Serdica
Bulgariacae mathematicae publicationes

Сердика
Българско математическо списание

ВЫЧИСЛЕНИЕ НА ЭВМ ОДНОГО КОММУТАТОРНОГО ТОЖДЕСТВА В ЗНАКОПЕРЕМЕННЫХ ГРУППАХ

ДАНИЕЛА Б. НИКОЛОВА

Тождества вида $[x, {}_r y] = [x, {}_s y]$, $r < s$, дают некоторые структурные характеристики групп, как например, коммутативность, нильпотентность, энгелевость. Эти тождества выполняются в каждой конечной группе. В работах автора показано каким образом их выполнение отражается на строении разрешимых групп с конечным числом образующих (см. [2]) и каким будет минимальное тождество этого вида в многообразиях $\mathfrak{A}_k \mathfrak{N}_l$ (см. [3]). Целью этой работы является составление алгоритмов минимального тождества вида $[x, {}_r y] = [x, {}_s y]$ в каждой группе из бесконечной серии простых знакопеременных групп и их реализация на ЭВМ. В статье описаны необходимые алгоритмы и приведены две программы на языке FORTRAN — первая вычисляет тождество в A_5, а другая предназначена для общего случая A_n. Они отличаются между собой стратегией запоминания элементов A_n в машине. Эти программы служат в качестве примеров теоретико-групповых программ, способных произвести математические результаты, которые трудно получить иным путем. Даются результаты вычислений для нескольких первых значений числа n.

1. Введение. Будем обозначать далее:

S_n — симметрическую группу степени n (группу всех подстановок на множестве $\{1, 2, \ldots, n\}$),

A_n — знакопеременную группу степени n (группу четных подстановок на множестве $\{1, 2, \ldots, n\}$),

$$x^y = yxy^{-1}, \quad [x, y] = xyx^{-1}y^{-1},$$

$$[x, {}_r y] = [x, \underbrace{y, y, \ldots, y}_{r}] = [[x, \underbrace{y, y, \ldots, y}_{r-1}], y].$$

В статье автора [2] рассмотрены группы, в которых выполняется тождество вида:

$$[x, {}_r y] = [x, {}_s y], \quad r < s. \tag{1}$$

Тождества вида (1), которые выполняются в данной группе G, упорядочены следующим образом: $[x, {}_{m_1} y] = [x, {}_{n_1} y] < [x, {}_{m_2} y] = [x, {}_{n_2} y]$, когда $(m_1, n_1) < (m_2, n_2)$ в лексикографическом смысле. Представляет интерес нахождение минимального тождества $[x, {}_{m_0} y] = [x, {}_{n_0} y]$ вида (1) в данной группе. Покажем вычисление этого тождества в простых знакопеременных группах A_n, где $n \geq 5$.

СЕРДИКА Българско математическо списание. Том 10, 1984, с. 28—40.

2. Тождество в A_5. В первом параграфе рассмотрена программа для исследования частной задачи — нахождение минимального тождества вида (1) в группе A_5.

Напомним практический способ получения чисел m_0, n_0 минимального тождества вида (1) в конечной группе G, который следует из теоремы 1 работы автора [2].

Алгоритм 1. Дана конечная группа G. Мы ищем числа $m_0, n_0 \in \mathbb{N}$, такие, что $[x, {}_{m_0}y] = [x, {}_{n_0}y]$ являлось тождеством в G и такие, что (m_0, n_0) — минимальная в лексикографическом смысле пара.

(a) Рассмотреть все $l = (|G|-1)(|G|-2)$ упорядоченных пар элементов из G, которые не равняются единице; соответствующие им минимальные равенства типа (1) и разницы:

$$g_1, h_1 \in G; \quad [g_1, {}_{m'}h_1] = [g_1, {}_{n'}h_1]; \quad d' = n' - m'$$

$$g_2, h_2 \in G; \quad [g_2, {}_{m''}h_2] = [g_2, {}_{n''}h_2]; \quad d'' = n'' - m''$$

$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$

$$g_l, h_l \in G; \quad [g_l, {}_{m^{(l)}}h_l] = [g_l, {}_{n^{(l)}}h_l]; \quad d^{(l)} = n^{(l)} - m^{(l)}.$$

(b) Найти $m_0 = \max(m', m'', \ldots, m^{(l)})$.

(c) Найти наименьшее общее кратное d чисел $d', d'', \ldots, d^{(l)}$.

(d) Положить $n_0 = m_0 + d$.

Ясно, что (m_0, n_0) — пара чисел, которую мы ищем, и она определяется единственным образом.

Шаг (c) будем выполнять по следующему алгоритму:

Алгоритм 2 (нахождения наименьшего общего кратного l чисел).

(a) Положить $r_1 = d'$, $r_2 = d''$.

(b) Найти наибольший общий делитель PGCD (r_1, r_2) чисел r_1, r_2 по алгоритму Эвклида.

(c) Найти наименьшее общее кратное PPCM (r_1, r_2) чисел r_1, r_2 по формуле: PPCM $(r_1, r_2) = r_1 r_2 / \mathrm{PGCD}(r_1, r_2)$.

(d) Положить $r_1 = \mathrm{PPCM}(r_1, r_2)$, r_2 — следующей разнице $d^{(i)}$. Перейти к шагу (b).

Следует сделать несколько замечаний по поводу реальной программы на языке FORTRAN:

(1) Необходимо обсудить метод запоминания группы подстановок в машине. Подстановку $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ \pi_1 & \pi_2 & \pi_3 & \pi_4 & \pi_5 \end{pmatrix}$ будем записывать в машине в виде упорядоченной 5-орки $(\pi_1, \pi_2, \pi_3, \pi_4, \pi_5)$. Элементы группы S_5 получим в лексикографическом порядке, который можно проследить на следующей блок-схеме (фиг. 1):

Блок-схема получения элементов группы S_5.

На каждом шагу отмечены только элементы очередной подстановки, которые действительно перемещаются. Блок-схема реализируется подпрограммой DATA. Так как группа S_5 невелика, то будем записывать элементы этой группы в двумерном массиве переменных $S5$ (5,120).

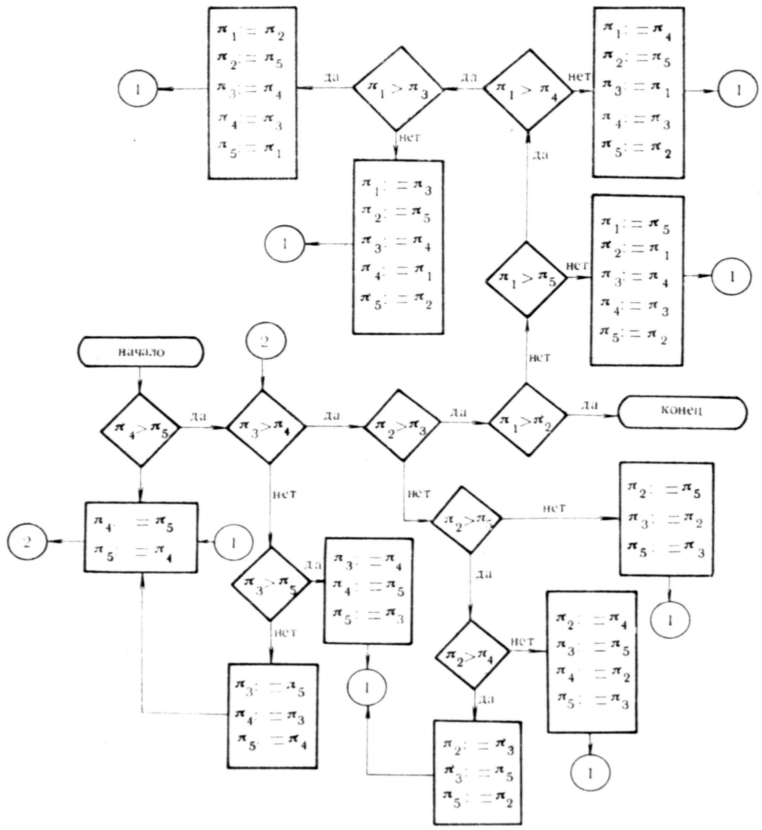
(2) Для данной подстановки $\pi \in S_5$ проверка ее принадлежности к группе A_5 должна быть достаточно легкой. Она реализируется подпрограммой ALT, по следующему алгоритму:

Алгоритм 3 (отделения четных от нечетных подстановок).

(a) Положить $A=0$, $i=1$ и рассмотреть i-й элемент $\pi^{(i)}$ группы S_5. В переменной A будем записывать число инверсий данной подстановки.

(b) Вычислить число инверсий $\pi^{(i)}$ и записать это число в A.

(c) Если 2 делить A, то записать $\pi^{(i)}$ в массиве Y. В противном случае перейти к шагу (d).



Фиг. 1

(d) Положить $i:=i+1$ и перейти к шагу (b).

Шаг (c) реализируется внутренней подпрограммой MOD, которая вычисляет остаток при делении двух целых величин. Эта функция встречается еще один раз в основной подпрограмме DANI 2 при реализации шага (b) алгоритма 2. Элементы знакопеременной группы A_5 будем хранить в двумерном массиве $Y(5,60)$.

(3) Чтобы уменьшить число вычислений шага (a) алгоритма 1, достаточно искать минимальные равенства типа (1) для упорядоченных пар элементов (x, y), где y пробегает всю группу A_5, а x — только представители классов сопряженности элементов из A_5. Действительно

$$[x, {}_n y] = a^{-1}[x^a, {}_n y^a]\, a, \quad \forall a \in A_5.$$

Тогда $[x, {}_m y] = [x, {}_n y] \Leftrightarrow a^{-1}[x^a, {}_m y^a]\, a = a^{-1}[x^a, {}_n y^a]\, a \Leftrightarrow [x^a, {}_m y^a] = [x^a, {}_n y^a]$, $\forall a \in A_5$.

Возможные типы циклического строения элементов из A_5 следующие: элементы типа (123) — они составляют один класс сопряженных элементов в A_5; элементы типа (12) (34) — они тоже все сопряжены в A_5 и, наконец, элементы типа (1 2 3 4 5) — они разбиваются на два класса сопряженных элементов. Выберем представители $x_1 = (123)$, $x_2 = (12)(34)$, $x_3 = (12345)$, $x_4 = (12354)$ и запишем их в двумерном массиве $X(5, 4)$.

(4) Нужно составить алгоритмы для работы с элементами группы; точнее для их сравнения, перемножения, обращения. Будем сравнивать два элемента группы A_5, записывая их при необходимости в виде 5-значных чисел.

Алгоритм 4 (перемножения подстановок степени 5). Заданы две подстановки: $(i_1, i_2, i_3, i_4, i_5)$ и $(j_1, j_2, j_3, j_4, j_5)$. Результат их перемножения — подстановка $(k_1, k_2, k_3, k_4, k_5)$, которая получается следующим образом:

$$\textbf{if } i_t = s, \textbf{ then } k_t = j_s, \; s = 1 \div 5, \; t = 1 \div 5.$$

Алгоритм 5 получения обратного элемента данной подстановки степени 5.

Дана подстановка $\pi = (\pi_1, \pi_2, \pi_3, \pi_4, \pi_5) \in A_5$. Положить $p = (p_1, p_2, p_3, p_4, p_5) = \pi^{-1}$. p получается следующим образом:

$$\textbf{if } \pi_l = k, \textbf{ then } p_k = l, \; k = 1 \div 5, \; l = 1 \div 5.$$

Все эти алгоритмы реализируются основной подпрограммой DANI 2 путем отыскания тождества. Обратные элементы записывается в одномерных массивах $X1(5)$, $Y1(5)$ на каждом шагу, когда появляется необходимость их употребления.

(5) Время вычисления этой задачи на машине ЕС 1040—4,07 мин. Отыскание минимального тождества вида (1) в A_5 можно было сделать обычным способом — путем ручных вычислений. Они оказались бы довольно трудоемкими, так как в группе нормальных делителей не существуют и заранее неизвестен порядок произведения двух элементов, принадлежих разным подгруппам группы A_5. Нам недостаточно знать тождества во всех подгруппах группы A_5, так как непонятно при $x, y \in A_5$ не порождают ли они всю группу. При этом программа этих вычислений послужит нам в качестве их обобщения при больших значениях n для групп A_n.

(6) Наконец, после соответствующих изменений (не пользуясь алгоритмом 3 и записывая в массиве Y элементы из S_5, а в массиве X — представители разных циклических строений подстановок из S_5) ту же программу можно использовать для нахождения минимального тождества вида (1) в группе S_5.

Для удобства читателя приведена машинная программа на FORTRAN, вычисляющая минимальное тождество вида (1) в A_5, а также основной результат выполнения программы — значения m_0, n_0, записанные на переменных MAX M и MAX N:

Программа DANI 2 для вычисления минимального тождества вида $[x, {}_m y] = [x, {}_n y\ v]$ в знакопеременной группе A_5:

```
6                                                                    73
   INTEGER Y, S5, D, X(5, 4)/2, 3, 1, 4, 5, 2, 1, 4, 3, 5, 2, 3, 4, 5, 1, 2, 3, 5, 1, 4/
   DIMENSIN Y(5, 60), S5(5, 120), D(240), M(240), N(240)
   CALL DATA (S5)
   D∅ 1 J=1, 120
 1 WRITE (3, 20) (S5(I, J), I=1, 5)
   CALL ALT (Y, S5)
   D∅ 2 J=1, 60
 2 WRITE (3, 20) (Y(I, J), I=1, 5)
   CALL DANI2 (X, Y, M, N, D, MAXM, MAXN)
   WRITE (3, 21)
   WRITE (3, 22) (M(I), N(I), D(I), I=1, 240)
   WRITE (3,23) MAXM, MAXN
20 FORMAT (10X, 5(12, 1X))
21 FORMAT (10X, 'M', 4X, 'N', 4X, 'D', /)
22 FORMAT (10X, 3(I3, 2X))
23 FORMAT (/, 10X, 'MAXM=', I3 2X, 'MAXN=', I3)
   ST∅P
   END

   SUBR∅UTINE DATA (S5)
   INTEGER S5(5, 120)
   K=1
   D∅ 40 I=1,5
40 S5 (I, K)=I
   IF (S5(4, K)—S5(5,K)) 42, 42, 43
42 K=K+1
   S5(5, K)=S5(4, K—1)
   S5(4, K)=S5(5, K—1)
   D∅ 60 I=1,3
60 S5(I, K)=S5(I, K—1)
43 IF (S5(3, K)—S5(4, K)) 44, 44, 45
44 IF (S5(3, K)—S5(5, K)) 46, 46, 47
46 K=K+1
   S5(3, K)=S5(5, K—1)
   S5(4, K)=S5(3, K—1)
   S5(5, K)=S5(4, K—1)
62 D∅ 61 I=1, 2
61 S5(I, K)=S5(I, K—1)
   G∅ T∅ 42
45 IF (S5(2, K)—S5(3, K)) 48, 48, 49
48 IF (S5(2, K)—S5(5, K)) 70, 70, 71
70 K=K+1
   S5(1, K)=S5(1, K—1)
   S5(2, K)=S5(5, K—1)
   S5(3, K)=S5(2, K—1)
   S5(4, K)=S5(4, K—1)
   S5(5, K)=S5(3, K—1)
   G∅ T∅ 62
47 K=K+1
   S5(3, K)=S5(4, K—1)
   S5(4, K)=S5(5, K—1)
   S5(5, K)=S5(3, K—1)
   G∅ T∅ 62
71 IF(S5(2, K)—S5(4, K)) 72, 72, 73
72 K=K+1
   S5(1, K)=S5(1, K—1)
   S5(2, K)=S5(4, K—1)
```

```
6                                                                    73
      S5(3, K)=S5(5, K—1)
      S5(4, K)=S5(2, K—1)
      S5(5, K)=S5(3, K—1)
      GØ TØ 42
73    K=K+1
      S5(1, K)=S5(1, K—1)
      S5(2, K)=S5(3, K—1)
      S5(3, K)=S5(5, K—1)
      S5(4, K)=S5(4, K—1)
      S5(5, K)=S5(2, K—1)
      GØ TØ 42
49    IF (S5(1, K)—S5(2, K)) 74, 74, 75
74    IF (S5(1, K)—S5(5, K)) 76, 76, 77
76    K=K+1
      S5(1, K)=S5(5, K—1)
      S5(2, K)=S5(1, K—1)
      S5(3, K)=S5(4, K—1)
      S5(4, K)=S5(3, K—1)
      S5(5, K)=S5(2, K—1)
      GØ TØ 42
77    IF (S5(1, K)—S5(4, K)) 78, 78, 79
78    K=K+1
      S5(1, K)=S5(4, K—1)
      S5(2, K)=S5(5, K—1)
      S5(3, K)=S5(1, K—1)
      S5(4, K)=S5(3, K—1)
      S5(5, K)=S5(2, K—1)
      GØ TØ 42
79    IF (S5(1, K)—S5(3, K)) 80, 80, 81
80    K=K+1
      S5(1, K)=S5(3, K—1)
      S5(2, K)=S5(5, K—1)
      S5(3, K)=S5(4, K—1)
 6                                                                   73
      S5(4, K)=S5(1, K—1)
      S5(5, K)=S5(2, K—1)
      GØ TØ 42
81    K=K+1
      S5(1, K)=S5(2, K—1)
      S5(2, K)=S5(5, K—1)
      S5(3, K)=S5(4, K—1)
      S5(4, K)=S5(3, K—1)
      S5(5, K)=S5(1, K—1)
      GØ TØ 42
75    RETURN
      END

      SUBRØUTINE ALT (Y, S5)
      INTEGER Y(5, 60), S5(5, 120)
      K=1
      DØ 90 I=1, 120
      IA=0
      DØ 91 J=1,4
      IB=S5(J, I)
      N=J+1
      DØ 92 L=N, 5
      IF (IB—S5(L, I)) 92, 92, 93
93    IA=IA+1
92    CØNTINUE
91    CØNTINUE
      IF (MØD(IA, 2)) 90, 95, 90
```

```
 6                                                                    73
95 DO  96 J=1, 5
   IC=S5(J, 1)
96 Y(J, K)=IC
   K=K+1
90 CONTINUE
   RETURN
   END

   SUBROUTINE DANI2 (X, Y, M, N, D, MAXM, MAXN)
   INTEGER Y, Y1, X, X1, COM, COM1, COMM, T, R, SD, P, PGCD, PPCM
   DIMENSION Y(5, 60), Y1(5), X(5, 4), X1(5), M(240), N(240), D(240), COM(5, 10
  *0), COM1(5), COMM(100), IR(50)
   P=1
   DO  5 J=1, 4
   DO  6 K=1, 60
   NO=1
   DO  8 I=1, 5
   L=X(I, J)
   S=Y(L, K)
 8 COM(I, NO)=S
   DO  9 I=1, 5
   T=X(I, J)
 9 X1(T)=I
   DO  10 I=1, 5
   L=COM(I, NO)
   S=X1(L)
10 COM(I, NO)=S
   DO  11 I=1, 5
   T=Y(I, K)
11 Y1(T)=I
   DO  12 I=1, 5
   L=COM(I, NO)
   S=Y1(L)
12 COM(I, NO)=S
   COMM(NO)=COM(1, NO)*10 000+COM(2, NO)*1000+COM(3, NO)*100
   +COM(4, NO)*10+*COM(5, NO)
   DO  13 NO=2, 100
   DO  14 I=1, 5
   L=COM(I, NO-1)
   S=Y(L, K)
14 COM(I, NO)=S
   DO  15 I=1, 5
   T=COM(I, NO-1)
15 COM1(T)=I
   DO  16 I=1, 5
   L=COM(I, NO)
   S=COM1(L)
16 COM(I, NO)=S
   DO  17 I=1,5
   L=COM(I, NO)
   S=Y1(L)
17 COM(I, NO)=S
   COMM(NO)=COM(1, NO)*10 000+COM(2, NO)*1000+COM(3, NO)*100
   *+COM(4, NO)*10+*COM(5, NO)
   DO  30 R=2, NO
   MO=NO-R+1
   IF (COMM(MO)-COMM(NO)) 30, 31, 30
31 D(P)=NO-MO
   M(P)=MO
   N(P)=NO
```

```
 6                                                                  73
    P=P+1
    GØ TØ 6
30  CØNTINUE
13  CØNTINUE
 6  CØNTINUE
 5  CØNTINUE
    MAXM=M(1)
    DØ 33 P=2, 240
    IF (MAXM-M(P)) 32, 33, 33
32  MAXM=M(P)
33  CØNTINUE
    IR(1)=D(1)
    IR(2)=D(2)
    DØ 54 J=3, 240
    IF (IR(1)-IR(2)) 53, 52, 52
53  IC=IR(1)
    IR(1)=IR(2)
    IR(2)=IC
52  DØ 50 I=2, 50
    IR(I+1)=MØD(IR(I-1), IR(I))
    IF (IR(I+1)) 50, 51, 50
50  CØNTINUE
51  PGCD=IR(I)
    PPCM=IR(1)*IR(2)/PGCD
    IR(1)=PPCM
    IR(2)=D(J)
54  CØNTINUE
    MAXN=MAXM+PPCM
    RETURN
    END
```

Рис. 2

Итак, минимальное тождество вида (1) в группе A_5 — это $[x, {}_3y]=[x, {}_{63}y]$.

3. Машинная программа вычисления тождеств в знакопеременных группах A_n, $n\geqq 5$. В этом параграфе вычисления, необходимые для нахождения минимального тождества вида (1) в конечной простой группе A_n при $n=5$ будут обобщены для больших значений n. Программная реализация этой задачи отличается от вышерассмотренной в параграфе 2 прежде всего стратегией запоминания подстановок в машине. Требование хранить все элементы в оперативной памяти является серьезным ограничением для больших n. Чтобы использовать экономно память машины, нам нужен эффективный метод получения каждой подстановки от предыдущей, всегда, когда это нужно в программе. Обычно этой процедурой достигается экономия объема памяти, однако за счет увеличения времени, которое расходуется на порождение элементов. Вот почему нас прежде всего интересует общее количество времени, требующегося для порождения всей группы S_n. В частности, в некоторых алгоритмах можно порождать все множество за время, пропорциональное его мощности. Такие алгоритмы, называемые линейными, предпочтительны, ибо их эффективность — асимптотически наилучшая из возможных. Представляет также интерес количество изменений, которые происходят при переходе к последующим объектам. Иногда желательно, чтобы изменения такого рода были возможно малыми. Такие алгоритмы называются алгоритмами с минимальным изменением. Так что нахождение последовательности $n!$ подстановок на множестве $\{1, 2, \ldots, n\}$, где соседние

подстановки различаются так мало, как только это возможно, — это лучшее, на что можно надеяться с точки зрения минимизации объема работы, необходимого для порождения последовательности. Чтобы такое различие было минимально возможным, любая подстановка в нашей последовательности должна отличаться от предшествующей транспозицией двух соседних элементов. Так как практически трудно порождать весь список последовательностей целиком, то подстановки, принадлежащие S_n, можно порождать итеративно, получая каждую подстановку из предшествующей с добавлением небольшого количества информации. Это делается с помощью трех векторов: текущей подстановки $\pi=(\pi_1, \pi_2, \ldots, \pi_n)$, обратной к ней подстановки $p=(p_1, p_2, \ldots, p_n)$ и записи направления d_i, в котором сдвигается каждый элемент i (-1, если он сдвигается влево, $+1$, если вправо, и 0, если не сдвигается). Элемент сдвигается до тех пор, пока не достигнет элемента, бо́льшего, чем он сам; в этом случае сдвиг прекращается. В этот момент направление сдвига данного элемента изменяется на противоположное и передвигается следующий элемент, меньше его, который можно сдвинуть. Поскольку сохраняется подстановка, обратная π, то в π легко найти место следующего меньшего элемента. Алгоритм 6 (см. [1, с. 193]) представляет собой реализацию этого метода. Как любой алгоритм систематического порождения, он состоит из трех компонентов: выбор начальной конфигурации, трансформация объекта в следующий и условие окончания, указывающее, когда нужно остановиться. Заметим, что в алгоритме присваивается $\pi_0=\pi_{n+1}=n+1$ для остановки передвижения n и $d_1=0$, чтобы в тех случаях, когда m становится равным 1 во внутреннем цикле, остаток внешнего цикла был правильно определен.

Алгоритм 6 (Порождение подстановок в порядке минимального изменения).

for $i=1$ **to** n **do** $\begin{cases}\pi_i \leftarrow p_i \leftarrow i\\ d_i \leftarrow -1\end{cases}$

$d_1 \leftarrow 0$

$\pi_0 \leftarrow \pi_{n+1} \leftarrow m \leftarrow n+1$

while $m \neq 1$ **do** $\begin{cases}\text{вывести } \pi=(\pi_1, \pi_2, \ldots, \pi_n)\\ m \leftarrow n\\ \textbf{while } \pi_{p_m+d_m} > m \textbf{ do } \begin{cases}d_m \leftarrow -d_m\\ m \leftarrow m-1.\end{cases}\\ \pi_{p_m} \leftrightarrow \pi_{p_m+d_m}\\ [[\text{в этот момент } \pi_{p_m+d_m}=m]]\\ p\pi_{p_m} \leftrightarrow p_m\end{cases}$

Рис. 3

Следует сделать несколько замечаний по поводу изложенного выше алгоритма и его программной реализации на FORTRAN-е.

1. По правилам записи операторов **DO** нельзя изменять значение параметра цикла от 0, так что текущая подстановка здесь будет $(n+2)$-мерный

вектор: $\pi=(\pi_1, \pi_2, \ldots, \pi_n, \pi_{n+1}, \pi_{n+2})$. Это не является необходимым для векторов p и d, которые для нас являются n-мерными.

2. Так как на каждом шагу подстановки отличаются только транспозицией, то четные подстановки следуют через одну, т. е. для получения элементов группы A_n не возникает необходимость специального алгоритма.

3. Корректность алгоритма 6 доказана в [1] простой индукцией по n, где показано, что значения d_i, $1 \leqq i < n$ являются в точности теми же, что и соответствующие значения, которые получаются, когда в процессе порождения всех подстановок на множестве $\{1, 2, \ldots, n-1\}$ алгоритм записывает $\pi - n$ (подстановку π с удалением n).

4. Алгоритм 6 — один из наиболее эффективных алгоритмов для порождения подстановок.

5. Алгоритм 6 линеен, с минимальным изменением.

6. Алгоритм 6 останавливается через конечное число шагов.

7. Машинный поиск минимальных тождеств вида (1) в знакопеременных группах A_n осуществляется основной подпрограммой DANI 3. Путем отыскания минимальных равенств типа (1) для упорядоченных пар элементов (x, y) здесь элементы x, y пробегают всю группу A_n. В параграфе 2 мы указали, что достаточно, чтобы x пробегало только представители классов сопряженности элементов из A_5. Обсудим задачу нахождения этих представителей. Они могут быть получены после индивидуальной ручной работы и потом записаны в двумерном массиве $X(N, S)$, где S — число классов сопряженности. Если мы хотим получать их автоматически, как n-мерные векторы, то алгоритм следующий. Пусть y_i пробегают A_n. Полагаем $x_1=y_1$. Если $y_2 \neq x_1^g$, $\forall g \in A_n$, полагаем $x_2=y_2$; если существует $g_1 \in A_n$, такое, что $y_2=x_1^{g_1}$, переходим к следующему y_3. Допустим, что уже выбраны первые i представителей классов сопряженности: $x_1, x_2, \ldots, x_i$. Мы должны сравнивать следующее y_j с x_1^g, $g \in G$; x_2^g, $g \in G$; $\ldots$; x_i^g, $g \in G$. Если нет совпадения, полагаем $x_{i+1}=y_j$; в противном случае рассматриваем следующее y_{j+1}. Если мы будем хранить элементы $x_1^g, x_2^g, \ldots, x_i^g$, когда g пробегает A_n, в памяти машины, то мы будем занимать довольно много памяти. Если мы будем вычислять их каждый раз, количество времени будет очень большое. Так что в этой записи не имеет смысла воспользоваться представителями классов сопряженности. Если представить любую подстановку π в виде произведения циклов, то необходимо рассмотреть способы введения круглых скобок в данной n-последовательности π.

Остальные замечания, которые мы сделали по поводу реальной программы в параграфе 2 для A_5, остаются без изменения и для A_n, когда $n \geqq 5$. Например, основной алгоритм можно использовать для нахождения минимального тождества вида (1) и в симметрических группах S_n.

Статья завершается приведением основной подпрограммы DANI 3, которая находит минимальное тождество вида (1) в бесконечной серии простых групп A_n. Необходимые вычисления были выполнены на ЭВМ для нескольких первых значений n и размерность массивов в этой подпрограмме согласована с наибольшим значением числа n. В главной подпрограмме осуществляется лишь обращение к подпрограмме DANI 3 для нужных значений n и печать чисел m_0, n_0 минимального тождества $[x, {}_{m_0}y]=[x, {}_{n_0}y]$ группы A_n, записанных на переменных MAX M, MAX N. Например, для знакопеременной группы A_6 степени 6 MAX $M=4$, MAX $N=124$.

Программа DANI 3 для вычисления минимальных тождеств вида $[x, {}_m y]=[x, {}_n y]$ в серии простых знакопеременных групп A_n, $n \geqq 5$.

```
 6                                                                      73
    DØ  1 IND=5, 10
    N=IND
    CALL DANI3 (N, NAXM, MAXN)
 1  WRITE (3, 23) N, MAXM, MAXN
23  FØRMAT( /, 10X, 'N=', I2, 2X, 'MAXM=', I10, 2X, 'MAXN= ', I10)
    STØP
    END

    SUBRØUTINE DANI3 (N, MAXM, MAXN)
    INTEGER Y, Y1, X, X1, CØM, CØM1, CØMM, PGCD, PPCM, R, D, T
    DIMENSIØN X(10), X1(10), Y(10), Y1(10), CØM(10, 100), CØM1(10),
   *JPI(12), JP(10), JD(10), JDX(10), CØMM(100), D(2), IR(50)
    MAXMO=1
    D(1)=1
    DØ  17 I=1, N
    X(I)=I
    X1(I)=I
    Y(I)=I
    Y1(I)=I
17  JDX(I)=-1
    JDX(1)=0
    GØ  TØ  26
11  DØ  7 I=1, N
    X(I)=JPI(I+1)
    X1(I)=JP(I)
 7  JDX(I)=JD(I)
26  IX=X(N)
    J=N-1
    DØ  37 I=1, Y
37  IX=IX+X(N-I)*10**I
    ASSIGN 4 TØ  L
    GØ  TØ  16
 4  DØ  34 J=1, N
    Y(J)=JPI(J+1)
34  Y1(J)=JP(J)
    IY=Y(N)
    J=N-1
    DØ  36 I=1, J
36  IY=IY+Y(N-I)*10**I
    NO=1
    DØ  8 I=1, N
    CØM(I, NO)=Y(X(I))
    CØM(I, NO)=X1(CØM(I, NO))
 8  CØM(I, NO)=Y1(CØM(I, NO))
    CØMM(NO)=CØM(N, NO)
    J=N-1
    DØ  9 I=1, J
 9  CØMM(NO)=CØMM(NO)+CØM(N-I, NO)*10**I
    DØ  13 NO=2, 100
    DØ  15 I=1, N
    T=CØM(I, NO-1)
15  CØM1(T)=I
    DØ  14 I=1, N
    CØM(I, NO)=Y(CØM(I, NO-1))
    CØM(I, NO)=CØM1 (CØM(I, NO))
14  CØM(I, NO)=Y1(CØM(I, NO))
    CØMM(NO)=CØMM (NO)+COM(N-1, NO)* 10**I
```

```
6                                                                    73
      J=N-1
      DO 10 I=1, J
10    COMM(NO)=COMM(N, + COM(N-1, NO)* 10**
      DO 30 R=2, NO
      MO=NO-R+1
      IF(COMM(MO)-COMM(NO)) 30, 31, 30
31    IF(MAXMO-MO) 32, 33, 33
32    MAXMO=MO
33    D(2)=NO-MO
      IR(1)=D(1)
      IR(2)=D(2)
      IF (IR(1)-IR(2)) 53, 52, 52
53    IC=IR(1)
      IR(1)=IR(2)
      IR(2)=IC
52    DO 50 I=2, 50
      IR(I+1)=MOD(IR(I-1), IR(I))
      IF (IR(I+1)) 50, 51, 50
50    CONTINUE
51    PGCD=IR(I)
      PPCM=IR(1)*IR(2)/PGCD
      D(1)=PPCM
      GO TO 19
30    CONTINUE
13    CONTINUE
27    IF (IX-IY) 38, 39, 38
38    ASSIGN 11 TO L
      DO 12 I=1, N
      JPI(I+1)=X(I)
      JP(I)=X1(I)
12    JD(I)=JDX(I)
      JPI(1)=N+1
      JPI(N+2)=N+1
      GO TO 19
16    DO 3 I=1, N
      JPI(I+1)=I
      JP(I)=I
 3    JD(I)=-1
      JD(1)=0
      JPI(1)=N+1
      JPI(N+2)=N+1
19    DO 28 J=1, 2
      M=N
18    IF (JPI(JP(M)+JD(M)+1)-M) 24, 24, 25
25    JD(M)=-JD(M)
      M=M-1
      IF (M-1) 18, 27, 18
24    JC1=JPI(JP(M)+1)
      JPI(JPIM)+1)=JPI(JP(M)+JD(M)+1)
      JPI(JP(M)+ JD(M)+1)=JC1
      JC2=JP(JPI(JP(M)+1))
      JP(JPI(JP(M)+1))=JP(M)
28    JP(M)=JC2
      GO TO L, (4, 11)
39    MAXM=MAXMO
      MAXN=MAXM+PPCM
      RETURN
      END
```

Рис. 4

ЛИТЕРАТУРА

1. Э. Рейнгольд, Ю. Нивергельт, Н. Део. Комбинаторные алгоритмы. Теория и практика. Москва, 1980.
2. Д. Б. Николова. Об одном классе тождеств в группах. *Сердика*, **9**, 1983, 189—197.
3. Д. Б. Николова. Тождества в метабелевых многообразиях групп $\mathfrak{A}_k \mathfrak{A}_l$. *Сердика*, **9**, 1983, 235—243.

Единый центр математики и механики
1090 София П. Я. 373

Поступила 18. 1. 1982