольно сложно сконструировать удовлетворяющую специфике информационного поиска совершенную хеш-функцию (т.е. функцию, не допускающую ситуаций типа $h(x_1) = h(x_2)$ при $x_1 \neq x_2$, квалифицируемых как наложения). Среди множества известных несовершенных (т.е. допускающих наложения) хеш-функций модульная характеризуется наименьшим в среднем числом наложений (см. результаты экспериментального исследования в [3]);

2) довольно слабые ограничения предъявляются к выбору параметра M: по возможности он не должен содержать малых делителей. У многих хеш-функций этот параметр может быть задан лишь в виде $M = 2^k$, где k - целое число, что может привести к нерациональному использованию памяти;

3) объекты с близкими значениями $K(x)(|K(x_1)-K(x_2)| \ll M)$, как правило, характеризуются близкими значениями $h(x)$, что может оказаться существенным в ряде приложений.

Пусть $T = a_1a_2\ldots a_N$ - исходный текст, а $x_1(i) = a_ia_{i+1}\ldots a_{i+l-1}$ и $x_1(i+1) = a_{i+1}a_{i+2}\ldots a_{i+1}$ - соответственно i-я и $(i+1)$-я l-граммы текста $(1 \le i \le N-l)$. Каждый символ алфавита будем интерпретировать как целое число в диапазоне от 0 до $n-1$, где n - мощность алфавита. Тогда $x_1(i)$ и $x_1(i+1)$ можно рассматривать как представления i-й и $i+1$-й l-грамм текста в n-ичной системе счисления. Эти представления определяют цело -

численные коды 1-грамм:

$$\left.\begin{aligned} K(x_1(i)) = K_i^{(1)} &= \sum_{q=1}^{1} a_{i+q-1}\cdot n^{1-q}, \\ K(x_1(i+1)) = K_{i+1}^{(1)} &= \sum_{q=1}^{1} a_{i+q}\cdot n^{1-q}. \end{aligned}\right\} \qquad (2)$$

Домножая первое равенство на n и вычитая его из второго, получим

$$K_{i+1}^{(1)} - n\cdot K_i^{(1)} = a_{i+1} - a_i\cdot n^1,$$

откуда

$$K_{i+1}^{(1)} = n\cdot K_i^{(1)} - a_i\cdot n^1 + a_{i+1}. \qquad (3)$$

При непосредственном вычислении (3) возникают трудности, связанные с возможностью выхода за пределы разрядной сетки ЭВМ при больших значениях n и 1. Желательно поэтому, не прибегая к вычислению (3), получить сразу связь между значениями хеш-функций двух соседних 1-грамм. В соответствии с (1) имеем:

$$\left.\begin{aligned} h(x_1(i)) = h_i^{(1)} = K_i^{(1)} \bmod M &= K_i^{(1)} - M\lfloor K_i^{(1)}/M\rfloor, \\ h(x_1(i+1)) = h_{i+1}^{(1)} = K_{i+1}^{(1)} \bmod M &= K_{i+1}^{(1)} - M\lfloor K_{i+1}^{(1)}/M\rfloor. \end{aligned}\right\} \qquad (4)$$

Подставим (3) в (4) и воспользуемся известными правилами модульной арифметики (см., например, [4, с.303]):

$$\left.\begin{aligned} u\cdot v \bmod M &= (u \bmod M)\cdot(v \bmod M) \bmod M, \\ (u\pm v) \bmod M &= (u \bmod M \pm v \bmod M) \bmod M. \end{aligned}\right\} \qquad (5)$$

Предполагая для простоты,что M > n (условие,которому всегда легко удовлетворить), имеем:

$$h_{i+1}^{(1)} = (n\cdot K_i^{(1)} - a_i n^1 + a_{i+1}) \bmod M =$$

$$= [n\cdot h_i^{(1)} \bmod M - a_i(n^1 \bmod M) \bmod M + a_{i+1}] \bmod M. \qquad (6)$$

Соотношение (6) определяет связь между значениями модульной функции для i-й и i+1-й l-грамм текста.

ПРИМЕР. Пусть u = TGCAGAA - фрагмент генетического текста, алфавит $\Sigma = \{A,G,T,C\}$, n = 4. Поставим в соответствие элементам алфавита целые числа: K(A) = 0, K(G) = 1, K(T) = 2, K(C) = 3. Проследим ход вычислений по рекуррентной схеме, используя с одной стороны соотношения (3) и (1), с другой стороны - соотношение (6). Пусть окно анализа l = 4, делитель M=53. Имеем:

$\underline{i = 1}$: $x_4(1) = TGCA$; $K(x_4(1)) = K_1^{(4)} = \sum_{q=1}^{4} a_q n^{4-q} =$

$= (K(T)\cdot n^3 + K(G)\cdot n^2 + K(C)\cdot n + K(A)) =$

$= (2\cdot 64+1\cdot 16+3\cdot 4+0) = 156$; $h_1^{(4)} = K_1^{(4)} \bmod 53 = 50.$

Используя $K_1^{(4)}$ и $h_1^{(4)}$ в качестве начальных значений, можно далее продолжать вычисления по рекуррентной схеме (3):

$\underline{i = 2}$: $x_4(2) = GCAG$; $K(x_4(2)) = K_2^{(4)} = (4\cdot K_1^{(4)} - K(T)\cdot 4^4 +$

$+ K(G)) = (4\cdot 156-2\cdot 156+1) = 113;$

$h_2^{(4)} = K_2^{(4)} \bmod 53 = 7.$

Аналогичный результат получим с помощью (6):

$h_2^{(4)} = (4\cdot h_1^{(4)} \bmod 53 - K(T))\cdot(4^4 \bmod 53) \bmod 53 + K(G) \bmod 53 =$

$= (4\cdot 50 \bmod 53 - 2\cdot 44 \bmod 53 + 1) \bmod 53 =$

$= (41-35+1) \bmod 53 = 7.$

$\underline{i = 3}$: $x_4(3) = CAGA$; $K_3^{(4)} = (4\cdot K_2^{(4)} - K(G)\cdot 4^4 + K(A)) =$

$= (4\cdot 113-1\cdot 256+0) = 196$; $h_3^{(4)} = K_3^{(4)} \bmod 53 = 37.$

Или $h_3^{(4)} = (4\cdot h_2^{(4)} \bmod 53 - K(G)\cdot(4^4 \bmod 53) \bmod 53 +$

$+ K(A)) \bmod 53 = (28-44+0) \bmod 53 = -16 \bmod 53 =$

$= -16-53\lfloor -16/53 \rfloor = -16-53\cdot(-1) = 37$ и т.д.

Рассмотренный пример носил иллюстративный характер. Соотношение (6) вычислялось непосредственно по формуле. Ниже будет предложена более эффективная схема вычисления.

2. Реализация вычислений по рекуррентной схеме (6)

Произведение $n \cdot h_i^{(1)}$ (первый член в квадратных скобках) очевидным образом удовлетворяет неравенству $0 \le n \cdot h_i^{(1)} \le n \cdot (M-1)$. Заводим упорядоченный по возрастанию массив значений: $0, M, 2M, \ldots, n \cdot M$ и методом двоичного поиска определяем положение величины $n \cdot h_i^{(1)}$ в этом ряду, т.е. определяем значение $0 \le j < n$ такое, что $j \cdot M \le n \cdot h_i^{(1)} < (j+1) \cdot M$. Тогда $n \cdot h_i^{(1)} \bmod M = n \cdot h_i^{(1)} - j \cdot M$. Трудоемкость поиска составляет порядка $\log_2 n$ сравнений. При $n = 4$, к примеру, потребуется всего два сравнения.

Значение $n^1 \bmod M$ (см. второй член в квадратных скобках) не связано с обрабатываемым текстом и может быть предвычислено заранее. Чтобы избежать возможности выхода за пределы разрядной сетки при больших значениях l, вычисления проводим рекурентно, используя соотношение (5):

$$n^i \bmod M = ((n^{i-1} \bmod M) \cdot n) \bmod M, \quad 1 \le i \le l.$$

Поскольку коэффициент a_i может принимать ограниченное число значений $(0,1,2,\ldots,n-1)$, величины $a_i \cdot (n^1 \bmod M) \bmod M$ могут быть затабулированы. Входом в соответствующую таблицу длины n являются значения a_i (это эквивалентно хешированию без наложений). Требуемая величина извлекается за одно обращение к таблице.

Поскольку для любых i имеем $0 \le a_{i+1} \le n-1 < M$, третий член в квадратных скобках не требует предварительного вычисления. Обозначим выражение, заключенное в квадратные скобки, через H_i. Поскольку $-M < H_i < 2M$, значение $h_{i+1}^{(1)}$ можно получить, избежав операции деления:

$$h_{i+1}^{(1)} = \begin{cases} H_i + M, \text{ если } H_i < 0, \\ H_i, \text{ если } 0 \le H_i < M, \\ H_i - M \text{ в остальных случаях.} \end{cases}$$

Нетрудно видеть, что трудоемкость вычисления $h_{i+1}^{(1)}$ не зависит от l. Для получения очередного значения хеш-функции требуется лишь знание предыдущего (по номеру позиции) значения. В определенных ситуациях это может оказаться неудобным, например, когда мы хешируем не каждую l-грамму, а лишь некоторые из них. В этом случае полезной может оказаться другая рекуррентная формула, связывающая значения хеш-функций для l-грамм, начинающихся в одной позиции, но имеющих разную длину (в данном случае для $x_l(i)$ и $x_{l+1}(i)$). Действительно, в соответствии с (3) и (2) имеем:

$$\begin{aligned} K_i^{(1)} &= n \cdot K_{i-1}^{(1)} - a_{i-1} \cdot n^l + a_{i+l-1} = \\ &= n \cdot \Big(\sum_{q=1}^{l} a_{i+q-2} \cdot n^{l-q} \Big) - a_{i-1} \cdot n^l + a_{i+l-1} = \\ &= a_{i-1} \cdot n^l + n \Big(\sum_{q=2}^{l} a_{i+q-2} \cdot n^{l-q} \Big) - a_{i-1} \cdot n^l + a_{i+l-1} = \\ &= n \cdot K_i^{(l-1)} + a_{i+l-1}. \end{aligned} \qquad (7)$$

Отсюда для значений хеш-функций получаем следующую взаимосвязь:

$$h_i^{(1)} = (n \cdot h_i^{(l-1)} \bmod M + a_{i+l-1}) \bmod M. \qquad (8)$$

Соотношение (8) несколько проще, чем (6), и вычисляется по аналогичной схеме. Для получения $h_i^{(1)}$ требуется запомнить значение хеш-функции в этой же позиции на предыдущей $(l-1)$-й итерации. Формула (8) позволяет идти по тексту с нерегулярным шагом. Хешируются лишь те l-граммы, которые входят в состав повторов, т.е. не являются единичными. Платой за возможность

обработки текста с нерегулярным шагом являются дополнительные затраты памяти.

Другой существенный аспект использования соотношения (8) - получение с его помощью начального значения хеш-функции для схемы (6), т.е. значения $h_1^{(1)}$. Начальным значением для (8), в свою очередь, является $h_i^{(1)} = a_i$. Развивая указанное соображение,отметим, что в общем случае соотношение (8) можно рассматривать как рекуррентную схему вычисления модульной функции для аргументов произвольной длины.

В заключение данного раздела отметим, что если исходные данные представлены в двоичной (возможно, избыточной) кодировке,то в соотношениях (6) и (8) следует n заменить величиной 2^k, где k - значность используемого для записи символов двоичного кода.

3. Экспериментальная проверка рекуррентной хеш-функции

Представляет интерес сравнить рекуррентную хеш-функцию и хеш-функцию, использующую редукцию аргумента, по числу наложений - показателю,определяющему эффективность поиска. Чем меньше число наложений - тем эффективнее поиск. Существуют два противоположных действующих фактора, приводящих к различию в числе наложений в обоих вариантах хеширования.

Первый фактор связан с тем, что в результате использова - ния преобразования, редуцирующего длинный аргумент x в короткий x', могут возникнуть дополнительные наложения, т.е. возможны ситуации, когда два разных аргумента x_1 и x_2 такие, что $x_1 \bmod M \neq x_2 \bmod M$ редуцируются в x_1' и x_2' соответст - венно такие, что $x_1' \bmod M = x_2' \bmod M$.

Второй фактор приводит к противоположному эффекту. В ре - зультате редуцирующего преобразования могут быть устранены некоторые наложения, имеющие место при точном вычислении модульной функции, т.е. возможны ситуации, когда два разных аргумен-

та x_1 и x_2 такие, что $x_1 \bmod M = x_2 \bmod M$, редуцируются в x_1' и x_2' такие, что $x_1' \bmod M \neq x_2' \bmod M$.

Заранее неочевидно, приведет ли суммарное воздействие этих факторов к увеличению или уменьшению числа наложений в схеме рекуррентного хеширования по сравнению с исходной. Соответствующий вопрос исследовался экспериментально по методике, описанной в [3]. Критерием для сравнения процедур вычисления модульной функции является коэффициент заполнения расстановочного поля $\beta(\alpha) = n_o/M|_{\alpha=n_1/M}$, где M - размер расстановочного поля (делитель модульной функции), n_1 - число различных 1-грамм в хешируемом тексте, n_o - число занятых позиций в расстановочном поле, т.е. таких позиций, к которым произошло хотя бы по одному обращению в процессе заполнения расстановочного поля ($n_1 - n_o$ - число наложений); $\alpha = n_1/M$ - коэффициент загрузки расстановочного поля.

Пусть $h_1(x)$ - модульная хеш-функция с предварительным редуцированием аргумента, $h_2(x)$ - модульная рекуррентная хеш-функция, а $h_3(x)$ - пуассоновская хеш-функция, которая каждому значению аргумента x с одинаковой вероятностью ставит в соответствие любое значение адреса внутри расстановочного поля ($0 \leq h_3(x) \leq M-1$). Функция $h_3(x)$ является ориентиром для реальных несовершенных (т.е. допускающих наложения) хеш-функций. Добиться существенно лучших показателей по числу наложений по сравнению с $h_3(x)$ можно лишь путем значительного увеличения трудоемкости вычисления хеш-функции, что часто неприемлемо.

Для пуассоновской модели распределения значений хеш-функции математическое ожидание значений коэффициента β оценивается аналитически [3]. Если через μ_r, $r = 0,1,\ldots,n_1$, обозначить случайную величину, равную числу позиций расстановочного поля, к которым произошло ровно r обращений, то при больших значениях n_1 и N справедливо соотношение $E\,\mu_r \sim M \cdot p_r$,

где E - знак матожидания, а $p_r = \frac{\alpha^r}{r!} e^{-\alpha}$. Учитывая, что

$$\beta = \frac{n_o}{M} = \frac{M - \mu_o}{M} = 1 - \frac{\mu_o}{M},$$

имеем

$$E\beta = 1 - \frac{E\,\mu_o}{M} \sim 1 - e^{-\alpha}. \qquad (9)$$

Значимое отклонение в меньшую сторону выборочного значения параметра β от оценки (9) характеризует меру несовершенства используемой хеш-функции на данном наборе символьных цепочек.

Эксперимент проводился на различных фрагментах генетического текста (размер алфавита $n = 4$). Выбирался размер расстановочного поля M (он же делитель в модульной функции), фиксировался коэффициент загрузки α и параметр l. Длина текста в каждом эксперименте выбиралась такой, чтобы обеспечить заданный коэффициент загрузки, т.е. являлась переменной величиной. Подсчитывались коэффициенты заполнения расстановочного поля β_1 и β_2 для хеш-функций h_1 и h_2 соответственно и сравнивались с аналогичной оценкой для пуассоновской модели ($E\,\beta_3$). Для иллюстрации в таблице приведены значения соответствующих коэффициентов для разных текстов (T_1, T_2, T_3), длин цепочек ($l = 5,10$) и коэффицентов загрузки ($\alpha = 0,5;\ 0,7;\ 0,9;\ 1,1$). Параметр $M = 529$.

Анализ таблицы (и не включенных в публикацию экспериментальных данных) показывает, что при малых значениях l ($l = 5$ и 6) функция h_1 в среднем немного уступает h_2 по эффективности, т.е. характеризуется несколько бо́льшим числом наложений. В то же время функция h_2 хорошо согласуется по числу наложений с результатами, прогнозируемыми в соответствии с пуассоновской моделью.

Таблица

Выборочные значения коэффициентов заполнения расстановочного поля для различных текстов

а) 1 = 5:

α	Текст № 1		Текст № 2		Текст № 3		Пуассоновская модель E β_3
	β_1	β_2	β_1	β_2	β_1	β_2	
0,5	0,376	0,399	0,397	0,389	0,378	0,401	0,394
0,7	0,478	0,499	0,490	0,508	0,482	0,495	0,504
0,9	0,550	0,594	0,563	0,611	0,575	0,590	0,594
1,1	0,612	0,686	0,639	0,669	0,691	0,656	0,668

б) 1 = 10:

α	Текст № 1		Текст № 2		Текст № 3		Пуассоновская модель E β_3
	β_1	β_2	β_1	β_2	β_1	β_2	
0,5	0,388	0,406	0,403	0,374	0,401	0,397	0,394
0,7	0,499	0,495	0,512	0,507	0,512	0,510	0,504
0,9	0,585	0,594	0,597	0,594	0,592	0,611	0,594
1,1	0,652	0,673	0,662	0,682	0,658	0,684	0,668

При больших значениях 1 ($1 \geq 10$) функция h_1 улучшает свои показатели и все результаты становятся сопоставимыми. Это объясняется тем, что при $1 \geq 10$ длинный ключ редуцируется в короткий (четырех-байтовый) путем, как минимум, трехкратного наложения "лишних" символов друг на друга (10 = 4+4+2 или 10 = = 4+3+3). В этом состоит один из простейших методов хеширования - метод свертки. При кратности наложений 3 и выше он обладает хорошими рандомизирующими свойствами [3]. При 1 = 5 имеем всего лишь однократное наложение одного "лишнего" байта (5 = = 4+1) на любой из оставшихся четырех, чего явно недостаточно для хорошей рандомизации.

В целом результаты экспериментов продемонстрировали, что модульная хеш-функция, вычисляемая по рекуррентной схеме, обладает близкими к оптимальным показателями в широком диапазоне изменения значения l и α.

Заключение

Метод хеширования является эффективным средством для отыскания в длинных текстах всевозможных повторов фиксированной длины l. Трудоемкость соответствующих алгоритмов в среднем линейным образом зависит от длины текста N и параметра l. В работе предложена рекуррентная схема организации вычислений в задачах такого сорта, которая позволяет нивелировать зависимость от l в выражении для трудоемкости. Экспериментальная проверка показала, что по числу наложений рекуррентные хеш-функции близки к оптимальным, обеспечивающим равномерное распределение значений в заданном диапазоне.

Идея быстрого пересчета значений хеш-функции при специфических искажениях значения аргумента (в данном случае символьной цепочки) носит довольно общий характер. В частности, она может быть использована не только для организации вычислений в режиме скользящего окна, как в данной работе, но и для отыскания несовершенных повторов - фрагментов, отличающихся друг от друга незначительным числом замен, вставок и устранений символов.

Литература

1. КНУТ Д. Искусство программирования для ЭВМ. Т.3: Сортировка и поиск: Пер.с англ. - М.: Мир, 1978.- 844 с.

2. ГУСЕВ В.Д., КОСАРЕВ Ю.Г., ТИТКОВА Т.Н. О задаче поиска повторяющихся отрезков текста// Вычислительные системы.Вып.62. Ассоциативное кодирование. - Новосибирск,1975.- С.49-71.

3. ГУСЕВ В.Д., ТИТКОВА Т.Н. Экспериментальные исследования эффективности функций расстановки// Методы обработки информации. - Новосибирск, 1978. - Вып.74: Вычислительные системы.- С.69-90.

4. КНУТ Д. Искусство программирования для ЭВМ. Т.2: Получисленные алгоритмы: Пер.с англ.- М.: Мир, 1977.- 724 с.

Поступила в ред.-изд.отд.
29 июня 1994 года