ZASTOSOWANIA MATEMATYKI
APPLICATIONES MATHEMATICAE
XI, 3 (1970)

ALGORITHM 8

KRYSTYNA JERZYKIEWICZ (Wrocław)

A PROGRAM TO SHORTEN A SEQUENCE OF SETS OF INTEGER NUMBERS

1. The Program.

begin

comment There is given a sequence of integer numbers

$z[1], z[2], \ldots, z[x]$.

The sequence contains iz sets Z_k ($k = 1, 2, \ldots, iz$). The set Z_k consists of the following elements:

$z[azd[k]], z[azd[k]+1], \ldots, z[azg[k]]$.

The program produces the sequence C that consists of the following elements:

$c[1], c[2], \ldots, c[ilc]$,

and is the shortest possible of those having the property that every set Z_k is a segment of it (the initial arrangement of the elements of Z_k is not necessarily preserved in the corresponding segment of C). The program reads in the following values:

iz, x,
$azd[1], azg[1]$,
$azd[2], azg[2]$,
$\ldots$
$azd[iz], azg[iz]$,
$z[1], z[2], \ldots, z[x]$.

After the sequence C is formed the program prints out the following:

iz, ilc,
$azd1[1], azg[1]$,
$azd1[2], azg[2]$,
$\ldots$
$azd1[iz], azg[iz]$,
$c[1], c[2], \ldots, c[ilc]$,

where $azd1[k]$ and $azg[k]$ ($k = 1, 2, \ldots, iz$) are the subscripts of the first and the last elements of that segment of the sequence C which corresponds to Z_k;

```
integer k, k1, k2, i, i1, j, j1, g, g1, wz, wz1, wz2, ni, iz, iz1,  x,  maxw,  maxi,
        ms1, mgg, mkrok, ilc, dg, d1, d2, krok, dgg, gg, iw, iw2, iw3, dgg1,
        s1, mi1;
boolean b, b1, b2, b3, b4, b5, mb1, mb2;
ininteger (1, iz);
ininteger (1, x);
begin
  procedure iloczyn (g, g1);
    value g, g1;
    integer g, g1;
    begin
      if g1 < g
        then
        begin
          krok: = d1 : = 1;
          d2 : = 0
        end g1 < g
        else
        begin
          krok : = -1;
          d1 : = 0;
          d2 : = 1
        end g1 > g;
      b : = false;
      dgg : = d[g]-d1;
      gg : = g;
      iw : = 0;
f:    gg : = gg-krok;
      dgg1 : = d[gg]-d2;
      b2 : = false;
      iw2 : = 0;
      j1 : = dgg+krok;
      for k : = dgg1 step krok until dgg do
        begin
          x : = c[k];
          for k1 : = wz step 1 until wz1 do
            if x = z[k1]
              then
              begin
                iw2 : = iw2+1;
                if b
```

```
                    then
                    for k2 := j1 step krok until dg do
                      if x = c[k2]
                        then
                        begin
                          iw2 := iw2 − 1;
                          go to e
                        end x = c[k2], k2, b;
                  go to e
                end x = z[k1], k1;
            b2 := true;
e:        end k;
        iw := iw + iw2;
        b3 := iw ≠ ni;
        b4 := gg ≠ g1;
        if b
          then
          begin
            if ¬ b2 ∧ b4
              then go to e1;
e2:         if b3
              then
              begin
                if b5
                  then iw := 0
                  else
                    if b1
                      then iw := iw3
              end b3;
            if maxw < iw
              then
              begin
                maxw := iw;
                maxi := i;
                mi1 := i1;
                ms1 := s1;
                mkrok := krok;
                if iw = iw3
                  then
                  begin
                    mb2 := b1;
                    mb1 : false;
                    mgg := s1 − krok
```

```
                    end iw = iw3
                    else
                    begin
                       mb2 : = b2;
                       mb1 : = b1;
                       mgg : = gg
                    end iw ≠ iw3
                 end maxw < iw;
              if  ¬ b4 ∨ krok < 0
                 then go to e4;
              if  ¬ b3
                    then go to koniec;
              iw : = iw2;
              b : = false
           end b;
        if iw2 ≠ 0
           then
           begin
              b : = true;
              b1 : = b2;
              s1 : = gg + krok;
              b5 : = s1 ≠ g;
              dg : = dgg;
              iw3 : = iw2;
              if ¬ (b4 ∧ b3)
                 then go to e2
           end iw2 ≠ 0;
e1:     dgg : = dgg1 − krok;
        if b4
           then go to f;
e4: end iloczyn;
   integer array d[1 : 4 × iz], c[1 : x + x], z[1 : x],
                 nr, azd, azg, azd1[1 : iz];
   g : = g1 : = iz + iz;
   d[g] : = x;
   ilc : = iz1 : = 0;
   for i : = 1 step 1 until iz do
     begin
        ininteger (1, j);
        ininteger (1, k);
        azd[i] : = azd1[i] : = j;
        azg[i] : = k;
```

```
          d[i] := k-j
        end i;
      for i := 1 step 1 until iz do
        begin
          maxw := -1;
          for j := 1 step 1 until iz do
            begin
              x := d[j];
              if maxw < x
                then
                begin
                  k := j;
                  maxw := x
                end maxw < x
              end j;
            if maxw < 0
              then go to zu;
            iz1 := iz1+1;
            nr[iz1] := k;
            d[k] := -1
          end i;
zu:
      for i := 1 step 1 until x do
      ininteger (1, z[i]);
      go to new;
ppi:
      maxw := 0;
      for i1 := 1 step 1 until iz1 do
        begin
          i := nr[i1];
          if i > 0
            then
            begin
              wz := azd[i];
              wz1 := azg[i];
              ni := wz1-wz+1;
              if ni < maxw
                then go to koniec;
              iloczyn (g, g1);
              iloczyn (g1, g)
            end i > 0
        end i1;
```

```
koniec:
    if maxw = 0
      then
      begin
        dg : = d[g]-1;
        dgg : = d[g1];
        k2 : = dgg-ilc-1;
        for i : = 1 step 1 until iz1 do
          begin
            k : = -nr[i];
            if k > 0
              then
              begin
                if azd1[k] = 0
                  then
                  begin
                    azd1[k] : = azd[k]-k2;
                    azg[k] : = azg[k]-k2
                  end azd1[k] = 0
              end k > 0
          end i;
        for i : = dgg step 1 until dg do
          begin
            ilc : = ilc+1;
            c[ilc] : = c[i]
          end i;
        g1 : = g;
new :   maxw : = 0;
        for i : = 1 step 1 until iz1 do
          begin
            k : = nr[i];
            if k < 0
              then go to endi;
            k1 : = azd[k];
            k2 : = azg[k];
            if k2-k1 < maxw
              then go to endnew;
            for j : = i+1 step 1 until iz1 do
              begin
                j1 : = nr[j];
                if j1 < 0
                  then go to endj;
                wz : = azd[j1];
```

```
                      wz 1: = azg[j1];
                      if wz1 — wz < maxw
                         then go to endi;
                      iw : = 0;
                      for d1 : = k1 step 1 until k2 do
                         begin
                            x : = z[d1];
                            for d2 : = wz step 1 until wz1 do
                               if x = z[d2]
                                  then
                                  begin
                                     iw : = iw + 1;
                                     go to endd1
                                  end x = z[d2], d2;
endd1:                   end d1;
                      if iw > maxw
                         then
                         begin
                            maxw : = iw;
                            mi1 : = i
                         end iw > maxw;
endj:                  end j;
endi:               end i;
endnew:
          if maxw = 0
             then
             begin
                for i : = 1 step 1 until iz1 do
                   begin
                      k : = nr[i];
                      if k > 0
                         then
                         begin
                            wz : = azd[k];
                            wz1 : = azg[k];
                            azd1[k] : = ilc + 1;
                            azg[k] : = ilc + 1 + wz1 — wz;
                            for j : = wz step 1 until wz1 do
                               begin
                                  ilc : = ilc + 1;
                                  c[ilc] : = z[j]
                               end j
                         end k > 0
```

```
                end i;
            go to druk
        end maxw = 0
        else
        begin
            maxi := nr[mi1];
            wz := azd[maxi];
            wz1 := azg[maxi];
            d1 := k := d[g1];
            g := g+1;
            for i := wz step 1 until wz1 do
                begin
                    c[k] := z[i];
                    k := k+1
                end i;
            d2 := k-1;
            d[g] := k;
            go to endmax
        end maxw ≠ 0
end maxw = 0
else
begin
    d1 := if mkrok < 0 then 0 else 1;
    d2 := abs(mkrok)-d1;
    wz := azd[maxi];
    wz1 := azg[maxi];
    ni := wz1-wz+1;
    k1 := d[mgg]-d2;
    j1 := d[ms1]-d1;
    b := maxw ≠ ni;
    if mb2
        then
        begin
            k2 := d[mgg+mkrok]-d1;
            for wz2 := wz step 1 until wz1 do
                begin
                    x := z[wz2];
                    for k := k1 step mkrok until k2 do
                        if x = c[k]
                            then
                            begin
                                c[k] := c[k2];
                                c[k2] := x;
```

```
                        k2 := k2 - mkrok;
                        go to e
                     end x = c[k], k;
e:                end wz2;
               k1 := k2 + mkrok
            end mb2;
         if mb1
            then
            begin
               k2 := d[ms1 - mkrok] - d2;
               for wz2 := wz step 1 until wz1 do
                  begin
                     x := z[wz2];
                     for k := k2 step mkrok until j1 do
                        if x = c[k]
                           then
                           begin
                              c[k] := c[k2];
                              c[k2] := x;
                              k2 := k2 + mkrok;
                              go to e1
                           end x = c[k], k;
e1:               end wz2;
               j1 := k2 - mkrok;
            end mb1
            else
            if b
               then
               for wz2 := wz step 1 until wz1 do
                  begin
                     x := z[wz2];
                     for k := k1 step mkrok until j1 do
                        if x = c[k]
                           then go to e2;
                     j1 := j1 + mkrok;
                     c[j1] := x;
e2:               end wz2, b, ¬ mb1;
         mgg := mgg + mkrok;
         if mkrok < 0
            then
            begin
               d1 := j1;
               d2 := k1;
```

```
        if b
          then
          begin
            g1 := g1 - 1;
            d[g1] := j1
          end b
          else
          if mb1
            then
            begin
              for k := g1 step 1 until ms1 do
                d[k - 1] := d[k];
              d[ms1] := j1;
              g1 := g1 - 1
            end mb1, ¬ b;
        if mb2
          then
          begin
            for k := g1 step 1 until mgg do
              d[k - 1] := d[k];
            g1 := g1 - 1;
            d[mgg] := k1 + 1
          end mb2
      end mkrok < 0
      else
      begin
        d1 := k1;
        d2 := j1;
        if b
          then
          begin
            g := g + 1;
            d[g] := j1 + 1
          end b
          else
          if mb1
            then
            begin
              for k := g step -1 until ms1 do
                d[k + 1] := d[k];
              d[ms1] := j1 + 1;
              g := g + 1
            end mb1, b;
```

```
            if mb2
              then
              begin
                for k : = g step -1 until mgg do
                  d[k+1] : = d[k];
                g : = g+1;
                d[mgg] : = k1
              end mb2
          end mkrok ≥ 0
      end maxw ≠ 0;
endmax:
    nr[mi1] : = -maxi;
    azd1[maxi] : = 0;
    azd[maxi] : = d1;
    azg[maxi] : = d2;
    go to ppi;
druk:
    outinteger(1, iz);
    outinteger(1, ilc);
    for i : = 1 step 1 until iz do
      begin
        outinteger(1, azd1[i]);
        outinteger(1, azg[i])
      end i;
    for i : = 1 step 1 until ilc do
      outinteger (1, c[i])
  end
end program
```

2. An application of the program. The problem of the economy of storage of a family of sets containing common elements was encountered during the preparation of the ODRA-ALGOL compiler (a hardware representation of Algol 60 for the ODRA 1204 computer). If the sets discussed here are understood as subtables of the Compactified Reducing Transition Tables (CRTT) of [1], then the immediate application of the program made it possible to shorten the Final Table (see [1]) by 335 locations. After the author of [1] had redefined the CRTT the application of the program has resulted in a gain of 862 locations, i.e., 12 per cent of the compiler size.

3. The method employed. Let the identifiers with subscripts not enclosed in square brackets denote sets. All the sets are supplied with single subscripts only. If therefore a subscript consists of several symbols (i.e., letters and/or digits) then it denotes an identifier used in the program.

The identifiers with subscripts enclosed in square brackets denote elements of a set or of a sequence.

Let $Z_1, Z_2, \ldots, Z_{iz}$ be the family of sets we are going to discuss, and let the shortened sequence of sets Z_k be denoted by the letter C. To every set Z_k added to the sequence C there is assigned a pair of numbers

$$azd1[k],\ azg[k],$$

in such a way that

$$Z_k = \{c[azd1[k]],\ c[azd1[k]+1],\ \ldots,\ c[azg[k]]\}.$$

The number of elements in a set Z_k is

$$n[k] = azg[k] - azd1[k] + 1.$$

Let us denote by the letter D a sequence, the elements of which

$$d[g],\ d[g+1],\ \ldots,\ d[g1],$$

such that

$$d[g] < d[g+1] < \ldots < d[g1],$$

are the subscripts of those elements of the sequence C which divide it into the following segments:

$$\begin{aligned}
S_g &= \{c[d[g]],\ c[d[g]+1],\ \ldots,\ c[d[g+1]-1]\},\\
S_{g+1} &= \{c[d[g+1]],\ c[d[g+1]+1],\ \ldots,\ c[d[g+2]-1]\},\\
\\
S_{g1-1} &= \{c[d[g1-1]],\ c[d[g1-1]+1],\ \ldots,\ c[d[g1]-1]\}.
\end{aligned}$$

For every set Z_k added to the sequence C there are such numbers p and $p1$ that

$$g \leqslant p < p1 \leqslant g1, \text{ and } azd1[k] = d[p], \text{ and } azg[k] = d[p1]-1.$$

Hence, every set Z_k consists only of a certain number of whole segments and any transposition of elements within a segment is regarded here as not affecting the set Z_k.

At the outset the program arranges the set Z_k in such an order that there is

$$n[1] \leqslant n[2] \leqslant \ldots \leqslant n[iz].$$

Example. A sequence of sets

1 2 6 7
3 4 5 6 7
3 4 5
2 3 4 6

will be arranged in the following order:

3 4 5 6 7
1 2 6 7
2 3 4 6
3 4 5.

Such an arrangement of sets reduces in most cases the time required to process the data.

Next, the program finds a set Z_k for which there exists another set Z_l $(l \neq k)$ such that $Z_l \cap Z_k$ is the maximum one of all sets of the form $Z_i \cap Z_k$ $(i = 1, 2, \ldots, iz,\ i \neq k)$.

In the example given above there is $k = 1$. The set

3 4 5 6 7

has three elements in common with the set

2 3 4 6.

The sequence C consists now of one segment, viz., the set Z_k just found]

Let us denote by the symbol $md[p, k]$ the number of elements $z[i.$ which fulfil the following relation:

$$z[i] \in S_p \cap Z_k.$$

Let $mc[k]$ be the number of elements common to the set Z_k and the sequence C, i.e.,

$$mc[k] = \max(mc1[k], mc2[k], mc3[k]).$$

Here

$$mc1[k] = \begin{cases} n[k] & \text{(if there exist such } p \text{ and } p1 \text{ that} \\ & \quad Z_k \subset \bigcup_{i=p}^{p1} S_i \text{ and } \bigcup_{i=p+1}^{p1-1} S_i \subset Z_k), \\ 0 & \text{(if otherwise)}, \end{cases}$$

$$mc2[k] = \sum_{i=p+1}^{g1} (d[i]-d[i-1]) + md[p-1, k],$$

where p is such that

$$S_{p-1} \not\subset Z_k \text{ and } S_j \subset Z_k \quad (j = p, p+1, \ldots, g1-1),$$

and

$$mc3[k] = \sum_{i=g}^{p-1} (d[i+1]-d[i]) + md[p, k],$$

where p is such that

Further, the program seeks such a set Z_k for which there is

$$mc[k] = \max_{Z_i \not\subset C}(mc[i]).$$

The program starts to evaluate numbers $mc[k]$ for the sets with greatest numbers of elements. At the same time it determines the maximum value of $mc[k]$. If for a set Z_l investigated as the next one the number of its elements, $n[l]$, is not greater than the altogether found maximum value of $mc[k]$, then, making use of the fact that the numbers of elements in the remaining sets are not greater than $n[l]$, the process of evaluating $mc[k]$ is terminated.

Let us assume that $mc[k] > 0$. The set Z_k has been therefore determined. According to the value of $mc[k]$, there are three different ways of adding the set Z_k to the sequence C.

1° $mc[k] = mc1[k]$.

The elements of the segment S_p and those of the segment S_{p1} are being arranged in such an order that the elements which also belong to Z_k have their subscripts greater in the segment S_p and smaller in the segment S_{p1} than the remaining elements of these segments. Both segments are therefore divided into two parts. The subscript r which divides the segment S_p and the subscript t dividing S_{p1} are now added to the sequence D in such a way that the resulting sequence is again an increasing one. The $azd1[k]$ and $azg[k]$ take on the values r and $t-1$, respectively. The number of segments in the sequence C has therefore increased by two. If $md[p, k] = 0$ and/or $md[p1, k] = 0$ then the respective segments S_p and/or S_{p1} are not divided, and the subscripts r and/or t have the values $d[p+1]$ and/or $d[p1-1]$.

Example. If the sequence C has the form

$$|1\ 3\ 7\,|\,4\ 5\ 8\ 9\,|\,2\ 6|,$$

then after adding the following set Z_k

$$2\ 3\ 4\ 5\ 8\ 9$$

it becomes

$$|1\ 7\,|\,3\,|\,4\ 5\ 8\ 9\,|\,2\,|\,6|.$$

The number of segments has increased by two.

If the set Z_k has the form

$$4\ 5\ 8,$$

then after adding it to C the number of segments increases by one and the resulting sequence C becomes

$$|1\ 3\ 7\,|\,9\,|\,4\ 5\ 8\,|\,2\ 6|.$$

2° $mc[k] = mc2[k]$.

The elements of the segment S_{p-1} are being arranged in such an order that those which also belong to Z_k have their subscripts greater than the remaining ones. The obtained in this way subscript r which divides S_{p-1} into two parts is now added to the sequence D and we have $azd1[k] = r$. Moreover, there is formed another segment, S_{g1}, consisting of those elements of Z_k which do not belong to $\bigcup_{i=p-1}^{g1-1} S_i$. The value of $azg[k]$ is equal to the greatest subscript of S_{g1}, i.e., $d[g1+1]-1$.

Example. If the sequence C is as follows

$$|1\ 3\ 7\,|\,4\ 5\ 8\ 9\,|\,2\ 6|,$$

then after adding to it the set Z_k

$$1\ 2\ 5\ 6$$

the resulting sequence C becomes

$$|1\ 3\ 7\,|\,4\ 8\ 9\,|\,5\,|\,2\ 6\,|\,1|.$$

3° $mc[k] = mc3[k]$.

The elements of the segment S_p are being arranged in such an order that those which also belong to Z_k have their subscripts smaller than the remaining ones. The obtained in this way subscript t which divides S_p into two parts is now added to the sequence D and we have $azg[k] = t-1$. Moreover, there is formed another segment, S_{g-1}, consisting of those elements of Z_k which do not belong to $\bigcup_{i=g}^{p} S_i$. The $azd1[k]$ is equal to the smallest subscript of S_{g-1}, i.e., $d[g-1]$.

Example. The sequence C has the form

$$|1\ 3\ 7\,|\,4\ 5\ 8\ 9\,|\,2\ 6|.$$

After the following set Z_k

$$1\ 2\ 3\ 4\ 7$$

is added to C, the latter becomes

$$|2\,|\,1\ 3\ 7\,|\,4\,|\,5\ 8\ 9\,|\,2\ 6|.$$

If $mc[k] = 0$ and not all the sets have been added to the sequence C, then these sets are added to each other separately (in the same manner as the previons ones) and the resulting sequence is joined to the sequence C.

In the sequel there are given the successive steps of forming the sequence C from the sets mentioned at the beginning of this section.

Step 1. The sequence C consists of one segment, viz., the set

$$3\ 4\ 5\ 6\ 7.$$

Step 2. There is added the set

$$2\ 3\ 4\ 6,$$

for which we have $mc[k] = mc2[k] = 3$. The sequence C now becomes

$$|5\ 7\ |\ 3\ 4\ 6\ |\ 2\ |.$$

Step 3. There is added the set

$$3\ 4\ 5,$$

for which $mc[k] = mc1[k] = 3$. The sequence C is now as follows:

$$|\ 7\ |\ 5\ |\ 3\ 4\ |\ 6\ |\ 2\ |.$$

Step 4. There is added the set

$$1\ 2\ 6\ 7,$$

for which $mc[k] = mc2[k] = 2$. The final form of the sequence C is therefore the following:

$$|\ 7\ |\ 5\ |\ 3\ 4\ |\ 6\ |\ 2\ |\ 1\ 7\ |.$$

Returning to the initial order of the sets Z_k we obtain

$$\begin{aligned} azd1[1] &= 5, & azg[1] &= 8,\\ azd1[2] &= 1, & azg[2] &= 5,\\ azd1[3] &= 2, & azg[3] &= 4,\\ azd1[4] &= 3, & azg[4] &= 6. \end{aligned}$$

Example. The number of sets = 6. The overall number of elements of the sets $Z_k = 34$.

The limits of the sets Z_k are equal to

1, 5,
6, 12,
13, 20,
21, 27,
28, 33,
34, 34.

The sequence of the sets Z_k is

1, 2, 3, 6, 7, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6,7, 8, 1, 2, 3, 6, 7, 8, 9, 1, 2, 5, 6, 7, 9, 2.

The number of elements in the sequence $C = 11$. The limits of the sets Z_k in the sequence C are equal to

3, 7,
5, 11,
3, 10,
2, 8,
1, 6,
4, 4.

The sequence C has the following form:

$$5,\ 9,\ 1,\ 2,\ 6,\ 7,\ 3,\ 8,\ 4,\ 5,\ 9.$$

4. Running time. A comparison of the amounts of time required to process different sets of data in the case where the program is written in the ODRA 1204 machine code is given in table 1.

The program in ODRA-ALGOL has required 40 sec. to process the first set of data.

TABLE 1.

Number of sets	Overall number of elements	Number of elements in the sequence C	Time (sec)
13	173	119	4
32	102	79	4
74	1133	798	137
87	1380	989	190
249	1311	861	961
327	2131	1291	2619

Reference

[1] J. Szczepkowicz, *On table-driven syntax-checking within an ALGOL compiler*, Zastosow. Matem. 11(1969),pp. 3–89.

DEPT. OF NUMERICAL METHODS
UNIVERSITY OF WROCŁAW

Received on 3. 11. 1969

ALGORYTM 8

KRYSTYNA JERZYKIEWICZ (Wrocław)

PROGRAM SKRACANIA CIĄGU ZBIORÓW

STRESZCZENIE

Dany jest ciąg liczb całkowitych

$$z[1], z[2], \ldots, z[x].$$

Ciąg ten zawiera iz zbiorów Z_k ($k = 1, 2, \ldots, iz$). Do zbioru Z_k należą elementy

$$z[azd[k]], z[azd[k]+1], \ldots, z[azg[k]].$$

Program tworzy możliwie najkrótszy ciąg C składający się z elementów

$$c[1], c[2], \ldots, c[ilc]$$

w ten sposób, aby każdy zbiór Z_k był odcinkiem tego ciągu (przy czym kolejność elementów zbioru Z_k w takim ciągu nie jest istotna).

Program czyta następujące wartości:

$$iz, x, azd[1], azg[1], azd[2], azg[2], \ldots, azd[iz], azg[iz], z[1], z[2], \ldots, z[x].$$

Po utworzeniu ciągu C program drukuje następujące wartości:

$$iz, ilc, azd1[1], azg[1], azd1[2], azg[2], \ldots, azd1[iz], azg[iz], c[1], c[2], \ldots, c[ilc],$$

gdzie $azd1[k], azg[k] (k = 1, 2, \ldots, iz)$ są odpowiednio wskaźnikami pierwszego i ostatniego elementu tego odcinka ciągu C, który stanowi zbiór Z_k.

АЛГОРИТМ 8

КРЫСТЫНА ЕЖИКЕВИЧ (Вроцлав)

ПРОГРАММА СЖАТИЯ ПОСЛЕДОВАТЕЛЬНОСТИ МНОЖЕСТВ

РЕЗЮМЕ

Пусть дана последовательность целых чисел

$$z[1], z[2], \ldots, z[x].$$

Она содержит iz множеств $Z_k (k = 1, 2, \ldots, iz)$. Множество Z_k состоит из элементов

$$z[azd[k]], z[azd[k]+1], \ldots, z[azg[k]].$$

Программа создает возможно краткую последовательность C, составленную из элементов

$$c[1], c[2], \ldots, c[ilc]$$

так, чтобы любое множество Z_k было отрезком этой последовательности (причем порядок элементов множества Z_k в этой последовательности не существенен). Программа читает следующие данные

$$iz, x, azd[1], azg[1], azd[2], azg[2], \ldots, azd[iz], azg[iz], z[1], z[2] \ldots, z[x]$$

и печатает следующие значения

$$iz, ilc, azd1[1], azg[1], azd1[2], azg[2], \ldots, azd1[iz], azg[iz], c[1], c[2], \ldots, c[ilc],$$

где $azd1[k]$, $azg[k]$ $(k = 1, 2, \ldots, iz)$ являются соответственно индексами первого и последнего элемента того отрезка последовательности C, который дает множество Z_k.