Информационные процессы, Том 3, № 1, 2003, стр. 39–46.

ТЕОРИЯ И МЕТОДЫ ОБРАБОТКИ ИНФОРМАЦИИ

Некоторые алгоритмы, связанные с конечными группами

В.А. Любецкий[*], **А.В. Селиверстов**[**]

[*]*Институт проблем передачи информации РАН,*
101447, Россия, Москва, Большой Каретный переулок, 19,
e-mail: lyubetsk@iitp.ru
[**]*Государственная Классическая Академия им.Маймонида, Россия, Москва,*
e-mail: slvstv@ipc.ru

Поступила в редколлегию 15.12.2002

Аннотация—В работе рассматриваются алгоритмы для вычисления порядка, системы порождающих и других характеристик пересечения подгрупп конечной группы. Они применяются к задаче решения однородных линейных уравнений, например, над конечным бимодулем. Рассмотрен также эвристический алгоритм поиска подграфа с большим числом рёбер в нём.

1. ВВЕДЕНИЕ

В [1] описан полиномиальный алгоритм решения систем линейных уравнений над кольцом вычетов целых чисел. В следующих пунктах 2, 3 приведены алгоритмы (второй из них — в сущности известный алгоритм Гаусса) решения таких систем над произвольной конечной абелевой группой с (левыми и правыми) операторами, действующими на ней. Первый из этих алгоритмов решает и неявно заданные системы, когда имеется только процедура, проверяющая выполнимость любого уравнения из системы при данных значениях переменных.

Затем (в пункте 4) приводится эвристический, но полезный в биоинформатике, алгоритм поиска плотных подграфов в данном графе.

2. АЛГОРИТМ ПОЛИНОМИАЛЬНОГО ВРЕМЕНИ ДЛЯ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ ОДНОРОДНЫХ УРАВНЕНИЙ НАД КОНЕЧНЫМ БИМОДУЛЕМ

Пусть A, B — некоторые кольца, а R — конечный A-B-бимодуль из r элементов. В частности, любое ассоциативное (но, вообще говоря, не коммутативное и без единицы) кольцо R является R-R-бимодулем. Рассматривается произвольная система однородных линейных уравнений вида

$$\sum_{k,j} a_{kj} x_k b_{kj} = 0,$$

над бимодулем R.

Множество решений системы из m таких уравнений, каждое от n переменных образует подгруппу в R^n, которую будем обозначать B_m.

Отметим два простых факта.

Лемма 1. *Пусть G — конечная абелева группа, H и K — её подгруппы. Индекс $(K : K \cap H)$ не превышает индекса $(G : H)$.*

Лемма 2. *Множество решений в R^n любого однородного линейного уравнения — подгруппа индекса не больше r, где r — порядок группы R.*

Доказательство. Рассмотрим линейное уравнение $f = 0$ в R^n. Число его решений равно числу решений системы из двух уравнений $f = z$ и $z = 0$ в R^{n+1}, где переменная z новая, не встречается в f. Уравнение $f = z$ выделяет в R^{n+1} подгруппу H индекса r, поскольку любое значения переменных $x_1, \ldots, x_n$ определяют ровно одно значение переменной $z = f(x_1, \ldots, x_n)$; индекс подгруппы H равен уравнения $r^{n+1}/r^n = r$. Уравнение $z = 0$ выделяет в R^{n+1} подгруппу $K = R^n$.

По лемме 1 $(K : K \cap H) \leq r$, т.е. исходное уравнение $f = 0$ выделяет в R^n подгруппу индекса не больше r.

Теорема. *Существует и ниже описан алгоритм, который для произвольной линейной однородной системы уравнений над R образует систему порождающих в группе B_m всех её решений и указывает число всех решений, выполняя не более $C \cdot (m + n)^2 \cdot m \cdot L \cdot r^4$ операций в R. Здесь m — число уравнений, n — число переменных, L — число операций, необходимых для проверки выполнимости любого уравнения системы при любых значениях переменных, r — число элементов в R, а C — некоторая константа.*
Алгоритм применим и к системе неявно заданных уравнений, когда коэффициенты уравнений не известны, но дана эффективная процедура для проверки выполнимости любого уравнения системы при любых значениях переменных. (Процедура говорит "да"или "нет".)

Доказательство. Образуем цепь коммутативных подгрупп

$$B_m \subseteq B_{m-1} \subseteq \cdots \subseteq B_2 \subseteq B_1 \subseteq B_0 = R^n,$$

где B_i — множество решений первых i уравнений системы.

Обозначим через A_i множество решений i-го уравнения этой системы. Очевидно, подгруппа

$$B_i = B_{i-1} \cap A_i.$$

По лемме индекс B_i в B_{i-1} не больше индекса A_i в R^n, который не превосходит r.

Продолжим предыдущую цепь групп

$$\{0\} = B_{m+n} \subseteq B_{m+n-1} \subseteq \cdots \subseteq B_{m+1} \subseteq B_m \subseteq B_{m-1} \subseteq \cdots \subseteq B_2 \subseteq B_1 \subseteq B_0 = R^n,$$

полагая для всех $i < n$ подгруппу

$$B_{m+i} = B_{m+i-1} \cap C_i,$$

где C_i — подгруппа в R^n, определяемая уравнением $x_i = 0$. Индекс $(R^n : C_i) = r$. По лемме индекс B_{m+i} в B_{m+i-1} не больше r.

Итак, для всей цепи групп индекс $(B_{i-1} : B_i)$ не больше r.

Представим группу B_i как дизъюнктное объединение

$$B_i = \sum_d B_{i,d} + B_{i+1},$$

где $B_{i,d}$ — все смежные классы группы B_i по подгруппе B_{i+1} за исключением класса B_{i+1} нуля. (Интуитивно говоря, это — представление B_i в виде суммы с точностью до B_{i+1}.) Тогда

$$B_j = \sum_{i=j}^{m+n-1} \sum_d B_{i,d}.$$

Ясно, что B_j порождается любым набором представителей классов $B_{i,d}$. Напомним, что нас интересует случай $j = m$, когда набор представителей порождает (в виде сумм) как раз группу B_m всех решений исходной системы.

Фиксируем в $B_0 = R^n$ тривиальную систему порождающих, состоящую из $(r-1)n$ элементов вида

$$(0, \ldots, 0, x, 0, \ldots, 0),$$

где $x \neq 0$.

Ниже описан алгоритм, который строит множества W_i представителей (не обязательно всех) смежных классов в группе B_i по подгруппе B_{i+1}. При этом из каждого смежного класса выбирается не более одного представителя, а представителем класса B_{i+1} нуля всегда берётся 0.

Сначала опишем вспомогательную процедуру П. Эта процедура получает на вход ненулевой элемент $g \in R^n$ и набор множеств $\{W_i\}$. Если g не удаётся разложить в сумму элементов из множеств W_i, то процедура П добавляет к множествам W_i новые элементы так, чтобы указанное разложение стало возможным.

Описание процедуры П.

1. Ищем наибольшее k, для которого $g \in B_k$.
2. Перебирая элементы множества W_k, ищем такой элемент $x \in W_k$, что $(g-x) \in B_{k+1}$. Если такого x нет, то присоединяем сам элемент g к множеству W_k. В ином случае, когда такой x найден, применяем процедуру П к элементу $g - x$.

Проверка принадлежности элемента g к группе B_k сводится к проверке k равенств, что требует $O(kL)$ операций. Если $k > m$, то эта проверка требует $O(mL)$ операций в исходном бимодуле R. Итак, процедура П требует $O((m+n)mLr)$ операций.

Описание основного алгоритма.

1. Полагаем $W_i = \{0\}$ для всех i.
2. Последовательно применяем процедуру П ко всем элементам указанной выше тривиальной системы порождающих группы R^n.
3. Для каждого i начиная от 0 и до $m+n-1$ применяем процедуру П ко всем ненулевым элементам вида $x+y$, где $x, y \in W_i$, до тех пор, пока множество W_i не перестанет пополняться новыми элементами. Точнее, пока для W_i не перестанут появляться новые пары $\langle x, y \rangle$.

Выполнение пункта 2 требует $O((m+n)mLr^2n)$ операций. Поскольку множество W_i содержит не более r элементов, то при каждом i просматривается не более r^2 пар элементов. Заметим, что применение процедуры П к суммам элементов из W_i не добавляет новых элементов в W_j при $j < i$. Итак, выполнение пункта 3 требует $O((m+n)^2mLr^3n)$ операций.

Элемент g из R^n, входящий в B_k и не входящий в B_{k+1}, назовём *элементом ранга* k.

Перейдем к обсуждению алгоритма. Он строит множество W_i представителей (не обязательно всех) смежных классов в группе B_i по подгруппе B_{i+1}, для всех i. Класс B_{i+1} всегда представляется элементом 0. Элементы множества W_i (кроме 0) имеют ранг i. Объединение множеств $\cup_{i=0}^{m+n-1} W_i$ порождает группу R^n.

Более того, каждый элемент $g \in R^n$ ранга k представим в виде суммы элементов из множества $\cup_{i=k}^{m+n-1} W_i$,

$$g = x_1 + \cdots + x_j,$$

где x_1 ранга k, и ранги всех элементов $x_1, \ldots, x_j$ строго возрастают.

Доказательство очевидно из двойной индукции по рангу и числу элементов одного ранга. Действительно, если $g = y_1 + \cdots + y_\ell$ — некоторое представление элемента g ранга k (пусть ранги элементов $y_1, \ldots, y_\ell$ не убывают), то y_1 имеет ранг k, и для каждого множества элементов одинакового ранга,

начиная с наименьшего ранга, заменим (пусть y_1 и y_2 одинакового ранга) сумму $y_1 + y_2$ на равную ей сумму элементов (из множества $\cup_{i=k}^{m+n-1} W_i$) попарно различных рангов. Тогда число элементов ранга k в правой части исходного равенства уменьшится на 1.

Отсюда сразу следуют такие свойства построенной системы множеств W_i.

1. Множество W_{m+n-1} равно B_{m+n-1}.
2. Элемент g ранга k (т.е. $g \in B_k \setminus B_{k+1}$) представим как сумма элементов из множества $\cup_{i=k}^{m+n-1} W_i$. В частности, это множество порождает группу B_k.
3. Множество W_i содержит по одному представителю каждого смежного класса в группе B_i по подгруппе B_{i+1}. Действительно, для любого смежного класса $[g]$ в B_i имеем $g = x_1 + \cdots + x_j$, и представителем класса $[g]$ является x_1.
4. Множество $\cup_{i=m}^{m+n-1} W_i$ порождает группу B_m, которая является искомым множеством решений системы линейных уравнений.
5. Построение системы множеств W_i требует выполнения

$$O((m+n)^2 m L r^4)$$

операций в исходном бимодуле R.
6. Число элементов в множестве B_m равно произведению мощностей множеств

$$W_m, \ldots, W_{m+n-1}.$$

Заметим, что без всяких изменений этот алгоритм и соответствующее доказательство переносятся на произвольную конечную абелеву группу с конечными множествами произвольных (левых и правых) операторов, действующих на ней.

3. СЛУЧАЙ ЯВНО ЗАДАННЫХ ЛИНЕЙНЫХ СИСТЕМ

Приведём здесь следующий известный алгоритм (ср. [2], глава 2, § 2.), по существу — алгоритм Гаусса для решения, как и в пункте 2, однородной системы из m линейных уравнений от n неизвестных вида

$$\sum_{k,j} a_{kj} x_k b_{kj} = 0$$

над A-B-бимодулем R из r элементов. В отличие от пункта 1 здесь нужно считать известными коэффициенты уравнений; точнее, — левые части всех уравнений, составляющих систему.

Фиксируем какое-то семейство из не более чем $\log_2 r$ порождающих в группе R. Это семейство тривиально возникает из цепочки подгрупп, порождаемых указанными множествами:

$$\{g_1\} \subset \{g_1, g_2\} \subset \cdots \subset \{g_1, \ldots, g_r\} = R,$$

где $\{g_1, \ldots, g_r\}$ — все элементы группы R.

Тогда фиксируем в R^n систему порождающих $\{w_i\}$, состоящую из μ элементов вида

$$(0, \ldots, 0, x, 0, \ldots, 0),$$

где x — одна из выше полученных порождающих в R. Число μ не превосходит $n \log_2 r$.

Очевидно, любой элемент группы R^n представим в виде суммы $\sum_i c_i w_i$, где c_i — строго положительные целые числа.

Сначала опишем процедуру поиска порождающих в группе B_1 всех решений какого-то одного, например, первого в системе уравнения над R. Подставляя выражение

$$\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \sum_{i=1}^{\mu} c_i w_i$$

в это уравнение, получим соотношение, которому должны удовлетворять коэффициенты c_i. Это соотношение имеет вид

$$\sum_{i=1}^{\mu} c_i g_i = 0, \tag{1}$$

где g_i — некоторые элементы группы R, а числа c_i — искомые неизвестные, $0 < c_i \leq r$. Вычисление элементов g_i требует $O(L\mu)$ операций, где L — число операций в уравнении.

Теперь найдём наименьшее строго положительное число c_1, при котором соотношение (1) выполняется при каких-то $c_2, \ldots, c_\mu$. Очевидно, такое c_1 существует и не превышает r. Для этого будем перебирать целые числа ℓ от 1 до r в порядке их возрастания, проверяя для каждого ℓ условие: принадлежит ли элемент $(-\ell \cdot g_1)$ множеству элементов из R вида $c_2 g_2 + \cdots + c_\mu g_\mu$ с произвольными c_i, где $0 < c_i \leq r$. Конечно, это множество содержит не более r элементов, а это условие заведомо выполняется при каком-то $\ell \leq r$.

Важно, что множество A всех элементов вида $c_2 g_2 + \cdots + c_\mu g_\mu$ находится за $O(r^2\mu)$ шагов. Действительно, множество A можно фактически строить, начиная с множества $\{0\}$, путём включения в него новых элементов: последовательно на каждом шаге для всех $i = 2, \ldots, \mu$ и всех уже включённых элементов (таковых не больше r) в множество включаем суммы (в группе R) всех его элементов и каждого g_i (если эта сумма не совпадает с ранее включёным элементом); такие шаги выполняются r раз (иными словами, до тех пор, пока в множестве A не перестанут появляться новые элементы). Таким образом, выполнив $O((L + r^2)\mu)$ операций в бимодуле R, найдём c_1 и соответствующие ему $c_2, \ldots, c_\mu$. А, следовательно, найдём и какое-то одно решение исходного уравнения.

Затем будем искать следующее решение этого уравнения уже вида $b_2 w_2 + \cdots + b_\mu w_\mu$, где коэффициент b_2 — наименьшее строго положительное число.

Повторяя эту процедуру μ раз мы найдём систему порождающих группы решений данного уравнения, что потребует выполнения $O((L + r^2)\mu^2)$ операций в бимодуле R. Легко проверить, что так полученная система векторов

$$\begin{cases} V_1 = c_1 w_1 + c_2 w_2 + \cdots + c_\mu w_\mu \\ V_2 = \quad b_2 w_2 + \cdots + b_\mu w_\mu \\ \vdots \\ V_\mu = \quad d_\mu w_\mu \end{cases}$$

из B_1 является системой порождающих в B_1.

Чтобы решить систему многих уравнений над R нужно последовательно выделять в группе B_k решений первых k уравнений подгруппу, определяемую $(k+1)$-м уравнением. В целом зто потребует $O(m(L + r^2)\mu^2)$ операций.

Учитывая, что $\mu \leq n \log_2 r$, получим оценку числа операций в R, необходимых для поиска порождающих подгруппы в R^n всех решений системы из m уравнений от n неизвестных:

$$O(m(L + r^2)(n \log_2 r)^2).$$

4. ПОИСК ПЛОТНЫХ ПОДГРАФОВ В ИСХОДНОМ МНОГОДОЛЬНОМ ГРАФЕ

Рассмотрим неориентированный граф Γ без петель и обозначим V число его вершин, E число его рёбер. Граф задаётся матрицей смежности: симметричной $0,1$ значной матрицей A порядка V, на главной диагонали которой стоят только нули.

Граф называется *n-дольным* (*многодольным*), если каждая его вершина окрашена в один из n цветов так, что все рёбра имеют разноцветные концы. Множество вершин одного цвета называется *долей*. Напомним, что подграф называется *кликой*, если в нём соединены ребром любые две его вершины. *Треугольником* назовём клику с тремя вершинами, *тетраэдром* — клику с четырьмя вершинами.

Подграф Γ' в фиксированном n-дольном графе Γ назовём (a,b,c)-*плотным*, если:

1. Все вершины в Γ' принадлежат разным долям.
2. Каждая вершина из Γ' соединена рёбрами с не менее чем a долями графа Γ'.
3. Существует вершина в Γ', которая смежна всем остальным вершинам в Γ'.
4. Каждое ребро в Γ' принадлежит не менее чем b треугольникам и не менее чем c тетраэдрам в Γ'.

Заметим, что клика с q вершинами является $(q-1, q-2, (q-2)(q-3)/2)$-плотным подграфом.

Для любых двух матриц M и N одинакового размера *произведением Адамара* $M * N$ называется матрица, элементы которой — произведения соответствующих элементов матриц M и N; эта операция коммутативна.

Тривиальный алгоритм умножения квадратных матриц порядка k требует $O(k^3)$ операций (в кольце целых чисел $\mathbb{Z}$). Известны алгоритмы умножения матриц, которые требуют $O(k^\omega)$ операций, где $\omega < 2,376$.

Рассмотрим матрицу $B = A * (A^2)$, вычисление которой требует $O(V^\omega))$ операций. Ее элемент, не лежащий на главной диагонали, равен числу треугольников, содержащих соответствующее ребро. Отсюда для каждой из вершин найдём число (неупорядоченных) пар различных треугольников с общим ребром, содержащих эту вершину. На главной диагонали матрицы A^3 находятся удвоенные числа треугольников, содержащих соответствующую вершину.

Все треугольники перечисляются за $O(VE)$ шагов. Поэтому при небольшом числе рёбер прямой перебор быстрее, чем использование матрицы B. Тетраэдры определяются парой скрещивающихся рёбер, поэтому прямым перебором за $O(E)$ шагов перечисляются все тетраэдры, содержащие какое-то фиксированное ребро.

Приведём алгоритм, требующий $O(E^3)$ операций, который находит (не обязательно все) (a,b,c)-плотные подграфы с числом вершин $q > 3$ в n-дольном графе Γ. Этот алгоритм не всегда находит даже один плотный подграф, когда он существует.

Алгоритм состоит в следующем: последовательно удаляются вершины и рёбра графа Γ в соответствии с ниже указанной цепочкой шагов.

Если на некотором шаге удалена хотя бы одна вершина или одно ребро, то переходят к инструкции 1. В ином случае — к следующей инструкции. В процессе работы алгоритма составляется список Δ подграфов, и шаги продолжаются до тех пор, пока не будут удалены все вершины, не входящие в какой-либо из подграфов, включённых в список Δ. Общее число шагов алгоритма не более $O(E)$. Граф, который получается из исходного графа Γ удалением некоторого числа вершин и рёбер, обозначается G. Сначала полагаем $G = \Gamma$.

Алгоритм.

1. Удаляем из G все вершины, которые соединены рёбрами с менее чем a долями графа G.
2. В графе G перебором находим все вершины степени $q-1$. Для каждой такой вершины проверяем, является ли подграф, состоящий из нее и всех смежных с ней вершин, плотным. Если так, то

заносим его в список Δ уже найденных подграфов. Затем удаляем все вершины степени $q-1$, которые не вошли в один из так полученных плотных подграфов.

3. Удаляем каждое ребро, принадлежащее менее чем b треугольникам или менее чем c тетраэдрам.
4. В графе G (который на каждом шаге может уменьшаться) перебором среди вершин, не принадлежащих графам из списка Δ, найдём вершину, принадлежащую наименьшему числу (неупорядоченных) пар различных треугольников с общим ребром. Если таких вершин несколько, то выберем из них ту, которая принадлежит наибольшему числу треугольников. Удаляем эту вершину (если таковых несколько, то удаляем любую из них).

Мотивировка алгоритма такова: если вершина принадлежит большой клике, то она принадлежит и большому числу пар треугольников с общим ребром. С другой стороны, при фиксированном числе пар треугольников с общим ребром, чем меньше всех треугольников, тем больше доля этих пар среди всех треугольников.

Этот алгоритм показал свою эффективность в ряде задач биоинформатики. В частности, — в задаче поиска консенсуса для регуляторных сигналов. В этой задаче по n данным последовательностям в алфавите $\{a, c, g, t\}$ образуем n-дольный граф Γ, вершинами которого служат слова из этих последовательностей фиксированной длины. Две вершины соединяются ребром в графе Γ, если они являются словами из разных последовательностей и похожи друг на друга больше некоторого фиксированного порога. Например, они отличаются друг от друга в неболее чем фиксированном числе позиций. Системе попарно похожих слов (по одному слову в каждой из $q \le n$ последовательностей) соответствует клика с q вершинами в графе Γ.

Ниже приводится один из многих примеров решения такой задачи. Сначала перечисляются 19 последовательностей — регуляторных областей генов кишечной палочки; каждая длиной примерно 240 знаков. Ответы длиной 16 показаны во всех последовательностях (в данном случае $q = 19$) заглавными буквами, по одному слову в каждой последовательности, кроме первой и четырнадцатой, в которых получено по два слова, что соответствует двум кликам такого размера.

1, purR: gctccgtgtcgtttttccggcgtaccgcaacacttttgttgtgcgtaaggtgtgtaaAGGCAAACGTTTACCT
tgcgattttgcaggagctgaagttagggtctggagtgaaatggaatggcaacaataaaagatgtag
cgaaacGAGCAAACGTTTCCACtacaactgtgtcacacgtgatcaacaaaacacgtttcgt
cgctgaagaaacgcgcaacgccgtgtgggcagcgattaaa

2, purEK: AGGAAAACGGTTGCGTggctgtgaaatcagcaaagttgcgggtttttttaaaacggaaaatgaatc
agctcaacgtcatccgccgtgactttcaccattgaaccttccgtatgccaggcacccagtaccacgcgaaaagc
aggttgctgattggcgataagttcatgcaccgccgggcgatgggtatgcccgtggatcagc

3, cvpApurF: AAGAAAACGTTTGCGTagggatttccttcccgcgcatcaataaaaatggcgctgaaaaaatattc
aacgccatcgacttttttatgcctttgcggcatcgggcaatgcgtgtcggatgcggcgtaaacgccttatccgacctacgg
ttctacccctgcgtaggcctgataagacgcgccagcgtcgcatcaggcaagaccg

4, purC: ACGCACACGTTTGCGTatcatatcagaaaaaagggccggatgattccagccctgtatttttacttgct
aaacgcagcctggaagacagctaccagcgcgtcgttctgactctgagtcagagtatgacctttcggatcga
tgaactgtaggctgctgcggttatctaaatcgccaacctgcagtttatagtcaccggatgc

5, purMN: tccgaaaactaacctttaccctggcacaagtcttctttcgccgcgcgcctggggaaaagacgtgcaaa
aaggttgtgtaaagcagtcTCGCAAACGTTTGCTTtccctgttagaattgcgccgaattttatttttctac
cgcaagtaacgcgtggggacccaagcagtgaccgataaaacctctcttagctacaaagatgccggt

6, purL: ACGAAACCGTTTGCGTggaaataaaatcaccatcgtgaattagcaacgcgtgccgccaatggctgtaa
taagttgccatctggcgcaggtttacgcaaaatgccgctcatttatgagtaaacctttcactattattacg
ttttttcaagctgggacgcgcacgacacagagaattaactaattgaaaaaattaaagatta

7, purB: AGGTAACCGATTGCGTcggcagcaaaagcaggaacttccttgatcgctgcgtgcgcggccagtttttg
cagccaacgtacttcaacttgtacacggaatttcagcaaaccatattcgctgaaaatcccgcgcagcgcgc
tgactttatcgccgtagcgtccatcgacaggggaaacggcggtcagtgaggataattccat

8, guaBA: GCGTAACCGATTGCATctacccctttttgcaaaaaatgcttgctatccccgaagggcgggttactatc
gactgaataacctgctgatttagaatttgatctcgctcacatgttaccttctcaatcccctgcaattttta
ccgttagtcgcctgaatcaaacggttcgtctgctgcttgagcatgagatgggacaggtttg
9, purHD: ACGAAAACGTTTGCGCaacgctcgcgaatttttctctttcaatggtgatcacaattttgactgtggtt
accgtgggcaaaatacagaaattacattgatgattgtggataactctgtgcgtaaaaaggtataaagcggg
cttttgctggggaatgcagcagtcagtcatttttctgcaatttttctattgcggcctgcgg
10, glyA: ACGCAAACGATTACCTtcaggctacgcaaggctttggagaataaagagcttgcaaccggaaacggatt
tcttttcaggtttgtgatgcaaatttttcacttcatcacattctttctgaaaaacaccaaagaaccattta
cattgcagggctatttttttataagatgcatttgagatacatcaattaagatgcaaaaaaag
11, pyrD: tcgcaccgtgtcatgtcgcctggcgggcgataaatgattacggcgggttgagtgcaaagaaggagcaa
aatctgccctgaaacaggttCGGAAAACGTTTGCGTttttttgccgcaggtcaattccctttggtccga
actcgcacataatacgccccggtttgcacaccgggaatccaggagagttcatgtactacc
12, prsA: GCGAAAACGTTTTCTTgcgataacctcgcaatcattgctgaattcacattttagcaacgtttctattg
acctttttggcgtattgcgcgggagttcaggatctttaaaaatcaccggagtcggaatacttacaccaggg
tgcgccaccagataccacttctctggcggatccaccggcgttagtatttcaccaacgcctt
13, glnB: TTGAAATCGTTTGCATccagctcgtgtcgggaaagcagtttataaaattctgtccggttgcgccccgc
cattctcgccgcgtgggtgacgttgcctttggtgatttgcagcagcttacgcaaatagttgagttcaaact
gattacgtgcctcaacaaaggttggcagcgccgtattttcaccctccagcgcctgctccac
14, purA: cggtggacttgtggttgcgggcgttgtggtctactacatgttgAGGAAAACGATTGGCTgaacaaaaa
acagactgatcgaggtcatttttgagtgcaaaaagtgctgtaactctgaaaaagcgatggtagaatccatt
tttAAGCAAACGGTGATTTgaaaaatgggtaacaacgtcgtcgtactgggcacccaatgg
15, codBA: ttttacaccgataatttttcccccaccttttgcactcattcatataaaaaatatatttccccACGAAAACGA
TTGCTTtttatcttcagatgaatagaatgcggcggatttttgggtttcaaacagcaaaaaggggg
aatttcgtgtcgcaagataacaactttagccaggggccagtcccgcagtcggcgcggaaag
16, pyrC: GCGGAAACGTTTTCCTttgcacgaaaaataaagggcgcgaatgcgccctcgtgattaatcagtaaatg
gaatgacaatttcgcctggcttcacttcaatgcctttcgccagttttttcgccattgcttcgccctggctg
ccatcttcgcgcaggacgtaagcaggttgctggttaaagtaattgcgtaatgcctggttca
17, purT: gtatttaacttcaaccgccatttgcagcctctcataataactgtgattttatacagtatatttctttt
cggttgagaaatcaacatcagcaataaagacacACGCAAACGTTTTCGTttatactgcgcgcggaattaat
caggggatattcgttatgacgttattaggcactgcgctgcgtccggcagcaactcgcgtga
18, gcvTHP: ACGCAATCGTTCTCTTttgcctgaacttaccaccgaaacagactgttaaccataaggtaaaattgat
catcacattagcttatggttaaaaaatgcaaaaatcgcgacagaataaaaaaccaaaaaatacaccagtttc
tatacaaagatgatgtgatgagaaagtcaatttgaataagacaatattaagagctaaaaaa
19, speAB: GCGCAACCGGTTTCTTtttcataacattattaagcacataaccgaacgtaagtgtgaaagttcggcg
aaaccacgagaaaactcttgtttttacaagagcgcccttgttcagtcctcagtaactgtaaccagctcttga
atcctgagaagcgccgagatgggtataacatcggcaggtatgcaaagcagagatgcagagt

Авторы благодарны К.Ю. Горбунову за помощь, обсуждение и многочисленные советы; в частности, он указал на статью [1].

СПИСОК ЛИТЕРАТУРЫ

1. Storjohann A., Mulders T. Fast Algorithms for Linear Algebra Modulo N. In: *Lecture Notes in Computer Science*, 1461, Springer, 1998.
2. Боревич З. И., Шафаревич И. Р. *Теория чисел*, М.: Наука, 1985.
3. Hoffmann C. M. Group-Theoretic Algorithms and Graph Isomorphism. *Lecture Notes in Computer Science*, 136, Springer, 1982.