О сжатии двоичного набора

М.Н. Назаров

В работе [1] был предложен метод кодирования двоичного строки, содержащей большое количество нулей, при помощи специальной функции, определенной на двоичных наборах. В данной работе будет рассмотрена задача сжатия произвольного двоичного набора большой длины и ее интерпретация с точки зрения хранении информации в базах данных. Особое внимание в этом случае уделяется возможности быстрого доступа к сжатой информации и ее обновлению.

1. Метод сжатия и некоторые оценки

Опишем сначала метод кодирования и декодирования двоичного набора, предложенные в [1]. Для любой булевой строки V обозначим через $l(V) = t$ длину этой строки, через $|V| = k$ количество единиц в этой строке, а через w — последний бит строки V. Тогда $V = V'w$ и

$$NUM(V) = \begin{cases} 0, & \text{если } t = k; \\ NUM(V'), & \text{если } w = 0; \\ C_{t-1}^{k} + NUM(V'), & \text{если } w = 1. \end{cases}$$

Строку V можно в сжатом виде представить как $(l(V), |V|, NUM(V))$. Процедура восстановления V по $(l(V), |V|, NUM(V))$ выглядит следующим образом:

Процедура 1.

1. Если $l(V) = |V|$, то $V = 1^{l(V)}$;

2. Если $C_{l(V)-1}^{|V|} \leqslant NUM(V)$, то $w = 1$ и полагаем:

$$NUM(V') = NUM(V) - C_{l(V)-1}^{|V|},\ |V'| = |V| - 1,\ l(V') = l(V) - 1;$$

3. Если $C_{l(V)-1}^{|V|} > NUM(V)$, то $w = 0$ и полагаем:

$NUM(V') = NUM(V),\ |V'| = |V|,\ l(V') = l(V) - 1.$

Очевидно, что число шагов этой процедуры есть линейная функция от длины строки V.

Рассмотрим теперь задачу сжатия двоичного набора большой длины с произвольным количеством единиц. Эту задачу можно интерпретировать как сжатие информации в базе данных. Итак, пусть задан произвольный двоичный набор W с достаточно большой длиной N и k — количество единиц в последовательности W. Очевидно, что применение описанного ранее метода кодирования ко всей базе данных сразу и вычисление функции $NUM(W)$ не оптимально. Действительно, в этом случае, чтобы извлечь информацию из начала базы потребуется декодировать всю строку W. Разобьем набор на части длиной n бит каждая. Получим $T = \lceil \frac{N}{n} \rceil$ таких частей и обозначим их через $V_1, V_2, \ldots, V_T$, а количество единиц в каждой части через $k_1, k_2, \ldots, k_T$. Вычислим для каждой части функцию $NUM(V_i)$.

Замечание 1. Из описания функции NUM ясно, что длина сжатого набора зависит от количества единиц в исходной последовательности. Чем больше единиц, тем больше величина функции NUM, а, следовательно, длина сжатого набора. Однако, если $k > \left[\frac{N}{2}\right]$, то можно инвертировать набор и только потом применить алгоритм сжатия. В этом случае, чтобы извлечь информацию необходимо определить, был инвертирован набор или нет. Для этого можно добавить один бит в начале закодированного набора. Если этот бит равен нулю, то набор не инвертирован, а если он равен единице, то после извлечения информации набор необходимо инвертировать. Таким образом, можно считать, что набор W может содержать только $k \leqslant \left[\frac{N}{2}\right]$ единиц.

При дальнейшем построении алгоритмов доступа это замечание учитываться не будет так как очевидно, что при необходимости все предложенные алгоритмы легко модифицировать.

Вернемся теперь к алгоритму сжатия двоичного набора W. Запишем каждую часть разбиения исходной булевой строки W в сжатом виде. Для этого, вычислив $NUM(V_1), NUM(V_2), \ldots, NUM(V_T)$, представим каждую часть в виде $(|V_1|, NUM(V_1)), (|V_2|, NUM(V_2)), \ldots, (|V_T|, NUM(V_T))$. Заметим, что $l(V_i)$ хранить не надо, так как длина всех частей равна n. Каждое десятичное число $|V_i|$ представим в двоичном виде и применим к двоичной записи оператор λ:

$$\lambda(a_1 a_2 \ldots a_r) = 00 a_1 (\neg a_1) a_2 (\neg a_2) \ldots a_r (\neg a_r) 11,$$

а число $NUM(V_i)$ просто представим в двоичном виде. Оператор λ необходим, чтобы можно было отделить двоичное представление $|V_i|$ от двоичного представления $NUM(V_i)$. В силу того, что числа $NUM(V_i)$ имеют разную длину в двоичном представлении необходимо хранить начало каждой сжатой части (смещение от начала строки), либо длины каждой сжатой части в отдельном массиве $BEGIN$ для обращения к нужному сжатому отрезку.

Совокупность всех сжатых частей образует сжатое представление последовательности W, которое обозначим через W'.

Лемма 1. *Длину сжатого набора можно оценить как*

$$l(W') \leqslant 2 \cdot \frac{N}{n} \cdot \log n + k \cdot \log \frac{3N}{k}.$$

Доказательство. Длина сжатой i-й части равна

$$2 \log |V_i| + \log(NUM(V_i)).$$

Если $|V_i| = 0$, то нам необходим один бит для хранения $|V_i|$, следовательно, в этом случае будем считать, что $\log |V_i| = 1$, а $|V_i| = 2$. Если $|V_i| = 1$, то также будем считать, что $|V_i| = 2$, так как требуется один бит для хранения $|V_i|$. В случае, когда $NUM(V_i) = 0$ или $NUM(V_i) = 1$ поступаем аналогично. Таким образом, получаем, что все k_i это целые числа больше единицы.

Воспользуемся оценкой из [1]

$$NUM(V) \leqslant C_{l(V)}^{|V|}.$$

Тогда,

$$l(W') = \sum_{i=1}^{T} (2 \log |V_i| + \log(NUM(V_i))) =$$

$$= 2 \log(k_1 \cdot k_2 \cdot \ldots \cdot k_T) + \log \left(\prod_{i=1}^{T} C_n^{k_i} \right).$$

Воспользуемся следующим неравенством [2]

$$C_{a_1}^{b_1} C_{a_2}^{b_2} \ldots C_{a_k}^{b_k} \leqslant C_{a_1+a_2+\ldots+a_k}^{b_1+b_2+\ldots+b_k}.$$

Получаем, что

$$\log\left(\prod_{i=1}^{T} C_n^{k_i}\right) \leqslant \log C_{nT}^{k_1+k_2+\ldots+k_T} \leqslant \log C_N^k.$$

Далее, используя известное неравенство

$$C_N^k \leqslant \left(\frac{3N}{k}\right)^k,$$

получаем

$$l(W') \leqslant 2\log\left(\prod_{i=1}^{T} k_i\right) + k \cdot \log\frac{3N}{k}.$$

Так как $k_1 + k_2 + \ldots + k_T = k$ и все k_i целые числа больше единицы, то можно воспользоваться оценкой

$$\prod_{i=1}^{T} k_i \leqslant \left(\frac{k}{T}\right)^T.$$

Следовательно,

$$l(W') \leqslant 2T\log\left(\frac{k}{T}\right) + k\log\frac{3N}{k}.$$

Учитывая, что $T = \lceil \frac{N}{n} \rceil$ и

$$\frac{k}{T} = \frac{kn}{N} \leqslant n,$$

так как $k \leqslant N$, получаем

$$l(W') \leqslant 2 \cdot \frac{N}{n} \cdot \log n + k \cdot \log\frac{3N}{k}.$$

Что и требовалось доказать.

Если в массиве $BEGIN$ хранятся длины сжатых частей, то длину этого массива можно оценить как

$$\log(l(W')) = \log\left(2 \cdot \frac{N}{n} \cdot \log n + k \cdot \log\frac{3N}{k}\right).$$

Оценим длину массива $BEGIN$, когда в нем хранится начало каждой сжатой части.

Лемма 2.

$$l(BEGIN) < \frac{N}{n} \cdot \log k + \frac{N}{n} \cdot \log\log(3n).$$

Доказательство. Так как в $BEGIN[p]$ хранится начало p-й части, то длину p-го элемента массива $BEGIN$ можно оценить как

$$l(BEGIN[p]) = \log\left(\sum_{i=1}^{p}(2\log|V_i| + \log NUM(V_i))\right) \leqslant$$

$$\leqslant \log\left(2\log(k_1k_2\ldots k_p) + \log\prod_{i=1}^{p} C_n^{k_i}\right).$$

По аналогии с леммой 1 оценим

$$2\log(k_1k_2\ldots k_p) + \log\prod_{i=1}^{p} C_n^{k_i} \leqslant$$

$$\leqslant 2\log\left(\frac{k_1+k_2+\ldots+k_p}{p}\right)^p + \log C_{np}^{k_1+k_2+\ldots+k_p} \leqslant$$

$$\leqslant 2\log\left(\frac{k_1+k_2+\ldots+k_p}{p}\right)^p + \log\left(\frac{3np}{k_1+k_2+\ldots+k_p}\right)^{k_1+k_2+\ldots+k_p} =$$

$$= \log\frac{(3n)^{k_1+k_2+\ldots+k_p}p^{k_1+\ldots+k_p-2p}}{(k_1+k_2+\ldots+k_p)^{k_1+\ldots+k_p-2p}}.$$

Заметим, что $k_1+k_2+\ldots+k_p \geqslant 2p$, так как $k_i \geqslant 2$. Далее получаем

$$\log\frac{(3n)^{k_1+k_2+\ldots+k_p}p^{k_1+\ldots+k_p-2p}}{(k_1+k_2+\ldots+k_p)^{k_1+\ldots+k_p-2p}} <$$

$$\log\left((3n)^k \cdot \left(\frac{p}{k_1+\ldots+k_p}\right)^{k_1+\ldots+k_p-2p}\right) <$$

$$< \log(3n)^k = k\log(3n).$$

Здесь учтено, что

$$\frac{p}{k_1 + \ldots + k_p} < 1,$$

так как $k_1 + \ldots + k_p \geqslant 2p$.

Следовательно,

$$l(BEGIN[p]) < \log(k \cdot \log(3n)).$$

Таким образом,

$$l(BEGIN) = \sum_{p=1}^{T} l(BEGIN[p]) < \sum_{p=1}^{T} \log(k \cdot \log(3n)) =$$

$$= \log \prod_{p=1}^{T} (k \cdot \log(3n)) == \log(k \cdot \log(3n))^T = T \log(k \cdot \log(3n)) =$$

$$= T \cdot (\log k + \log\log(3n)) \leqslant \frac{N}{n} \cdot \log k + \frac{N}{n} \cdot \log\log(3n).$$

Что и требовалось доказать.

Замечание 2. Полученные оценки завышены в силу неточности верхних оценок комбинаторных величин. Практические результаты степени сжатия информации должны получиться значительно лучше. На практике можно использовать оценки через величину C_N^k.

Исследуем самый худший случай сжатия информации.

Теорема 1. *Степень сжатия информации описанным выше методом в самом худшем случае можно оценить как*

$$N + \frac{N}{n} \cdot \left(2\log n + \log \frac{N}{2} + \log\log(3n) \right).$$

Доказательство. Общая оценка степени сжатия информации складывается из оценки длины сжатого набора W' и длины массива $BEGIN$. В лемме 1 доказано, что

$$l(W') \leqslant 2 \cdot \frac{N}{n} \cdot \log n + k \cdot \log \frac{3N}{k},$$

а в лемме 2 доказано, что

$$l(BEGIN) < \frac{N}{n} \cdot \log k + \frac{N}{n} \cdot \log\log(3n).$$

Следовательно, общую степень сжатия информации можно оценить как

$$2 \cdot \frac{N}{n} \cdot \log n + k \cdot \log \frac{3N}{k} + \frac{N}{n} \cdot \log k + \frac{N}{n} \cdot \log\log(3n).$$

Первое слагаемое в лемме 1 и второе слагаемое в лемме 2 не зависит от набора W, а зависит только от его длины. Как следует из замечания 1 можно ограничиться рассмотрением случая, когда $k \leqslant \left[\frac{N}{2}\right]$, поэтому первое слагаемое в лемме 2

$$\frac{N}{n} \cdot \log k \leqslant \frac{N}{n} \cdot \log \frac{N}{2}.$$

Известно, что для любых k выполнено

$$C_N^k \leqslant C_N^{\left[\frac{N}{2}\right]}.$$

Тогда худший случай для сжатия информации будет при $k = \left[\frac{N}{2}\right]$ и при распределении единиц по набору таким образом, что каждая часть разбиения длины n содержит $\left[\frac{n}{2}\right]$ единиц и имеет следующую структуру 00...011...1, то есть все единицы расположены в конце набора. В этом случае для второго слагаемого в лемме 1 получим

$$k \cdot \log \frac{3N}{k} \geqslant \frac{N}{2} \log 6 > N.$$

Это связано с грубостью оценки $C_N^k \leqslant \left(\frac{3N}{k}\right)^k$ при $k = \left[\frac{N}{2}\right]$, из которой следует

$$C_N^{\left[\frac{N}{2}\right]} \leqslant 6^{\left[\frac{N}{2}\right]}.$$

Однако можно доказать, что выполнено

$$C_N^{\left[\frac{N}{2}\right]} \leqslant 4^{\left[\frac{N}{2}\right]}.$$

Тогда второе слагаемое в лемме 1 не будет превышать N.

Следовательно, в самом худшем для сжатия случае мы получаем следующую оценку

$$N + 2 \cdot \frac{N}{n} \cdot \log n + \frac{N}{n} \cdot \log \frac{N}{2} + \frac{N}{n} \cdot \log\log(3n) =$$

$$= N + \frac{N}{n} \cdot \left(2\log n + \log \frac{N}{2} + \log\log(3n) \right).$$

Получается, что в этом случае произойдет увеличение, а не сжатие информации. Однако, как было отмечено в замечании 2, это связано с грубостью оценок комбинаторных величин.

Что и требовалось доказать.

2. Доступ к информации

Рассмотрим теперь возможность доступа к сжатым данным. Пусть требуется извлечь отрезок строки W длиной M бит, начиная с j-го бита.

Алгоритм 1.

1. Найдем номер части в сжатой последовательности W', которая хранит начало нужного нам отрезка или весь отрезок. Если нумерация частей начинается с единицы, то $p = \lceil \frac{j}{n} \rceil$.

2. Найдем номер бита с которого начинается отрезок в p-й части

$$s = j - \left[\frac{j}{n} \right] \cdot n.$$

3. Если отрезок не полностью находится в одной части, то есть $M > n - s + 1$, то вычислим сколько частей полностью содержат наш отрезок

$$R = \left\lceil \frac{M - (n - s + 1)}{n} \right\rceil + 1$$

и вычислим сколько бит находится в последней части

$$X = M - (n - s + 1) - \left[\frac{M - (n - s + 1)}{n} \right] \cdot n.$$

Заметим, что если $X = 0$, то наш отрезок включает последний бит последней части.

Если же $M < n - s + 1$, то просто полагаем $R = 1,\ X = 0$.

4. Считаем R частей, начиная с p-й из W'. Начало p-й части хранится в $BEGIN[p]$. Длину каждой сжатой части можно определить по массиву $BEGIN$. Например, длина p-й части равна $BEGIN[p+1] - BEGIN[p]$.

5. Применим процедуру 1 к каждой части. Отметим, что процедура начинает работу с конца сжатой части, поэтому для первой части нужно дойти только до s-го бита, а затем процедуру прервать.

6. Если часть одна, то извлекаем из нее M бит, начиная с s-го. Если частей несколько, то из первой извлекаем последние $n - s + 1$ бит, из последней извлекаем первые X бит, а из остальных частей все биты. Формируем нужный нам отрезок.

Оценим время доступа к информации с помощью этого алгоритма.

Теорема 2. *Количество операций алгоритма 1 можно оценить как*

$$O(R \cdot n^2) + \mathbf{extime}.$$

Доказательство. Первые три шага алгоритма требуют $O(1)$ операций. Четвертый шаг зависит от времени считывания информации. Оценим это время через величину **extime**. Если отрезок находится в одной части, то пятый шаг алгоритма потребует порядка $O(n-s+1)$ шагов процедуры 1. Если частей хотя бы две, то необходимо извлечь первые X бит из последней части, а для этого нужно полностью декодировать всю часть. Таким образом необходимо полностью декодировать $R-1$ часть. Это потребует $O(n)$ шагов процедуры 1 для каждой части. Следовательно, пятый шаг потребует порядка $O(n - s + 1) + O((R - 1)n)$ шагов процедуры 1. На каждом шаге необходимо выполнить порядка $O(n)$ операций, так как каждый раз потребуется вычислить некоторое число сочетаний, зависящее от n. Поэтому количество операций на пятом шаге можно оценить как

$$O(n \cdot (n - s + 1)) + O((R - 1) \cdot n^2).$$

Шестой шаг алгоритма потребует $O(1)$ операций.

Окончательно получаем, что общее количество операций алгоритма оценивается как

$$O(n \cdot (n-s+1)) + O((R-1) \cdot n^2) + O(1) + \mathbf{extime} \leqslant O(R \cdot n^2) + \mathbf{extime}.$$

Что и требовалось доказать.

Рассмотрим теперь процедуру замены информации в сжатой строке. Пусть в последовательности W, хранимой в сжатом виде, необходимо заменить отрезок строки длиной M бит, начиная с j-го бита, на новый отрезок такой же длины. Отметим, что при сжатии нового отрезка его длина может отличаться от длины старого сжатого отрезка. Поэтому при записи новой информации может потребоваться сдвигать остальные части. Будем считать, что время необходимое для этого мы учтем в величине **extime**.

Перед описанием алгоритма замены данных рассмотрим следующую задачу.

Пусть строка V разбита на две части $V = V'V''$ и для строки V' известно ее сжатое представление $(l(V'), |V'|, NUM(V'))$. Строка V' может быть неизвестна. Необходимо сжать строку V. Для этого не надо декодировать V', а затем сжимать всю строку V. Положим $t = l(V) = l(V') + l(V'')$, $k = |V| = |V'| + |V''|$. Для каждого бита ω строки V'', начиная с последнего, выполним:

Если $\omega = 1$, то $NUM(V) = C_{t-1}^{k} + NUM(V')$, t заменим на $t-1$, k заменим на $k-1$.

Если $\omega = 0$, то $NUM(V)$ не меняем, а t заменяем на $t-1$.

Тройка $(l(V), |V|, NUM(V))$ это сжатое представление строки V. Таким образом, можно сжать строку, зная сжатый вариант первой части строки и вторую часть строки.

Опишем теперь алгоритм замены информации в сжатой строке. Будем считать, что содержимое заменяемой части знать не обязательно. Если это не так, то алгоритм изменится не существенно.

Алгоритм 2.

1,2,3. Первые три шага полностью эквивалентны алгоритму 1.

4. Считаем из W' нужные нам части.

i) Если $s = 1$ и $X = 0$, то не надо ничего считывать.

ii) Если $s \neq 1$ и $X = 0$, то считать только первую часть. Ее начало в $BEGIN[p]$.

iii) Если $s = 1$ и $X \neq 0$, то считать только последнюю часть. Ее начало в $BEGIN[p + R - 1]$.

iv) Если $s \neq 1$ и $X \neq 0$, то считать первую и последнюю части.

Части, целиком вошедшие в нужный нам отрезок, считывать не надо, так как на их место будет записана новая информация.

5. Получим R новых сжатых частей, соответствующих старым частям.

i) Если $s = 1$ и $X = 0$, то разобьем новый отрезок на части по n бит каждая, вычислим функцию NUM и определим сжатый вид для каждой части.

ii) Если $s \neq 1$ и $X = 0$, то применим к первой части процедуру 1 до извлечения s-го бита. Запомним сжатое представление оставшихся $n - s - 1$ бит. Выберем первые $n - s + 1$ бит из нового отрезка. Для строки, состоящей из $s - 1$ бита старой части и $n - s + 1$ бит новой, найдем сжатое представление. Как было показано выше, для этого не потребуется декодировать первые $s-1$ бит. Оставшиеся $M-(n-s+1)$ бит разобьем на части по n бит и сожмем каждую часть.

iii) Если $s = 1$ и $X \neq 0$, то применим процедуру 1 к последней части до извлечения всех бит. Заменим первые X бит на последние X бит из нового отрезка и сожмем полученную последовательность. Это будет последняя из новых сжатых частей. Для получения первых $R - 1$ новых частей разобьем оставшиеся $M - X$ бит нового отрезка на части по n бит и сожмем каждую.

iv) Если $s \neq 1$ и $X \neq 0$, то получим случай, аналогичный ii) и iii). Первые $n - s + 1$ бит из нового отрезка сожмем как в ii), а последние X бит сожмем как в iii). Оставшиеся $M-(n-s+1)-X$ бит разобьем на части по n бит и сожмем каждую из них.

6. Обозначим все сжатые части через $V_1, \ldots, V_R$, а их длины через $l(V_1), \ldots, l(V_R)$. Так как длины новых сжатых частей могут отличаться от старых, то необходимо изменить массив $BEGIN$. Это можно сделать с помощью следующих операторов:

$Z := BEGIN[p + R]$**;**
for $I := 1$ **to** R **do** $BEGIN[p + I] := BEGIN[p + I - 1] + l(V_I)$**;**
for $I := p + R + 1$ **to** T **do**
{ $Y := BEGIN[I] - Z$; $Z := BEGIN[I]$**;**
$BEGIN[I] := BEGIN[I - 1] + Y$**;**
}

7. Запишем новые сжатые части, начиная с p-й. Для этого используем массив $BEGIN$. Части, начиная с $(p + R)$-й, необходимо сдвинуть в соответствии с новыми длинами.

Теорема 3. *Количество операций алгоритма 2 можно оценить как*

$$O(R \cdot n^2) + O\left(\frac{N}{n}\right) + \mathbf{extime}.$$

Доказательство. Первые три шага требуют $O(1)$ операций. Время выполнения четвертого шага можно оценить через **extime**. В эту оценку можно включить и седьмой шаг алгоритма. Рассмотрим подробнее пятый шаг.

i) Необходимо вычислить функцию NUM для R частей длиной n бит. Для одной части эта функция находится за n шагов. Но на некоторых шагах необходимо вычислять число сочетаний (если бит равен единице). Если единиц k, то количество операций для одной части оценивается как $O(n \cdot k + n - k) \leqslant O(n^2)$. Таким образом, для R частей получаем $O(R \cdot n^2)$ операций.

ii) Необходимо декодировать $n - s + 1$ бит и затем вычислить для такого же количества бит функцию NUM. Это потребует $O(n \cdot (n - s + 1))$ операций. Далее необходимо сжать $R - 1$ частей по n бит каждая. По аналогии с i) заключаем, что потребуется $O((R-1) \cdot n^2)$ операций.

Аналогично можно доказать, что пункт iii) потребует $O(R \cdot n^2)$ операций, а пункт iv) потребует $O(n \cdot (n - s + 1)) + O((R - 1) \cdot n^2)$ операций.

Шестой шаг требует $O(T - p) \leqslant O(T) \leqslant O(\frac{N}{n})$ операций.

Окончательно получаем, что количество операций алгоритма можно оценить как

$$O(1) + O(R \cdot n^2) + O(n \cdot (n - s + 1)) + O((R - 1) \cdot n^2) + \\ + O\left(\frac{N}{n}\right) + \mathbf{extime} \leqslant O(R \cdot n^2) + O\left(\frac{N}{n}\right) + \mathbf{extime}.$$

Что и требовалось доказать.

Список литературы

[1] Andreev A.E., Clementi A.E.F., Rolim J.D.P. Hitting sets derandomize BPP // Lecture Notes in Computer Science. 1996. V. 1099. P. 357–368.

[2] Нечипорук Э.И. О топологических принципах самокорректирования // Проблемы кибернетики. 1969. **21**. С. 5–103.