ПАРАЛЛЕЛЬНАЯ ГЕНЕРАЦИЯ СОЧЕТАНИЙ И ПЕРЕСТАНОВОК

Н.Е. Тимошевская

Томский государственный университет

Предлагается метод параллельной генерации перестановок и сочетаний, ориентированный на использование многопроцессорных систем. В основе метода лежит возможность нумерации перечисляемых объектов так, что по номеру можно легко восстановить сопоставленный ему объект. Построенная таким образом нумерация позволяет разбить все множество объектов на классы, мощности которых отвечают производительности процессоров в системе, и генерировать объекты различных классов независимо и параллельно на соответствующих процессорах.

1. Введение

При решении многих задач возникает необходимость перебора или упорядоченного перечисления комбинаторных объектов определенного типа – подмножеств заданного множества, перестановок его элементов, размещений, подстановок, разбиений множеств и других подобных объектов. Порождение объектов указанных типов, как правило, трудоемко по времени, так как в реальных задачах их число очень велико. Применение параллельных вычислений позволяет сократить время перебора.

2. Генерация сочетаний

Здесь под сочетанием понимается всякое упорядоченное по возрастанию k-элементное подмножество n-элементного множества $N = \{1,2,...,n\}$, т.е. любой такой вектор $(b_1, b_2, ..., b_k)$ длины $k \le n$ с компонентами в N, называемый также сочетанием из n по k, в котором $b_i < b_{i+1}$ для $i = 1,...,k-1$, а под генерацией сочетаний – перечисление всех таких векторов. Известны алгоритмы, в которых сочетания порождаются в лексикографическом порядке [1], т.е. так, что сочетание $(b_1, b_2, ..., b_k)$ предшествует сочетанию $(a_1, a_2, ..., a_k)$, если для некоторого $i, 1 \le i \le k$, имеет место $b_1 = a_1, b_1 = a_1, ..., b_i = a_i, u\ b_{i+1} < a_{i+1}$.

Например, все сочетания из 5 по 3 перечисляются в следующем порядке: (1, 2, 3), (1, 2, 4), (1, 2, 5), (1, 3, 4), (1, 3, 5), (1, 4, 5), (2, 3, 4), (2, 3, 5), (2, 4, 5), (3, 4, 5).

С целью распараллеливания генерации сочетаний введем понятие индекса сочетания и покажем, как сочетание строится по его индексу. Индексом сочетания назовем порядковый номер I этого сочетания в последовательности всех сочетаний, расположенных в лексикографическом порядке, $1 \le I \le C_n^k$. Каждому индексу I из указанного диапазона соответствует одно сочетание и наоборот.

Рассмотрим теперь, как сочетание $(b_1, b_2, ..., b_k)$ строится по его номеру I. Пусть $1 \le t \le k$ и $0 = b_0 < b_1 < b_2 < ... < b_{t-1}$. Для любого j в $\{b_{t-1} + 1, b_{t-1} + 2, ..., n - k + t\}$, через s_j^t обозначим число всех сочетаний с префиксами $(b_1, b_2, ..., b_{t-1}, r)$, где $r \in \{b_{t-1} + 1, b_{t-1} + 2, ..., j - 1\}$. В каждом из этих сочетаний недостающие $k - t$ элементов берутся из $n - r$ элементов $r + 1, ..., n$, поэтому

$$s_j^t = \sum_{r=b_{t-1}+1}^{j-1} C_{n-r}^{k-t}.$$

Положим также $s^t = \sum_{i=1}^{t-1} s_{b_i}^i$. Это есть число всех сочетаний с префиксами $(b_1, b_2, ..., b_{i-1}, a_i)$ для $i \le t - 1$ и $a_i < b_i$, т.е. всех сочетаний, предшествующих сочетаниям с префиксом $(b_1, b_2, ..., b_{t-1})$. Отметим, что величины s_j^t и s_t не убывают с ростом j и t соответственно.

Например, для сочетаний из 5 по 3 при $t = 1$: s_1^1=0, s_2^1=6, s_3^1=9; при $t = 2$ и $b_1 = 1$: s_2^2=0, s_3^2=3, s_4^2=5; при $b_1 = 2$: s_3^2=0, s_4^2=2; при $t = 3$ и (b_1, b_2) = (1, 2): s_3^3=0, s_4^3=1, s_5^3=2. Кроме того, s^1=0; при $t = 2$ и $b_1 = 1$: s^2=0, при $b_1 = 2$: s^2=6, при $b_1 = 3$: s^2=9; при $t = 3$ и (b_1, b_2) = (1, 2): s^3=0, при (b_1, b_2) = (1, 4): s^3 =5, при (b_1, b_2) = (2, 3): s^3=6.

Теорема 1. Пусть $1 \le t \le k$ и $(b_1, b_2, ..., b_{t-1})$ есть префикс сочетания с индексом I; тогда t-й элемент b_t этого сочетания равен наименьшему из таких j, что $b_{t-1} < j < n - k + t$ и $I \le s^t + s_{j+1}^t$, если такие j существуют, и $b_t = n - k + t$ в противном случае.

Пример. Следуя методу данной теоремы, построим сочетание из 5 по 3 с индексом I=9.

$t = 1$: s^t=0; j=1, $s_{J+1}^t = s_2^1$=6, $s^t + s_{J+1}^t$=6< I; j=2, s_3^1=9, $s^t + s_{J+1}^t$=9≥ I; следовательно, b_1=2.

$t = 2$: $s^t = s_2^1$=6; j=3, $s_{J+1}^t = s_4^2$=2, $s^t + s_{J+1}^t$=8< I; j=4=$n - k + t$; следовательно, b_2=4.

$t = 3$: не существует j для 4=b_{t-1}< j <$n - k + t$=5, поэтому $b_3 = n - k + t$=5.

В результате получаем сочетание (2, 4, 5).

Данный метод вычисления сочетания из n по k по его индексу I можно выразить следующим алгоритмом.

Алгоритм 1

1. $b_0=0$, $s=0$, $t=1$.

2. Если $t \le k$, то п.3, иначе п.8.

3. $j = b_{t-1}+1$.

4. Если $j<n-k+t$ и $s+C_{n-j}^{k-t}<I$, то п.5, иначе п.6.

5. $s = s+C_{n-j}^{k-t}$; $j=j+1$; п.4.

6. $b_t=j$.

7. $t=t+1$; п.2.

8. $(b_1,b_2,...,b_k)$ – искомое сочетание.

Реализация алгоритма 1 предполагает использование «больших» чисел (целых чисел, двоичное представление которых требует большего числа разрядов, чем в стандартных типах) для задания индекса и подсчета числовых значений C_n^k. Вместе с тем, опираясь на него, можно предложить следующий алгоритм параллельной генерации сочетаний.

Пусть требуется построить все сочетания из n по k на однородной многопроцессорной вычислительной системе, состоящей из p одинаковых процессоров с общей или распределенной памятью. Максимальную загруженность процессоров можно обеспечить, разделив последовательность всех сочетаний на блоки по $s = [C_n^k / p]$ элементов в каждом. Блок с номером m начинается с сочетания, соответствующего индексу $I = s(m-1)+1$, и генерация сочетаний в нем возлагается на m-й процессор. Если число всех сочетаний не кратно p, то процессор, генерирующий сочетания последнего блока, строит также $C_n^k \bmod p$ оставшихся сочетаний, число которых меньше p. При выполнении этого алгоритма отсутствует взаимодействие между процессорами и все процессоры одинаково загружены, что позволяет добиться ускорения, близкого к p. В случае использования неоднородной системы равномощные блоки сочетаний генерируются не за одинаковое время, что приводит к простаиванию процессоров и падению эффективности алгоритма. В этом случае все множество сочетаний следует разбивать на блоки, мощности которых отвечают производительности процессоров в системе.

3. Генерация перестановок

Требуется построить все перестановки $(a_1,a_2,...,a_n)$ из элементов множества $N=\{1,2,...,n\}$, т.е. перестановки степени n. Известны алгоритмы, перечисляющие все перестановки в лексикографическом порядке, начиная с перестановки $(1,2,...,n)$ [2], т.е. так, что перестановка $(a_1,a_2,...,a_n)$ предшествует перестановке $(b_1,b_2,...,b_n)$, если для некоторого $i, 1 \le i \le k$, имеет место $b_1 = a_1, b_1 = a_1, ..., b_i = a_i, u\ b_{i+1} < a_{i+1}$.

Например, перестановки степени 3 перечисляются в следующем порядке: 123, 132, 213, 231, 312, 321.

Назовем индексом перестановки ее порядковый номер I в последовательности всех перестановок степени *n*, расположенных в лексикографическом порядке, $1 \le I \le n!$. Так перестановка 123 имеет индекс 1, перестановка 132 – индекс 2, перестановка 321 – индекс 6. Покажем, каким образом перестановку можно вычислить по ее индексу.

Пусть $0 \le t \le n$ и $A = (a_1, a_2, ..., a_n)$ есть префикс перестановки с индексом I. Обозначим J_t число перестановок, предшествующих всем перестановкам с префиксом A, и положим $M_t = I - J_t$. Назовем M_t кратностью префикса A. По определению $M_0 = I$ и M_t есть число перестановок с префиксом A, индексы которых не превосходят I. Пусть $Y = \{y_1, ..., y_{n-t+1}\} = N - \{a_1, ..., a_{t-1}\}$ и $y_1 < ... < y_{n-t+1}$. Тогда по свойству перестановки $a_t \in Y$. Допустим, что $a_t = y_k$, $1 \le k \le n - t + 1$. Тогда $A = (a_1, a_2, ..., a_{t-1}, y_k)$ и $J_t = J_{t-1} + (k-1)(n-t)!$, поэтому $M_t = I - J_t = I - J_{t-1} - (k-1)(n-t)!$. Таким образом, зная индекс перестановки и любой ее префикс, можно вычислить кратность последнего.

Пример. Пусть n=3, I=4. Тогда

1) если $t = 0$ (A пусто), то M_0=4;

2) если $t = 1$ и A=2, то $Y = \{1,2,3\}$, $k = 2$ и $M_1 = M_0 - (k-1)(n-1)! = 2$;

3) если $t = 2$ и A=23, то $Y = \{1,3\}$, $k = 2$ и $M_2 = M_1 - (k-1)(n-2)! = 1$.

Теорема 2. Пусть $1 \le t \le n-1$, $(a_1, ..., a_{t-1})$ – префикс перестановки с индексом I и M_{t-1} – его кратность. Пусть также $\{y_1, ..., y_{n-t+1}\} = N - \{a_1, ..., a_{t-1}\}$ и $y_1 < ... < y_{n-t+1}$. Тогда t-й элемент a_t в перестановке с индексом I есть y_k для

$$k = \left[\frac{M_{t-1} + (n-t)! - 1}{(n-t)!} \right].$$

При известном префиксе $(a_1, ..., a_{n-1})$ перестановки с индексом I ее последний член находится как единственный элемент множества $N - \{a_1, ..., a_{n-1}\}$.

Пример. Построим перестановку (a_1, a_2, a_3) по индексу $I = 4$.

$t = 1$: $M_0 = 4$, $Y = \{1,2,3\}$, $k = 2$, следовательно, $a_1 = 2$.

$t = 2$: $M_1 = M_0 - (k-1)(n-1)! = 2$, $Y = \{1,3\}$, $k = 2$,

следовательно, $a_2 = 3$.

$t = 3$: $Y = \{1\}$, следовательно, $a_3 = 1$.

В результате получаем перестановку (2, 3, 1).

Алгоритм 2 (вычисление перестановки $(a_1, ..., a_n)$ по индексу I)

1. $t = 1$, $Y = \{y_1, ..., y_n\} = \{1,2,...,n\}$.

2. Если $t = n$, то п.5; иначе $k = \left[\frac{I + (n - t)!}{(n - t)!} \right]$ и

3. $a_t = v_k$, $Y = \{y_1, ..., y_{n-t}\} = Y - \{y_k\}$.

4. $I = I-(k-1)(n-t)!$, $t = T + 1$, перейти к п.2.

5. $a_n = v_1$.

Параллельная генерация перестановок осуществляется так же, как и сочетаний, а именно все множество перестановок делится на блоки, мощности которых отвечают производительности процессоров в системе; в каждом блоке по алгоритму 2 с помощью индекса вычисляется первая перестановка, начиная с которой один из процессоров системы генерирует перестановки всего блока в лексикографическом порядке. Как и в случае сочетаний, этот параллельный алгоритм обладает высокой эффективностью благодаря равномерному распределению вычислений и отсутствию обменов данными между процессорами.

Литература

1. Липский В. Комбинаторика для программистов. – М.: Мир, 1988. – 200 с.
2. Дейкстра Э. Дисциплина программирования / Под ред. Э.З. Любимского. – М.: Мир, 1978. – 280 с.