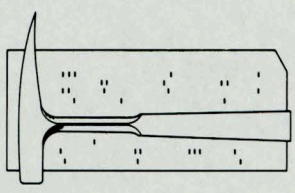
DANIEL F. MERRIAM, Editor

# FORTRAN IV PROGRAM FOR ESTIMATION OF CLADISTIC RELATIONSHIPS USING THE IBM 7040

By

**RONALD L. BARTCHER**

The University of Kansas



**COMPUTER CONTRIBUTION 6**
State Geological Survey
The University of Kansas, Lawrence
1966

# Editor's Remarks

Cladistic relationships, according to the author "...are ... the evolutionary branching sequence among taxonomic units without regard to phenetic similarities among them or to an absolute time scale." Determining these cladistic relationships is a quantitative procedure used in numerical taxonomy. Although this technique has been developed for use in the life sciences, paleontologists undoubtedly will find many applications of such methods in studying fossil groups. Paleontology is entering a new era, but not without its problems and as very aptly stated by R.A. Reyment ("Procedures of quantitative evaluation of variability in foraminifers," Proc. 2nd W. African Micropaleont. Coll., Leiden, 1966):

> "...I think the best way to clarify thought on this subject [quantitative methods in paleontology] is by asking a question. 'Where would Physics and Chemistry be today, if they had not passed from the qualitative and descriptive phase of their development to the quantitative and analytic phase'? I do not think anyone would demand their return to their primary state, nor would anyone claim that it had been 'wrong' to permit these sciences to become 'exact'. Yet, very few paleontologists feel the slightest alarm over the fact that their science has been marking time for a hundred years in the qualitative phase of its development. ...The general run of modern Paleontology is still virtually unaltered after the lapse of sixty to seventy years. It is still entirely subjective and descriptive...."

Here then, hopefully, is a quantitative technique which can be used to solve some of the problems that have perplexed the paleontologist for the last century.

Classification is important in all phases of geology. It is necessary to organize items into meaningful groups in order to develop concepts and ideas. The components of a system are assembled mainly by determining the degree of similarity between them, based on defined properties. This classificational technique is no different.

The Kansas Survey is the only geological organization now known to be distributing computer program decks as well as data decks. The programs are written in ALGOL, FORTRAN II, and FORTRAN IV, and sold for a limited time at a nominal cost. The programs are for Burroughs B5500, IBM 1620, 7040, and 7090/1401 or 7094/1401 computer systems. A list of available decks is given below.

| | ALGOL | FORTRAN II | FORTRAN IV |
|---|---|---|---|
| Marine Simulation (CC 1) | | | $20.00 |
| 2D Regression (CC 2) | $10.00 | $10.00 | $10.00 |
| Trend-6 (CC 3) | | | $25.00 |
| Discrim (CC 4) | | $ 5.00 | |
| Nongridded double Fourier (CC 5) | | | $10.00 |
| Cladistics (CC 6) | | | $25.00 |
| *Trend-3 (SDP 3) | | | |
| Match-Coeff (SDP 4) | | $ 2.00 | |
| *Correlation and distance Coeff (SDP 9) | | | |
| Time-trend (SDP 12) | | $ 5.00 | $ 5.00 |
| Covap (SDP 13) | | $15.00 | |
| Trend-3 (SDP 14) | | $25.00 | $25.00 |
| Cross-Association (SDP 23) | | | $10.00 |
| Single and double Fourier (SDP 24) | | | $ 5.00<br>$15.00 |
| Precambrian wells (SDP 25) List of about 2,600 Precambrian wells | $50.00 | | |
| Trend-4 (SDP 26) | | $ 7.50 | |
| Sediment analysis (SDP 28) | | $10.00 | |
| 4D Trend (KGS B171) | | | $10.00 |
| Conversion of T&R to Cartesian Coordinates (B.170-3) | | $ 5.00 | $ 5.00 |
| Hydrodynamic oil-trap mapping (reprint, Colo. Sch. Mines) | | | $10.00 |

* Out of print, therefore not available.

The Survey will make available for a limited time the card deck (in FORTRAN IV) pertaining to the cladistic relationships at a cost of $25.00.

Comments and suggestions concerning the COMPUTER CONTRIBUTION series are welcome and should be addressed to the editor.

# FORTRAN IV PROGRAM FOR ESTIMATION OF CLADISTIC RELATIONSHIPS USING THE IBM 7040

By

**RONALD L. BARTCHER**

## INTRODUCTION

This series of three programs provides a method for estimating cladistic relationships among taxa as developed by J. H. Camin and R. R. Sokal (1965). By cladistic relationships are meant the evolutionary branching sequence among taxonomic units without regard to phenetic similarities among them or to an absolute time scale. Recent or fossil organisms are presented in a data matrix of the type employed in numerical taxonomy (see Sokal and Sneath, Principals of Numerical Taxonomy, 1963) with the character states arrayed according to a presumed evolutionary sequence. The reconstruction proceeds on the hypothesis that the cladogram which implies the minimum number of evolutionary steps is the best estimate of the correct cladistic relationships. This report is primarily a documentation of the programs and not a description of the theory behind them. Reference should, therefore, be made to the above publication (Camin and Sokal) for the assumptions underlying the method and the justification of the computational steps.

## GENERAL DESCRIPTION

The first program, CLADN1, computes a compatibility matrix from the original data matrix. The compatibility matrix reveals which characters provide "good" patterns that are relatively close to the presumed correct cladogram. It also points out "bad" characters which appear to be miscoded or not to fit the assumptions of the method and should, therefore, be removed from consideration before proceeding with the analysis.

The second program, CLADN2, applies the monothetic clustering procedure to the data matrix to yield a relatively parsimonious cladogram as a basis for further studies. The monothetic procedure is applied to a data matrix from which the "bad" characters have been deleted.

The third program, CLADN3, takes as input the procladogram produced by the monothetic procedure and by performing a number of operations it improves its structure to yield a simpler or more "parsimonious" tree. The following operations are performed: (1) Program removes all empty internodes. (2) Program moves all common steps of branches with a common origin to the base stem of these branches. (3) It moves branches by one or two branching points closer to the base, and tries to rearrange the various branches into clusters that are more parsimonious in terms of the number of evolutionary steps.

## GENERAL DATA INPUT

All three programs require a data matrix of the same form as input. This is a conventional data matrix for numerical taxonomy with the input rowwise, character by character, and a list of correct names of the Operational Taxonomic Unit's (OTU's) and of the characters. (Note: Refer to the comment cards at the start of each program to see the exact format specifications of the input.) The character states are coded as single digit states with the presumed primitive state coded as zero. Bidirectional evolutionary trends are indicated by coding the character states positively in one direction, negatively in the other direction. The present input format of the program (which could be changed) requires that negative character states be restricted to single digits. The positive states can be coded in two digits from 0 to 99 although, in practice, most work to date has restricted itself to a range of from -9 to +9. (Note: Unless the tree printing routine in CLADN2 and CLADN3 is altered, the character states must be in the range -99 to +99.) The data matrix is the only input required by CLADN1. Input for CLADN2 requires a list of "bad" characters which are to be omitted during construction of the procladogram to precede the data matrix.

CLADN3 requires several parameter cards in addition to the data matrix as follows. The first of these cards gives the number of "bad" characters and a list of the names of these "bad" characters. The program will then ignore these characters while practicing parsimony. The second parameter card (or cards) gives the numerical representation of the tree on which the program is to operate. This representation of the tree has in its odd locations the correct OTU names and in its even locations the furcation level at which the two OTU's to each side of it join. It is normally obtained as output from CLADN2, but can also be handpunched if it is desired to use a special tree. Next will follow one or more cards, each containing two numbers, N and K. N tells how many furcation levels to move branches when practicing parsimony and K tell how many passes are to be made through the tree while moving the branches by N levels. Moves of any

number of passes for each magnitude can be made in any sequence. A blank card is used to signal the end of the data for this run.

## PROGRAM OUTPUT

The programs number the NN characters and NT OTU's consecutively from 1 to NN and from 1 to NT, respectively. Since this numbering system may not correspond to the numbering or naming system by which the investigator has been referring to these characters and OTU's, the input affords an opportunity of naming each character and each OTU by a label which can be an alphabetical, numerical, or mixed name of up to six characters including blank spaces. At the beginning of the output a symbol table in which the character numbers assigned by the program will be listed with their corresponding correct names will be printed. The OTU's are listed by their correct names in the output.

The output of CLADN1 consists of the symbol table, just mentioned, and a compatibility matrix as described in Table 3 of Camin and Sokal (1965). The diagonal elements are labeled zero in the computer output, but are not included in the count of the row or column compatibilities. Characters and patterns are labeled at the left and upper margins of the table while the right and lower margins of the table list compatibilities and extra steps for rows and columns, respectively.

The output of CLADN2 consists of the symbol table, followed by a list of the "bad" characters which have been omitted from the computation, followed by the procladogram resulting from the monothetic method. At the left-hand margin of the procladogram are given the level numbers of successive furcations. Evolutionary changes in the states of the characters are indicated on each branch of the cladogram. These are expressed in terms of two numbers; the left number is the character number, using the consecutive numbering system at the left column of the symbol table. The right-hand number expresses the number of evolutionary steps required for this character in the indicated branch. For bidirectional characters, a positive number indicates one direction and a negative number indicates the other direction. The character steps are ordered by character number within any one furcation level. Following the tree there is a summary table which shows the total number of steps for each character and a total count for the entire procladogram.

When "bad" characters have been designated for the input of CLADN2, the output discussed above is repeated once more with "bad" characters added on to the tree and the separate and total counts revised.

The last lines of the output are the numerical representation of the tree. This representation has in its odd locations the correct OTU names and in its even locations the furcation level at which the two OTU's to each side of it join. The representation may then be used as input to the "CLADN3" program.

The output of CLADN3 consists of the symbol table, followed by a list of the "bad" characters which have been omitted from the computation, followed by the procladogram provided as input to the program. At the left-hand margin of the procladogram are given the level numbers of successive furcations. Each branch of the cladogram carries evolutionary steps for all affected characters. These are expressed in terms of two numbers; the left number is the character number, using the consecutive numbering system at the left column of the symbol table. The right-hand number expresses the number of evolutionary steps required for this character in the indicated branch. The character steps are ordered by character number within any one furcation level. Following the tree there is a summary table which shows the total number of steps for each character and a total count for the entire procladogram.

Next will come several sets of output as described above except that before the tree is printed, the equally parsimonious cases (different branching patterns of one branch of the procladogram which have the same number of evolutionary steps) resulting during the moving of branches are printed.

The comments about equally parsimonious solutions refer to the cladogram immediately preceding them. When studying the results of the decisions about equally parsimonious solutions one has to look on the previous cladogram to see what the decisions are based on, but one has to look at the subsequent cladogram to note the results of the decisions that were taken.

The printout for equal parsimony contains the following information. It will first indicate the level number. This refers to levels in the previous cladogram, and it is the level number to which the branches are pulled down, so that the actual furcations that we are referring to occur at one level number higher (if the value of N is one) than the level indicated. Considering level J we are breaking up clusters of level J + N (where N has the same meaning as in the previous paragraph describing the input to CLADN3).

The numbered groups are the OTU's or clusters of OTU's being pulled down to level J. Their composition is identified in the list.

The two or more lines underneath the group identification indicate the combination of groups which come together in a furcation to form a cluster. Usually, two groups will come together, but on special occasions there may be three or four groups which come together most parsimoniously. The other groups which are not listed in any one line can be imagined to join singly at level J, but a second clustering cycle which need not have equal parsimonious solutions could then cluster these other

groups which would be so shown in the following cladogram. Thus, if you have four OTU's A, B, C, D, the program may indicate that A and B should come together leaving C and D unclustered, but a subsequent clustering cycle could put C and D together also, so that the final solution would be A-B and C-D.

Of the equally parsimonious solutions, the one generally chosen is the one in the first line. However, there are special cases when this is not so. The program compares entire trees and portions of trees prior and subsequent to the parsimony procedure. If it finds that after parsimony has been practiced on trees or parts thereof and the total count is not improved, the program will retain the old cladogram and not go on to the new cladogram. Therefore, the occasion may arise where the resulting cladogram is not according to the first line of the equally parsimonious solution but to the second or a subsequent line because this is the way in which the previous cladogram was formed and no improvement resulted by the current parsimony. Whatever final solution is adopted must be reflected in one of the equally parsimonious solutions, although not necessarily the first one.

The stages 1, 2, 3, and so forth refer to successive stages in the clustering cycle. It there are only three OTU's there can be only one stage, because once two of these have been clustered the third has only one way to join the cluster. But if there are four OTU's then we can have two stages. By way of explanation we might go through the way the clustering procedure works. The machine goes through successive pairs, first against second, first against third, first against fourth, and so forth, then continues to second against third, second against fourth, and so forth. After all the paired comparisons have been made the program tries all groups of 3. It will compare the step count for the best group of three to that of the best group of 2. If it is equal, the program will go on to groups of 4. If it is greater, it will select the best cluster it has found. This is repeated with all groups of higher numbers. Thus, in an example of stage 2 clustering, after OTU's A and B have joined in the set A, B, C, D; C could joint A-B first or it could join D. (Note: The step count for a cluster of three, say A, B, C, cannot be less than the step count for a group of two, say A, B. If it were, this would imply that A, B, and C have more steps in common than A and B, which is clearly an impossibility.)

When "bad" characters have been designated in the input of the program, the tree is printed once more at the end of the parsimony procedure with "bad" characters added on and the separate and total counts revised.

The last lines of the output are the numerical representation of the tree. This representation has in its odd locations the correct OTU names and in its even locations the furcation level at which the two OTU's to each side of it join. The representation may then be used as input to the "CLADN3" program, if it is desired to practice further parsimony.

## RESTRICTIONS

This program was written in FORTRAN IV for an IBM 7040 system with 4 scratch tapes. In a 16K machine the programs are limited as follows:

| | | |
|---|---|---|
| CLADN1 | 60 char. | 40 OTU |
| CLADN2 | 50 char. | 30 OTU |
| CLADN3 | 40 char. | 30 OTU |

These programs have been combined into one chain program for a 32K 7040. The programs are available as separate 16K versions or as a combined chain version for 32K. The 32K version can handle 90 char. and 70 OTU's.

Running time for 7 characters and 7 OTU's was about one minute for CLADN1 and CLADN2 and 2 minutes for CLADN3. The running time was increased to 1 minute for CLADN1, 2 minutes for CLADN2, and 11 minutes for CLADN3 for 12 characters and 24 OTU's.

## EXAMPLES

Two sets of data were run on each of the programs. The first of these was data on Recent horses, whose cladistics are best known. The second was data on fusulinids furnished by Roger Kaesler, Department of Geology, The University of Kansas. First we shall follow the analysis of the data on Recent horses through CLADN1, CLADN2, and CLADN3; second we shall follow the analysis for the fusulinids through each of the programs. A printout of each set of sample data follows the printout of each program.

The output of Recent horses from CLADN1 is shown in Figure 1. None of the characters seem to be very bad, although characters 2 and 3 have only two column compatibilities. We shall not designate any characters as "bad."

The output of Recent horses from CLADN2 is shown in Figure 2. While the counts for characters 2, 3, and 6 are higher than the rest, (indicating that they may be "bad" characters), there is not much we can say, since there is little difference between these three characters and the rest.

The output of Recent horses from CLADN3 is shown in Figure 3.

Note that OTU's 7 and 9 have no steps in common at level 4. The joining of OTU's 7 and 9 at level 4 has produced an empty internode (Fig. 3a); therefore, they should join at level 3, not level 4. This is illustrated in the second tree of the output (Fig. 3b). We now have a cluster with more than two groups joining at the same level. This is acceptable, since no subset of two or more of the

new groups has any steps in common. (Note: To resolve this cluster, more characters are clearly needed.)

On the first pass through the tree moving branches down by 1 level, we find two equally parsimonious cases (Fig. 3c). From the tree in Figure 4, we could arrive at either of the trees in Figure 5.

In this case, the chosen tree is identical to the original one. (This is true so long as the original tree is one of the equally parsimonious cases.) The total count is the same as before, so we proceed to the next number of levels it is desired to move.

On the second pass through the tree, moving down two levels at a time, we find no equally parsimonious cases and no other improvements (Fig. 3d). The total count for this tree is the same as the previous total count.

Since we wished to move no other number of levels, the program terminates.

The output of the fusulinids from CLADN1 is shown in Figure 6. None of the characters seem very good; however, character 11 appears worse than the others. We shall call character 11 "bad."

The output of fusulinids from CLADN2 is shown in Figure 7. Character 8, as well as character 11, now seems to be a "bad" character, since its count is considerably higher than the others. (Note: The tree with the "bad" characters added on which normally appears has been deleted from Figure 7 to conserve space.)

The output of fusulinids from CLADN3 is shown in Figure 8.

Note that while there are no empty internodes, there is one cluster (OTU's 1, 2, 22, 24 at level 2) which has more than two groups joining at the same level with several subsets which have steps in common (Fig. 7). In this case, the group are separate OTU's, but they could have been clusters of OTU's.

When CLADN3 attempts to resolve this cluster, it finds that OTU's 22 and 24 (this is stage 1) should join first, but then (at stage 2) it finds that either 1 and 2 should join or that 2 and 22, 24 should join (Fig. 8a). Because the original tree isn't one of the equally parsimonious cases, CLADN 3 decides that 1 and 2 should join, since they are first in the list of equally parsimonious cases. This shows up in the first tree of the output, with an improvement by 5 of the total count (Fig. 8b).

The first pass through the tree is then made, moving down by 1 level. Many equally parsimonious cases may be found (Fig. 8c). We also note that there was a marked improvement in the total count (by 28; Fig. 8d).

A second pass is then made through the tree moving down by 1 level. We note further improvement (by 3), so we make another pass (Fig. 8e, 8f). On the third pass there is no improvement, so we stop and go to the next number of levels to be moved. (Had we asked it to make at most 1 pass moving by 1 level, it would have stopped with a total count of 121.)

When moving by 2 levels, we again find many equally parsimonious cases, but no improvement in the total count, so we stop (Fig. 8g).

Because we designated character 11 as "bad," we will print our final tree once more, with all characters (including character 11, which up to now had been left out) put on the tree (Fig. 8h). (Note: To conserve space we did not show the actual tree.)

Note that the count for character 11 has improved from the tree that CLADN2 produced, but it is still quite large. Also the count for character 8 is large, indicating that perhaps it is an unreliable character.

Acknowledgments.-This project was supported in part by a grant (GB 4927) from the Systematic Program of the National Science Foundation. I am indebted to Dr. Joseph H. Camin, Dr. Robert R. Sokal, and Dr. F. James Rohlf for their valuable suggestions and editorial guidance throughout the writing of this paper. These programs were developed and tested at The University of Kansas Computation Center.

REFERENCES

Camin, J. H., and Sokal, R. R., 1965, A method for deducing branching sequences in phylogeny: Evolution v. 19, p. 311-326.

Sokal, R. R., and Sneath, P.H.A., 1963, Principles of numerical taxonomy: W.H. Freeman and Co., San Francisco and London, 359 p.

Listing of FORTRAN IV statements for cladistic program.

```
$IBSYS
$JOB            0582,BARTCHER                                          ,06000,06000,
$TIME           030
$IBJOB          MAP
$IBFTC CLADN1  NODECK
      INTEGER STATE(60,41),ISW(3,40),COMPAT(60,60),CHECK(3,40),COLSUM(2,
     160),ROWSUM(60,2)  ,LABEL(10)
      INTEGER TOT,A,B,C,BIG,SMALL,STEM,D,DT    ,U,V
      LOGICAL TEST,PVT
      EQUIVALENCE (ISW(1,1),COLSUM(1,1)),(TEST,PVT),(CHECK(1,1),ROWSUM(1
     1,1))  ,(NNN,NN1)
C
C     THE PROGRAM IS LIMITED TO NN=60 CHARACTERS AND NT=40 OTU-S
C     BUT COULD BE EXPANDED FOR SYSTEMS WITH GREATER CAPACITY BY INCREASING
C     THE DIMENSIONS.
C     DIMENSIONS FOLLOW-
C         STATE(NN,NT+1),ISW(3,NT),COMPAT(NN,NN),CHECK(3,NT),COLSUM(2,NN),
C         ROWSUM(NN,2),LABEL(10)
C
C     THE PROGRAM USES THE FOLLOWING LIBRARY FUNCTIONS-
C          IABS(J)    (INTEGER ABSOLUTE VALUE)
C          MAXO(J,K)       (INTEGER MAXIMUM)
C          MINO(J,K)       (INTEGER MINIMUM)
C          EXIT       (TERMINATES THE PROGRAM)
C                CLOCK(I)      (PRINTS TIME SINCE LAST CALL)
C
C     THE FORM OF INPUT IS AS FOLLOWS-
C
C     FIRST CARD       NUMBER OF SEPARATE DATA SETS ON WHICH CLADN1 SHOULD
C                      BE PERFORMED   FORMAT(I2)
C
C                      NN=NUMBER OF CHARACTERS AND NT=NUMBER OF OTU-S
C     SECOND CARD      THE NEXT 60 COLUMNS ARE FOR THE NAME OF THIS SET OF
C                      DATA AND FOR ANY ADDITIONAL COMMENTS      FORMAT(2I2,10A6)
C
C                      EACH DATA CARD CONTAINS THE CHARACTER NAME AND THE
C                      CHARACTER STATES OF ALL OTU-S FOR THAT PARTICULAR
C     DATA             CHARACTER    FORMAT(A6,(37I2)). IF THERE ARE MORE THAN
C     CARDS            37 OTU-S A SECOND DATA CARD  (FORMAT(37I2))  WILL
C                      FOLLOW THE DATA CARD FOR EACH CHARACTER. THE CHARACTER
C                      NAME IS NOT REPEATED IN THE SECOND CARD. THERE MUST
C                      BE NN DATA CARDS (OR SETS OF DATA CARDS).
C
C     OTU NAME         THESE CONTAIN THE OTU NAMES IN PROPER SEQUENCE
C       CARDS          FORMAT(13A6)
C
C
C     CARDS FOR NEXT DATA SET IF  MORE THAN ONE
C
C      U IS THE INPUT UNIT AND V IS THE OUTPUT UNIT
      CALL CLOCK(0)
      U=5
      V=6
C     READ NUMBER OF SETS OF DATA
      READ(U,101)IJKL
      DO 156 IJKLM=1,IJKL
C     READ NUMBER OF CHARACTERS AND NUMBER OF OTU-S AND DATA NAME AND
C     COMMENTS
      READ(U,101)NN,NT,(LABEL(I),I=1,10)
  101 FORMAT(2I2,10A6)
```

```
      NN1=NN+1
      NT1=NT+1
C     READ CHARACTER NAME AND CHARACTER STATES
      DO 102 I=1,NN
  102 READ(U,103)STATE(I,NT1),(STATE(I,J),J=1,NT)
C     READ OTU NAMES
      READ(U,104)(STATE(NN1,J),J=1,NT)
  103 FORMAT(A6,(37I2))
  104 FORMAT(13A6)
C     WRITE A HEADING
      WRITE(V,270)(LABEL(I),I=1,10)
  270 FORMAT(26H1COMPATIBILITY MATRIX FOR ,10A6////30X,12HSYMBOL TABLE /
     1//31X,10HCHARACTERS//25X,6HNUMBER,11X,4HNAME)
  272 FORMAT(27X,I2,12X,A6)
C     WRITE A SYMBOL TABLE AND ASSIGN CHARACTER NUMBERS AND OTU NUMBERS
      DO 273 I=1,NN
      WRITE(V,272)I,STATE(I,NT1)
  273 STATE(I,NT1)=I
      DO 274 I=1,NT
  274 STATE(NN1,I)=I
C   BEGIN COMPUTATION OF COMPAT. MATRIX
   21 DO 9 KK=1,NN
C      ORDER THE COLUMNS OF THE DATA MATRIX INTO THE PATTERN FOR
C      CHARACTER KK, FROM LEAST TO LARGEST STATE CODES.
      J=NT
    7 J=J-1
      DO 5 I=1,J
      IF(STATE(KK,I).LE.STATE(KK,I+1))GO TO 5
      II=I+1
      DO 6 JJ=1,NNN
      IC=STATE(JJ,I)
      STATE(JJ,I)=STATE(JJ,II)
    6 STATE(JJ,II)=IC
    5 CONTINUE
      IF(J.GT.1)GO TO 7
C      DEFINE CHECK (1,I) FOR EACH STATE (SEQUENTIALLY NUMBERED FROM
C      LOWEST TO HIGHEST STATE CODE) I AS THE LAST COLUMN IN THE
C      PATTERN TABLE WITH THAT STATE CODE FOR KK.
      IA=1
      TEST=.FALSE.
      DO 3 J=1,NT
      DO 1 I=IA,NT
    1 IF(STATE(KK,I).NE.STATE(KK,IA)) GO TO 2
      TEST=.TRUE.
    2 CHECK(1,J)=I-1
      IF(TEST) GO TO 10
    3 IA=I
   10 A=J
C      START PROCESSING FOR PATTERN OF CHAR. KK
C      INITIALIZE PARAMETERS
      DO 9 K=1,NN
      DO91 I=1,3
      DO91 J=1,A
   91 ISW(I,J)=0
      PVT=.TRUE.
      COUNT=0
      IZ1=0
      IZ2=0
      IZ3=0
      TOT=0
```

```
C       LOCATE THE PIVOT (ZERO STATE) OF THE PATTERN CHARACTER. IF
C       THE ZERO STATE IS NOT INCLUDED IN THE DATA, AND IF THE
C       CHATACTER HAS POSITIVE STATES, PVT IS SET TO .FALSE.
        DO 90 I=1,A
        J=CHECK(1,I)
        IF(STATE(KK,J).LT.0) GO TO 90
        IF(STATE(KK,J).NE.0)PVT=.FALSE.
        GO TO 33
   90   CONTINUE
        I=A
   33   II=I+1
        IPVT=I
        L=1
C       FOR EACH STATE (I) OF THE PATTERN CHARACTER, KK,STORE THE
C       LARGEST VALUE OF CHARACTER K (WITHIN THAT SECTION OF THE
C       PATTERN TABLE) IN CHECK (2,I), AND STORE THE LEAST SUCH VALUE IN
C       CHECK (3,I).
        DO93 I=1,A
        B=CHECK(1,I)
        CHECK(2,I)=-30
        CHECK(3,I)=30
        DO92 KI=L,B
        IF(STATE(K,KI).GT.CHECK(2,I)) CHECK(2,I)=STATE(K,KI)
   92   IF(STATE(K,KI).LT.CHECK(3,I)) CHECK(3,I)=STATE(K,KI)
   93   L=KI
C       STORE IN LARGE THE LARGEST VALUE TAKENBY CHARACTER K AND IN
C       LITTLE THE SMALLEST VALUE TAKEN BY CHARACTER K.
        LITTLE=CHECK(3,1)
        LARGE=CHECK(2,1)
        DO 4 I=1,A
        IF(CHECK(2,I).GT.LARGE) LARGE=CHECK(2,I)
    4   IF(CHECK(3,I).LT.LITTLE) LITTLE=CHECK(3,I)
C       STORE THE LARGEST VALUE OF CHARACTER K ON THE LEFT (NEGATIVE)
C       SIDE OF THE PIVOT INMAXLFT, AND THE LEAST SUCH VALUE IN MINLFT.
C       SIMILARLY ON THE RIGHT (POSITIVE) SIDE FOR MAXRT AND MINRT.
        MAXLFT=CHECK(2,1)
        MAXRT=CHECK(2,A )
        MINLFT=CHECK(3,1)
        MINRT=CHECK(3,A)
        I1=IPVT-1
        IF(IPVT.EQ.1)GO TO 11
        DO95 I=1,I1
        IF(MAXLFT.LT.CHECK(2,I))MAXLFT=CHECK(2,I)
   95   IF(MINLFT.GT.CHECK(3,I))MINLFT=CHECK(3,I)
   11   IF(IPVT.EQ.A)GO TO 18
        IF(.NOT.(PVT))I1=IPVT-2
        I2=I1+2
        DO96 I=I2,A
        IF(MAXRT.LT.CHECK(2,I))MAXRT=CHECK(2,I)
   96   IF(MINRT.GT.CHECK(3,I))MINRT=CHECK(3,I)
   18   I=IPVT
        IF(LARGE.LT.0) TOT=-LARGE
        IF(LITTLE.GT.0)TOT=LITTLE
        IF(I.EQ.1.OR.I.EQ.A) GO TO 17
C       IF EITHER MINRT OR MINLFT IS .LE. ZERO GO TO 66. OTHERWISE , SET
C       IZ1 EQUAL TO THE SMALLER AND SUBTRACT TOT (WHICH EQUALS
C       LITTLE).
        IF(.NOT.(MINLFT.GT.0.AND.MINRT.GT.0))GO TO 66
        IZ1=MIN0(MINRT,MINLFT)-TOT
        GO TO 17
```

```
C       IF BOTH MAXLFT AND MAXRT ARE NEGATIVE, SET IZ1 EQUAL TO THE
C       LARGER ABSOLUTE VALUE, DECREASED BY TOT (THE ABSOLUTE VALUE OF
C       LARGE).
   66 IF(.NOT.(MAXRT.LT.0.AND.MAXLFT.LT.0))GO TO 17
      IZ1=IABS(MAX0(MAXRT,MAXLFT))-TOT
C       IF THERE IS NO PIVOT IN THE PATTERN GO TO 60.
   17 IF(.NOT.(PVT))GO TO 28
      IF(CHECK(3,I).EQ.0)GO TO 60
      IF(.NOT.(I.NE.A.AND.CHECK(3,I).GT.0.AND.MINRT.GT.0.AND.MIN0(CHECK(
     13,I),MINRT).GT.TOT))GO TO 61
      IZ2=MIN0(CHECK(3,I),MINRT)-TOT
      GO TO 60
   61 IF(.NOT.(I.NE.1.AND.CHECK(3,I).GT.0.AND.MINLFT.GT.0.AND.MIN0(CHECK
     1(3,I),MINLFT).GT.TOT))GO TO 62
      IZ3=MIN0(CHECK(3,I),MINLFT)-TOT
      GO TO 60
   62 IF(.NOT.(I.NE.A.AND.CHECK(2,I).LT.0.AND.MAXRT.LT.0.AND.IABS(MAX0(C
     1HECK(2,I),MAXRT)).GT.TOT))GO TO 63
      IZ2=IABS(MAX0(CHECK(2,I),MAXRT))-TOT
      GO TO 60
   63 IF(.NOT.(I.NE.1.AND.CHECK(2,I).LT.0.AND.MAXLFT.LT.0.AND.IABS(MAX0(
     1CHECK(2,I),MAXLFT)).GT.TOT))GO TO 60
      IZ3=IABS(MAX0(CHECK(2,I),MAXLFT))-TOT
   60 IF(.NOT.(PVT))GO TO 28
C       SET ISW= NO. OF STEPS FOR SMALLEST CLUSTERS WITHIN LEFT,
C       RIGHT, AND PIVOT CLUSTERS
C       HANDLE PIVOT CLUSTER
      IF(CHECK(2,I).GE.0.AND.CHECK(3,I).LE.0)GO TO 26
      ISW(2,I)=MIN0(IABS(CHECK(3,I)),IABS(CHECK(2,I)))-IZ3-IZ2-TOT
      ISW(3,I)=MAX0(IABS(CHECK(2,I)),IABS(CHECK(3,I)))-ISW(2,I)-IZ3-IZ2-
     1TOT
      GO TO 28
   26 ISW(3,I)=CHECK(2,I)-CHECK(3,I)-ISW(2,I)-IZ3-IZ2-TOT
   28 STEM=0
C       HANDLE RIGHT CLUSTER
      IF(.NOT.(PVT))II=II-1
      IF(II-1.EQ.A)GO TO 42
      DO 45 C=II,A
      SMALL=30
      DO 40 I=C,A
   40 IF(CHECK(3,I).LT.SMALL) SMALL=CHECK(3,I)
      IF(SMALL.LE.(TOT+STEM+IZ1+IZ2)) GO TO 43
      ISW(1,C)=SMALL-STEM-TOT-IZ1-IZ2
      STEM=STEM+ISW(1,C)
      GO TO 46
   43 BIG=-30
      DO 44 I=C,A
   44 IF(CHECK(2,I).GT.BIG) BIG=CHECK(2,I)
      IF(BIG.GE.0) GO TO 46
      IF(IABS(BIG).LE.(TOT+STEM+IZ1+IZ2)) GO TO 46
      ISW(1,C)=IABS(BIG)-STEM-TOT-IZ1-IZ2
      STEM=STEM+ISW(1,C)
   46 IF(CHECK(2,C).GE.0.AND.CHECK(3,C).LE.0)GO TO 39
      ISW(2,C)=MIN0(IABS(CHECK(3,C)),IABS(CHECK(2,C)))-IZ1-IZ2-TOT-STEM
      ISW(3,C)=MAX0(IABS(CHECK(2,C)),IABS(CHECK(3,C)))-ISW(2,C)-IZ1-IZ2-
     1TOT-STEM
      GO TO 45
   39 ISW(3,C)=CHECK(2,C)-CHECK(3,C)-ISW(2,C)-IZ1-IZ2-TOT-STEM
   45 CONTINUE
C       HANDLE LEFT CLUSTER
```

```
   42 IF(.NOT.(PVT))II=II+1
      IF(II-1.EQ.1)GO TO 47
      STEM=0
      D=II-2
      DO 49 J=1,D
      IJK=D+1-J
      SMALL=30
      DO16  I=1,IJK
   16 IF(CHECK(3,I).LT.SMALL) SMALL=CHECK(3,I)
      IF(SMALL.LE.(TOT+STEM+IZ1+IZ3)) GO TO 50
      ISW(1,IJK)=SMALL-TOT-STEM-IZ1-IZ3
      STEM=STEM+ISW(1,IJK)
      GO TO 52
   50 BIG=-30
      DO 51 I=1,IJK
   51 IF(CHECK(2,I).GT.BIG) BIG=CHECK(2,I)
      IF(BIG.GE.0) GO TO 52
      IF(IABS(BIG).LE.(TOT+STEM+IZ1+IZ3)) GO TO 52
      ISW(1,IJK)=IABS(BIG)-TOT-STEM-IZ1-IZ3
      STEM=STEM+ISW(1,IJK)
   52 IF(CHECK(2,IJK).GE.0.AND.CHECK(3,IJK).LE.0)GO TO 59
      ISW(2,IJK)=MIN0(IABS(CHECK(3,IJK)),IABS(CHECK(2,IJK)))-IZ1-IZ3-TOT
     1-STEM
      ISW(3,IJK)=MAX0(IABS(CHECK(2,IJK)),IABS(CHECK(3,IJK)))-ISW(2,IJK)-
     1IZ1-IZ3-STEM-TOT
      GO TO 49
   59 ISW(3,IJK)=CHECK(2,IJK)-CHECK(3,IJK)-ISW(2,IJK)-IZ1-IZ3-TOT-STEM
   49 CONTINUE
C       ADD ALL STEPS TOGETHER
   47 COMPAT(K,KK)=TOT+IZ1+IZ2+IZ3
      DO 27 IJ=1,3
      DO 27 IK=1,A
   27 COMPAT(K,KK)=COMPAT(K,KK)+ISW(IJ,IK)
    9 CONTINUE
C       ALMOST FINISHED - SUBTRACT MIN. NO. OF STEPS
      DO 106 K=1,NN
      DO 105 L=1,NN
  105 IF(K.NE.L) COMPAT(K,L)=COMPAT(K,L)-COMPAT(K,K)
  106 COMPAT(K,K)=0
C   COMPAT. MATRIX NOW FINISHED
      DO 108 I=1,NN
      COLSUM(2,I)=0
      ROWSUM(I,2)=0
      ROWSUM(I,1)=0
      COLSUM(1,I)=0
      DO 108 J=1,NN
      IF(COMPAT(J,I).EQ.0.AND.J.NE.I) COLSUM(1,I)=COLSUM(1,I)+1
      IF(COMPAT(I,J).EQ.0.AND.J.NE.I) ROWSUM(I,1)=ROWSUM(I,1)+1
      COLSUM(2,I)=COLSUM(2,I)+COMPAT(J,I)
  108 ROWSUM(I,2)=ROWSUM(I,2)+COMPAT(I,J)
C   ROWS AND COLUMNS SUMMED
C       OUT PUT COMPACT. MATRIX
      WRITE(V,25)
   25 FORMAT(//33H1THIS IS THE COMPATIBILITY MATRIX//)
      IF(NN.GT.38)GO TO 160
      WRITE(V,192)(STATE(I,NT1),I=1,NN)
   13 DO 117 I=1,NN
  117 WRITE(V,116)STATE(I,NT1),(COMPAT(I,J),J=1,NN),(ROWSUM(I,J),J=1,2)
  116 FORMAT(1H ,42I3)
      DO 118 I=1,2
```

```
  118 WRITE(V,192)(COLSUM(I,J),J=1,NN)
  192 FORMAT(4H    ,41I3)
      GO TO 140
  160 WRITE(V,192)(STATE(I,NT1),I=1,38)
      DO 130 I=1,NN
  130 WRITE(V,116)STATE(I,NT1),(COMPAT(I,J),J=1,38)
      DO 131 I=1,2
  131 WRITE(V,192)(COLSUM(I,J),J=1,38)
      WRITE(V,132)
  132 FORMAT(1H1)
      WRITE(V,192)(STATE(I,NT1),I=39,NN)
      DO 133 I=1,NN
  133 WRITE(V,116)STATE(I,NT1),(COMPAT(I,J),J=39,NN),(ROWSUM(I,J),J=1,2)
      DO 134 I=1,2
  134 WRITE(V,192)(COLSUM(I,J),J=39,NN)
  140 CONTINUE
      CALL CLOCK(1)
  156 CONTINUE
      CALL EXIT
      END
$ENTRY          CLADN1
 2
 9 6 HORSE DATA
HORSE1 0 1 2 2 2 3
HORSE2 2 0 1 2 1 2
HORSE3 2 0 1 2 1 2
HORSE4 0 0 1 1 1 1
HORSE5 0 0 1 1 1 1
HORSE6 0 0 1 1-1-1
HORSE7 0 0 1 1 1 1
HORSE8 0 1 2 2 2 2
HORSE9 0 0 0 0 0 0
     2      3      7      8      9     10
1224   FUSULINIDS DATA
FUS  1 1 2 2 3 2 3 3 3 3 3 1 2 1 2 4 4 4 4 4 4 4 3 3 3
FUS  2 0 0 0 1 0 4 4 3 2 4 2 2 2 2 4 4 4 4 4 4 4 3 2 2
FUS  3 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0 0
FUS  4 0 0 0 1 3 2 2 4 2 2 1 2 3 4 2 2 2 2 2 2 3 0 0 0
FUS  5 0 0 1 0 2 2 2 3 1 2 0 2 2 2 2 2 3 3 4 4 4 0 0 0
FUS  6 1 2 0 1 1 3 1 2 1 4 1 2 2 2 3 3 2 3 3 2 3 0 1 1
FUS  7 0 0 1 1 3 2 1 4 2 2 1 4 3 4 1 1 2 1 2 2 2 0 0 0
FUS  8 0 0 1 1 4 4 3 4 4 2 4 4 5 5 2 2 2 2 4 1 1 0 1 0
FUS  9 0 0 1 1 1 1 1 4 3 3 1 2 2 3 2 2 2 3 3 3 4 0 0 0
FUS 10 1 2 0 0 1 4 1 2 1 2 0 1 1 0 2 2 1 2 2 2 2 1 0 1
FUS 11 0 0 2 3 4 2 3 4 4 1 3 4 4 3 2 2 1 1 2 1 0 0 0 0
FUS 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0
     1      2      3      4      5      6      7      8      9     10     11     12     13
    14     15     16     17     18     19     20     21     22     23     24
```

```
$IBSYS
$JOB            0582,BARTCHER                                          ,06000,06000,
$TIME           030
$IBJOB          MAP
$IBFTC CLADN2   NODECK
       COMMON /ALWAYS/JJUNM(14)
       COMMON ZERO,TWO,THREE,FIVE,SIX,LABEL(10),NN,NT,NZ,
      1NN1,NT11,NZ1,CATA(51,31),MMM(62),NOBAD,BAD(50),
      2                                A(52,32),B(31),E(101),D(31),EE(10
      31),NEG,POS,NNNN,IJK,IKK,N,NNNN1,NT1,I,J,K,NN2,MIN,MAX,II,JJ,FIN,JJ
      4J,SUM2,HINUM,MM,M,SUM1,IKKN,IKK1
       INTEGER STR,EQL,STORE(50) ,ZERO,TWO ,THREE,FIVE,SIX
       INTEGER          CATA,BAD,A,B,E,D,EE,FIN,SUM2,HINUM,SUM1
       LOGICAL NEG,POS
C
C     THE PROGRAM IS LIMITED TO NN=50 CHARACTERS AND NT=30 OTU-S
C     BUT COULD BE EXPANDED FOR SYSTEMS WITH GREATER CAPACITY BY INCREASI
C     THE DIMENSIONS.
C     DIMENSIONS FOLLOW-
C         CLADN2
C             LABEL(10),CATA(NN+1,NT+1),MMM(2*NT+2),BAD(NN),A(NN+2,NT+2),
C             B(NT+1),E(2*NN+1),D(NT+1),EE(2*NN+1),STORE(NN),JJUNM(14)
C         TREE
C             LABEL(10),STATE(NN+1,NT+1),OTUNO(2*NT+2),BAD(NN),INFO(NT,NN
C             GP(2*NT+1),CHR(NN)
C
C     THE PROGRAM USES THE FOLLOWING LIBRARY FUNCTIONS-
C          IABS(J)    (INTEGER ABSOLUTE VALUE)
C          EXIT       (TERMINATES THE PROGRAM)
C                CLOCK(I)      (PRINTS TIME SINCE LAST CALL)
C
C     THE FORM OF INPUT IS AS FOLLOWS-
C
C     FIRST CARD       NUMBER OF SEPARATE DATA SETS ON WHICH CLADN2 SHOULD
C                      BE PERFORMED   FORMAT(I2)
C
C                      NN=NUMBER OF CHARACTERS AND NT=NUMBER OF OTU-S
C     SECOND CARD      THE NEXT 60 COLUMNS ARE FOR THE NAME OF THIS SET OF
C                      DATA AND FOR ANY ADDITIONAL COMMENTS     FORMAT(2I2
C
C     THIRD CARD       NUMBER OF *BAD* CHARACTERS AND THE LIST OF *BAD*
C                      CHARACTER NAMES   FORMAT(I2,(13A6))
C
C                      EACH DATA CARD CONTAINS THE CHARACTER NAME AND THE
C                      CHARACTER STATES OF ALL OTU-S FOR THAT PARTICULAR
C     DATA             CHARACTER    FORMAT(A6,(37I2)). IF THERE ARE MORE T
C     CARDS            37 OTU-S A SECOND DATA CARD  (FORMAT(37I2))  WILL
C                      FOLLOW THE DATA CARD FOR EACH CHARACTER. THE CHARAC
C                      NAME IS NOT REPEATED IN THE SECOND CARD. THERE MUST
C                      BE NN DATA CARDS (OR SETS OF DATA CARDS).
C
C     OTU NAME         THESE CONTAIN THE OTU NAMES IN PROPER SEQUENCE
C       CARDS          FORMAT(13A6)
C
C
C     CARDS FOR NEXT DATA SET IF  MORE THAN ONE
C
C      THIS PROGRAM NEEDS THREE SCRATCH TAPES      (0,2,3)
C      READ IN THE NUMBER OF SETS OF DATA
C      FIVE IS THE INPUT UNIT
```

```
C      SIX IS THE OUTPUT UNIT
C      ZERO,TWO,THREE ARE SCRATCH UNITS
       CALL CLOCK(0)
       ZERO=0
       TWO=2
       THREE=3
       FIVE=5
       SIX=6
       READ(FIVE,101)IIIIII
       DO 3692 JJJJJJ=1,IIIIII
C      READ IN THE SIZE OF THE ENTIRE DATA MATRIX , THE NAME OF THE DATA,
C      THE NUMBER OF *BAD* CHARACTERS,AND THE *BAD* CHARACTERS.
       READ(FIVE,101)NNNN,NT,(LABEL(I),I=1,10),NOBAD,(BAD(I),I=1,NOBAD)
  101  FORMAT(2I2,10A6/I2,(13A6))
       IJK=0
       IKK=0
       N=1
       NN=NNNN-NOBAD
       NNNN1=NNNN+1
       NZ=2*NT-1
       NZ1=NZ+1
       NN1=NN+1
       NT11=NT+1
       NT1=NT+1
C      READ IN THE DATA MATRIX ,THE CHARACTER NAMES,AND THE OTU NAMES.
       DO 2 I=1,NNNN
     2 READ(FIVE,103)CATA(I,NT1),(CATA(I,J),J=1,NT)
       READ(FIVE,102)(CATA(NNNN1,I),I=1,NT)
       REWIND ZERO
       WRITE(ZERO)(CATA(NNNN1,I),I=1,NT)
       REWIND ZERO
C      ASSIGN CHARACTER NUMBERS TO BAD CHARACTERS
       I=0
  279  IF(I.GE.NOBAD)GO TO 280
       I=I+1
       DO 276 J=1,NNNN
  276  IF(CATA(J,NT1   ).EQ.BAD(I))GO TO 277
       WRITE(SIX,278)
  102  FORMAT(13A6)
  103  FORMAT(A6,(36I2))
  278  FORMAT(13H ERROR IN BAD)
       CALL EXIT
  277  BAD(I)=J
       GO TO 279
C      WRITE A HEADING
  280  WRITE(SIX,270)(LABEL(I),I=1,10)
  270  FORMAT(26H1MONOTHETIC PROCEDURE FOR ,10A6////30X,12HSYMBOL TABLE /
     1//31X,10HCHARACTERS//25X,6HNUMBER,11X,4HNAME)
  272  FORMAT(27X,I2,12X,A6)
C      ASSIGN NUMBERS TO CHARACTERS
       DO 273 I=1,NNNN
C      ASSIGN NUMBERS TO OTU-S
       WRITE(SIX,272)I,CATA(I,NT1)
  273  CATA(I,NT1)=I
       DO 274 I=1,NT
  274  CATA(NNNN1,I)=I
       WRITE(SIX,275)
  275  FORMAT(1H1)
       IF(NOBAD.LE.0)GO TO 230
C      REMOVE BAD CHARACTERS FROM DATA MATRIX
```

```
      REWIND TWO
      DO 231 I=1,NNNN1
  231 WRITE(TWO)(CATA(I,J),J=1,NT1)
      REWIND TWO
      K=1
      DO 232 I=1,NNNN
      READ(TWO)(CATA(K,J),J=1,NT1)
      DO 233 J=1,NOBAD
  233 IF(CATA (I,NT1).EQ.BAD(J))GO TO 232
      K=K+1
  232 CONTINUE
      READ(TWO)(CATA(NN1,J),J=1,NT1)
      WRITE(SIX,238)(BAD(I),I=1,NOBAD)
  238 FORMAT(33HOTREE FOLLOWS--BAD CHARACTERS ARE  20I4)
      WRITE(SIX,243)
  243 FORMAT(////)
      GO TO 239
  230 WRITE(SIX,240)
  240 FORMAT(34HOTREE FOLLOWS -- NO BAD CHARACTERS ////)
C      START PROCESSING
  239 NT2=NT11+1
      DO 4 I=1,NN1
      DO 4 J=1,NT11
    4 A(I,J)=CATA(I,J)
C     FIRST WE CHECK TO SEE IF ANY CHARACTER HAS A STEP FOR ALL OTU-S
      NN2=NN+2
      DO 34 I=1,NN
      MIN=A(I,1)
      MAX=A(I,1)
      DO 36 J=1,NT
      IF(A(I,J).GT.MAX)MAX=A(I,J)
   36 IF(A(I,J).LT.MIN)MIN=A(I,J)
      IF(MAX*MIN.LE.0)GO TO 34
      IF(MIN.LT.0)MIN=MAX
      DO 37 J=1,NT
   37 A(I,J)=A(I,J)-MIN
   34 CONTINUE
      NT1=NT
C     WE NOW START THE MAIN PROGRAM
C     SUM THE NUMBER OF ZERO STATES FOR EACH OTU
   19 DO 6 I=1,NT1
      A(NN2,I)=0
      DO 6 J=1,NN
    6 IF(A(J,I).EQ.0)A(NN2,I)=A(NN2,I)+1
C     ORDER THESE SUMS
      K=1
    9 J=K
      DO 7 I=K,NT1
    7 IF(A(NN2,J).LT.A(NN2,I))J=I
      DO 8 I=1,NN2
      II=A(I,K)
      A(I,K)=A(I,J)
    8 A(I,J)=II
      K=K+1
      IF(K.LT.NT1)GO TO 9
C     ORDER ANY TIES OF THESE SUMS
C      ACCORDING TO OTU NO.
      K=0
   11 K=K+1
      IF(K.GE.NT1)GO TO 10
```

```
      IF(A(NN2,K).NE.A(NN2,K+1))GO TO 11
      J=K+1
      I=J
   12 J=I-1
      IF(A(NN2,K).NE.A(NN2,I))GO TO 13
      I=I+1
      IF(I.LE.NT1)GO TO 12
      J=NT1
   13 IF(K.GE.J)GO TO 11
      JJ=K
      DO 14 I=K,J
   14 IF(A(NN1,JJ).GT.A(NN1,I))JJ=I
      DO 15 I=1,NN2
      II=A(I,K)
      A(I,K)=A(I,JJ)
   15 A(I,JJ)=II
      K=K+1
      GO TO 13
C      START MONOTHETIC PROCEDURE FOR THIS PASS
   10 JJJ=0
      SUM2=0
      IF(NT1.LE.1)GO TO 91
      HINUM=A(NN2,1)
  706 DO 701 I=1,NT1
      FIN=I-1
  701 IF(A(NN2,I).LT.HINUM)GO TO 702
      II=NT1
      FIN=NT1
      GO TO 705
  702 II=FIN+1
      DO 704 I=1,NN
      MAX=A(I,II)
      MIN=MAX
      DO 703 J=II,NT1
      IF(A(I,J).GT.MAX)MAX=A(I,J)
  703 IF(A(I,J).LT.MIN)MIN=A(I,J)
  704 IF(MAX*MIN.GT.0)GO TO 705
      HINUM=HINUM-1
      IF(HINUM)91,706,706
  705 DO 714 I=1,NN
      J=0
      K=0
      L=0
      DO 710 M=1,FIN
      IF(A(I,M))711,712,713
  711 K=K+1
      GO TO 710
  712 L=L+1
      GO TO 710
  713 J=J+1
  710 CONTINUE
      IF(FIN.LT.NT1)GO TO 825
      A(I,NT2)=L+MIN0(J,K)
      GO TO 714
  825 MAX=A(I,II)
      MIN=MAX
      DO 720 M=II,NT1
      IF(A(I,M).GT.MAX)MAX=A(I,M)
  720 IF(A(I,M).LT.MIN)MIN=A(I,M)
      A(I,NT2)=999
```

```
      IF(MAX*MIN.LE.0)GO TO 714
      M=J
      IF(MIN.GT.0)M=K
      A(I,NT2)=L+M
  714 CONTINUE
      STR=A(1,NT2)
      DO 707 I=1,NN
  707 IF(A(I,NT2).LT.STR)STR=A(I,NT2)
      IF(STR.GE.NT1)GO TO 91
      STORE(1)=1
      EQL=1
      IJ=1
      DO 801 I=2,NN
      IF(A(I,NT2)-A(IJ,NT2))803,802,801
  803 EQL=0
      IJ=I
  802 EQL=EQL+1
      STORE(EQL)=I
  801 CONTINUE
      GO TO 72
   91 N=NT1
      DO75 I=1,N
   75 B(I)=A(NN1,I)
      GO TO 76
   72 K=STR
      DO 77 LM=1,EQL
      L=1
      J=STORE(LM)
      JJ=A(J,FIN+1)
      IF(FIN.LT.NT1)GO TO 811
      LL=0
      ML=0
      DO 812 M=1,FIN
      IF(A(J,M))814,812,813
  814 LL=LL+1
      GO TO 812
  813 ML=ML+1
  812 CONTINUE
      JJ=-1
      IF(ML.GT.LL)JJ=1
  811 DO 810 I=1,FIN
      IF(A(J,I)*JJ.GT.0)GO TO 810
      D(L)=I
      L=L+1
  810 CONTINUE
      SUM1=0
      JJ=0
C      LOCATE OTU-S TO BE THROWN OUT ON THIS PASS
      DO 82 I=1,NN
      MIN=0
      NEG=.FALSE.
      POS=.FALSE.
      DO 81 J=1,NT1
      DO 80 II=1,K
   80 IF(D(II).EQ.J)GO TO 81
      II=A(I,J)
      IF(II)250,82,251
  250 NEG=.TRUE.
      II=-II
      GO TO 252
```

```
  251 POS=.TRUE.
  252 IF(NEG.AND.POS)GO TO 82
      IF(MIN.EQ.0)MIN=II
      IF(II.LT.MIN)MIN=II
   81 CONTINUE
      JJ=JJ+1
      E(JJ)=A(I,NT11)
      JJ=JJ+1
      IF(NEG)MIN=-MIN
      E(JJ)=MIN
      SUM1=SUM1+IABS(MIN)
   82 CONTINUE
      IF(JJ-JJJ)77,85,84
   85 IF(JJ.EQ.0.OR. SUM1.LE.SUM2)GO TO77
   84 DO 83 I=1,JJ
   83 EE(I)=E(I)
      DO 89 I=1,K
      J=D(I)
   89 B(I)=A(NN1,J)
      N=K
      JJJ=JJ
      SUM2=SUM1
   77 CONTINUE
   76 IJK=IJK+1
      IKKN=IKK+2*N
      IKK1=IKK+1
C      CREATE ANOTHER PART OF THE TREE
      J=0
      DO 55 I=IKK1,IKKN,2
      J=J+1
      MMM(I)=B(J)
   55 MMM(I+1)=IJK+1
      MMM(IKKN)=IJK
      IKK=IKKN
C      ARE WE FINISHED     (YES= GO TO 22)
      IF(NT1-N.LE.0)GO TO 22
C      SUBTRACT STEPS FROM PROPER CHAR.
      DO 43 I=1,NN
      DO 44 J=1,JJJ,2
   44 IF(A(I,NT11).EQ.EE(J))GO TO 45
      GO TO 43
   45 DO 46 K=1,NT1
   46 A(I,K)=A(I,K)-EE(J+1)
   43 CONTINUE
C      DELETE PROPER OTU-S
      KK=0
   32 KK=KK+1
      IF(KK.GT.N)GO TO 19
      K=0
   30 K=K+1
      IF(B(KK).NE.A(NN1,K))GO TO 30
      DO 31 I=1,NN1
      II=A(I,K)
      A(I,K)=A(I,NT1)
   31 A(I,NT1)=II
      NT1=NT1-1
      GO TO 32
   22 MMM(IKK)=0
C      THE MONOTHETIC PROCEDURE HAS PRODUCED A TREE (STORED IN MMM(I))
C      WE NOW CALL ON *TREE * TO PUT THE CHARACTERS ONTO THE TREE THEN
```

```
C       TO PRINT THE RESULTS
        CALL TREE
        CALL CLOCK(1)
 3692 CONTINUE
        CALL EXIT
        END
$IBFTC DATBLK  NODECK
        BLOCK DATA
        COMMON /ALWAYS/BLANK,MINUS,EX,NUM(11)
        INTEGER BLANK,EX
        DATA BLANK,MINUS,EX,NUM/1H ,1H-,1H*,1H0,1H1,1H2,1H3,1H4,1H5,1H6,
       11H7,1H8,1H9,4HJUNK/
        END
$IBFTC TREE    NODECK
        SUBROUTINE TREE
        COMMON /ALWAYS/BLANK,MINUS,EX,NUM
        COMMON ZERO,TWO,THREE,FIVE,SIX,LABEL(10),NN,NT,NZ,NN1,NT1,NZ1,
       1STATE(51,31),
       1OTUNO(62),NOBAD,BAD(50),     INFO(30,50),ODATA(51,31),COUNT(50)
        INTEGER ZERO,TWO,THREE,FIVE,SIX
        INTEGER NUM(11),STATE,OTUNO,BAD,ODATA,COUNT,TOP,SEC,GROUP,FIN,ST,R
       1OW,GP1          ,OUTPUT(211),TEMP(211),GP(71),BLANK,MINUS,EX
        LOGICAL OTUFIN,POS     ,TTSS
        INTEGER CHR(50)
        EQUIVALENCE                              (STATE(1,6),TEMP(1)),(STATE(1,1
       12),GP(1)),(STATE(1,15),CHR(1))
        NNNN=NN+NOBAD
        POS=.FALSE.
        IF(NOBAD.LE.0)POS=.TRUE.
C       SET LEND = LOWEST LEVEL NO. IN SECOND PART OF TREE
        IF(NT.GT.18)GO TO 631
        LEND=-2
        GO TO 632
  631 J=36
        MIN=OTUNO(J)
        N=NT-1
        DO 633 I=19,N
        MIN=MINO(MIN,OTUNO(J))
  633 J=J+2
        LEND=MIN-2
  632 TTSS=.TRUE.
C       SET IZ= HIGHEST LEVEL NO. IN TREE
  211 IZ=OTUNO(2)
        DO326 I=4,NZ,2
  326 IF(OTUNO(I).GT.IZ)IZ=OTUNO(I)
C       ORDER STATE ARRAY TO AGREE WITH TREE
        II=0
        I=-1
  341 I=I+2
        II=II+1
        J=II-1
  302 J=J+1
        IF(J.GE.NT1)GO TO 386
        IF(STATE(NN1,J).NE.OTUNO(I))GO TO 302
        DO 333 K=1,NN1
        IT=STATE(K,J)
        STATE(K,J)=STATE(K,II)
  333 STATE(K,II)=IT
        IF(II.LT.NT    )GO TO 341
C       INITIALIZE PARAMETERS
```

```
      TOP=IZ
      WRITE(SIX,498)
  498 FORMAT(6H1LEVEL/5H  NO./)
      NG=7*NT
      JEND=MINO(NG,126)
      REWIND THREE
      DO 450 I=1,NG
  450 OUTPUT(I)=BLANK
      DO 301 J=1, NT1
      DO 301 I=1,NN1
  301 ODATA(I,J)=STATE(I,J)
      DO330 I=1,NN
  330 COUNT(I)=0
C      START LOOP
  303 SEC=-1
      TOP=TOP-1
      LEV=TOP-1
      DO309J=1,NT1
      DO309I=1,NN1
  309 STATE(I,J)=ODATA(I,J)
      DO 335 J=1,NN
      DO 335 I=1,NT
  335 INFO(I,J)=0
      IF(TOP+1)306,305,320
  305 SEC=1
      LEV=-1
C      START INNER LOOP
C      ON FIRST PASS REMOVE ALL STEPS FOR LEVEL LOWER THAN LEV
C      ON SECOND PASS PICK UP STEPS FOR LEVEL LEV
  320 GROUP=0
      SEC=SEC+1
      LEV=LEV+1
      OTUFIN=.FALSE.
      FIN=0
C      SELECT NEXT CLUSTER
  321 GROUP=GROUP+1
      IF(LEV.NE.0)GO TO323
      ST=1
      FIN=NT
      OTUFIN=.TRUE.
      GO TO324
  323 ST=FIN+1
      J=2*ST-2
  325 J=J+2
      IF(OTUNO(J).GT.LEV)GO TO325
      FIN=J/2
      IF(OTUNO(J).EQ.0)OTUFIN=.TRUE.
  324 DO310 ROW=1,NN
      MAXNO=STATE(ROW,ST)
      MINNO=MAXNO
      DO 381 I=ST,FIN
      IF(STATE(ROW,I).GT.MAXNO)MAXNO=STATE(ROW,I)
  381 IF(STATE(ROW,I).LT.MINNO)MINNO=STATE(ROW,I)
      IF(MAXNO*MINNO.LE.0)GO TO310
      NZM=MINNO
      IF (MAXNO.LT.0)NZM=MAXNO
      DO313 J=ST,FIN
  313 STATE(ROW,J)=STATE(ROW,J)-NZM
      IF(SEC.LT.1)GO TO310
      INFO(GROUP,ROW)=NZM
```

```
      COUNT(ROW)=COUNT(ROW)+IABS(NZM)
  310 CONTINUE
      IF(.NOT.OTUFIN)GO TO321
      IF(SEC.LE.0)GO TO 320
C      END OF INNER LOOP
      DO 501 I=1,NN
  501 CHR(I)=STATE(I,NT1)
      IF(LEV.LT.LEND)TTSS=.FALSE.
C      PREPARE TO PRINT PART OF TREE FOR LEVEL LEV
C      GO TO 403 IF THIS IS THE TOP LEVEL
      IF(LEV.EQ.IZ)GO TO 403
      K=1
      DO440 I=2,NZ1,2
      IF(OTUNO(I).GT.LEV+1)GO TO 440
      GP(K)=OTUNO(I)
      K=K+1
  440 CONTINUE
C      JOIN BRANCHES WHICH JOIN AT LEVEL LEV+1
      I=1
      J=1
  443 IF(I.GT.NG)GO TO441
      IF(OUTPUT(I).EQ.EX)GO TO442
      I=I+1
      GO TO443
  442 IF(GP(J).EQ.LEV+1)GO TO 444
      J=J+1
      I=I+1
      GO TO443
  444 I=I+1
  445 OUTPUT(I)=EX
      I=I+1
      IF(OUTPUT(I).NE.EX.AND.I.LE.NG)GO TO445
      IF(I.GT.NG)GO TO441
      IF(OUTPUT(I).EQ.EX)I=I-1
      I=I+1
      J=J+1
      GO TO443
  441 LV=LEV+1
      I=LV/10
      J=LV-I*10
      LV=NUM(I+1)
      IF(I.LE.0)LV=BLANK
      J=NUM(J+1)
      WRITE(SIX,457)LV,J,(OUTPUT(I),I=1,JEND)
      IF(NG.GT.126.AND.TTSS)
     1WRITE(THREE)LV,J,(OUTPUT(I),I=127,NG)
      DO492 I=1,NG
  492 TEMP(I)=OUTPUT(I)
  403 IF(LEV.LT.IZ)GO TO 405
C      PRINT OTU NAMES AND INITIALIZE TEMP
      READ(ZERO)(OUTPUT(I),I=1,NT)
      J=1
      DO 641 I=1,NT
      K=OTUNO(J)
      TEMP(I)=OUTPUT(K)
  641 J=J+2
      WRITE(THREE,610)(TEMP(I),I=1,NT)
  610 FORMAT(18(A6,1X))
      REWIND THREE
      READ(THREE,611)(OUTPUT(I),I=1,NG)
```

```
  611 FORMAT(126A1)
      REWIND THREE
      REWIND ZERO
      DO 404 I=1,NG
  404 TEMP(I)=NUM(11)
      DO 451 I=1,NG,7
      TEMP(I+6)=BLANK
      TEMP(I)=BLANK
  451 TEMP(I+5)=BLANK
      WRITE(SIX,407)(OUTPUT(I),I=1,JEND)
      IF(NG.GT.126.AND.TTSS)
     1WRITE(THREE)BLANK,BLANK,(OUTPUT(I),I=127,NG)
  407 FORMAT(5X,127A1)
      GO TO434
C      SET UP NEW BRANCHES FOR THIS LEVEL
  405 N=1
  436 IF(N.GE.NG)GO TO434
      IF(TEMP(N).EQ.EX.AND.TEMP(N+1).EQ.EX)GO TO430
      N=N+1
      GO TO436
  430 I=N+2
  437 IF(I.GT.NG)GO TO435
      IF(TEMP(I).NE.EX)GO TO435
      I=I+1
      GO TO437
  435 J=(I-N)/2
      DO480 KK=1,J
      TEMP(N)=BLANK
  480 N=N+1
      TEMP(N)=EX
      DO481 KK=1,J
      N=N+1
  481 TEMP(N)=BLANK
      GO TO436
  434 ASSIGN 422 TO KK
      I=1
  482 IF(I.GT.NG)GO TO421
      IF(TEMP(I).EQ.EX)GO TO483
      I=I+1
      GO TO482
  483 TEMP(I-2)=NUM(11)
      TEMP(I-1)=NUM(11)
      TEMP(I  )=NUM(11)
      TEMP(I+1)=NUM(11)
      I=I+2
      GO TO482
  421 DO 408 I=1,NG
  408 OUTPUT(I)=TEMP(I)
      I=1
  410 IF(I.GT.NG)GO TO411
      IF(OUTPUT(I).EQ.NUM(11))GO TO 409
      I=I+1
      GO TO410
  409 OUTPUT(I)=BLANK
      OUTPUT(I+1)=BLANK
      OUTPUT(I+2)=EX
      OUTPUT(I+3)=BLANK
      I=I+4
      GO TO410
  411 WRITE(SIX,407)(OUTPUT(I),I=1,JEND)
```

```
      IF(NG.GT.126.AND.TTSS)
     1WRITE(THREE)BLANK,BLANK,(OUTPUT(I),I=127,NG)
      GO TO KK,(422,303)
C      PUT CHAR. NO. AND STEPS ON BRANCHES FOR THIS LEVEL
  422 J=0
  412 J=J+1
      IF(J.LE.NN)GO TO416
      ASSIGN 303 TO KK
      GO TO421
  416 IND=0
      JI=0
      DO413 I=1,NG
  413 OUTPUT(I)=TEMP(I)
      I=1
  414 IF(I.GT.NG)GO TO415
      IF(OUTPUT(I).EQ.NUM(11))GO TO425
      I=I+1
      GO TO414
  425 JI=JI+1
      JJ=INFO(JI,J)
      IF(JJ.NE.0)GO TO470
      OUTPUT(I   )=BLANK
      OUTPUT(I+1)=BLANK
      OUTPUT(I+2)=EX
      OUTPUT(I+3)=BLANK
      I=I+4
      GO TO414
  470 IND=1
      IF(JJ.GT.0)GO TO426
      JJ=-JJ
      OUTPUT(I+2)=MINUS
      GO TO427
  426 OUTPUT(I+2)=BLANK
  427 JK=JJ/10
      JJ=JJ-JK*10
      OUTPUT(I+3)=NUM(JJ+1)
      IF(JK.LE.0)GO TO 110
      OUTPUT(I+3)=NUM(JK+1)
      OUTPUT(I+4)=NUM(JJ+1)
  110 JJ=CHR(J)/10
      JK=CHR(J)-JJ*10
      IF(JJ.EQ.0)GO TO428
      OUTPUT(I)=NUM(JJ+1)
      GO TO429
  428 OUTPUT(I)=BLANK
  429 OUTPUT(I+1)=NUM(JK+1)
      I=I+4
      GO TO414
  415 IF(IND.EQ.0)GO TO 412
      WRITE(SIX,407)(OUTPUT(JJ),JJ=1,JEND)
      IF(NG.GT.126.AND.TTSS)
     1WRITE(THREE)BLANK,BLANK,(OUTPUT(JJ),JJ=127,NG)
      GO TO412
C      PRINT REST OF TREE
  306 LEV=IZ
  452 OUTPUT(127)=-100
      IF(NG.LE.126)GO TO 105
      WRITE(THREE)BLANK,BLANK,(OUTPUT(I),I=127,NG)
      WRITE(SIX,498)
      REWIND THREE
```

```
  455 READ(THREE)J,K,(OUTPUT(I),I=127,NG)
      IF(OUTPUT(127).EQ.-100)GO TO 105
  454 WRITE(SIX,457)J,K,(OUTPUT(I),I=127,NG)
  457 FORMAT(1X,2A1,2X,127A1)
      GO TO 455
  105 DO 106 J=1,NT1
      DO 106 I=1,NN1
  106 STATE(I,J)=ODATA(I,J)
      REWIND THREE
  100 ITCNT=COUNT(1)
      DO340 I=2,NN
  340 ITCNT=ITCNT+COUNT(I)
C      PRINT COUNTS FOR EACH CHAR.
      WRITE(SIX,331)(STATE(I,NT1),COUNT(I),I=1,NN)
  331 FORMAT(/20H COUNT FOR CHARACTER,I3,3H IS,I3)
      WRITE(SIX,351)ITCNT
  351 FORMAT(/15H TOTAL COUNT IS,I4//)
C      ARE THERE BAD CHAR.   (NO= GO TO 213)
C      IF YES THEN ADD THEN AND PRINT SECOND TREE
      IF(POS)GO TO 213
      POS=.TRUE.
      REWIND TWO
      NN=NNNN
      NN1=NN+1
      DO 210 I=1,NN1
  210 READ(TWO)(STATE(I,J),J=1,NT1)
      WRITE(SIX,212)
    1 FORMAT(40I2)
  212 FORMAT(32HOTREE FOLLOWS-ALL CHARACTERS ON   ////)
      GO TO 632
  102 FORMAT(25I3)
C      PRINT TREE REP. AND RETURN
  213 REWIND ZERO
      READ(ZERO)(OUTPUT(I),I=1,NT)
      J=1
      DO 604 I=1,NT
      K=OTUNO(J)
      OTUNO(J)=OUTPUT(K)
  604 J=J+2
      WRITE(SIX,605)(OTUNO(I),I=1,NZ)
  605 FORMAT(///(10(1H ,A6,I2)))
      RETURN
  386 WRITE(SIX,387) OTUNO(I),I,II,J,NT1
  387 FORMAT(5I10,5HERRRR//////)
      WRITE(SIX,385)(OTUNO(I),I=1,NZ)
      WRITE(SIX,385)(STATE(NN1,I),I=1,NT)
  385 FORMAT(25I5)
      CALL EXIT
      END
$ENTRY          CLADN2
 2
 9 6 HORSE DATA
 0
HORSE1 0 1 2 2 2 3
HORSE2 2 0 1 2 1 2
HORSE3 2 0 1 2 1 2
HORSE4 0 0 1 1 1 1
HORSE5 0 0 1 1 1 1
HORSE6 0 0 1 1-1-1
HORSE7 0 0 1 1 1 1
```

```
HORSE8 0 1 2 2 2 2
HORSE9 0 0 0 0 0 0
     2       3       7       8       9      10
1224    FUSULINIDS DATA
 1FUS 11
FUS   1 1 2 2 3 2 3 3 3 3 3 1 2 1 2 4 4 4 4 4 4 4 3 3 3
FUS   2 0 0 0 1 0 4 4 3 2 4 2 2 2 2 4 4 4 4 4 4 4 3 2 2
FUS   3 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0 0
FUS   4 0 0 0 1 3 2 2 4 2 2 1 2 3 4 2 2 2 2 2 2 3 0 0 0
FUS   5 0 0 1 0 2 2 2 3 1 2 0 2 2 2 2 2 3 3 4 4 4 0 0 0
FUS   6 1 2 0 1 1 3 1 2 1 4 1 2 2 2 3 3 2 3 3 2 3 0 1 1
FUS   7 0 0 1 1 3 2 1 4 2 2 1 4 3 4 1 1 2 1 2 2 2 0 0 0
FUS   8 0 0 1 1 4 4 3 4 4 2 4 4 5 5 2 2 2 2 4 1 1 0 1 0
FUS   9 0 0 1 1 1 1 1 4 3 3 1 2 2 3 2 2 2 3 3 3 4 0 0 0
FUS  10 1 2 0 0 1 4 1 2 1 2 0 1 1 0 2 2 1 2 2 2 2 1 0 1
FUS  11 0 0 2 3 4 2 3 4 4 1 3 4 4 3 2 2 1 1 2 1 0 0 0 0
FUS  12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0
     1       2       3       4       5       6       7       8       9      10      11      12      13
    14      15      16      17      18      19      20      21      22      23      24
$IBSYS
```

```
$IBSYS
$JOB            0582,BARTCHER                                          ,06000,      ,
$TIME           060
$IBJOB          MAP
$IBFTC CLADN3  NODECK
      COMMON /ALWAYS/ JJJJUN(14)
      COMMON /TAPES/ZERO,ONE,TWO,THREE,FIVE,SIX
      INTEGER ZERO,ONE,TWO,THREE,FIVE,SIX
      COMMON /AL1/STATE(41,31),INFO(30,40),OTUNO(62),NN,NT,NZ,NT1,NN1,NZ1
     11                      .
      COMMON /AL2/BLANK(7),IZ,ZX,ITCNT,RITE,ZZXX
      LOGICAL RITE
      INTEGER OLD(62),STATE,OTUNO ,BAD(40),IOTU(62)  ,LABEL(10)
C
C     THE PROGRAM IS LIMITED TO NN=40 CHARACTERS AND NT=30 OTU-S
C     BUT COULD BE EXPANDED FOR SYSTEMS WITH GREATER CAPACITY BY INCREASING
C     THE DIMENSIONS.
C     DIMENSIONS FOLLOW-
C         CLADN3
C            STATE(NN+1,NT+1),INFO(NT,NN),OTUNO(2*NT+2),OLD(2*NT+2),BAD(NN),
C            IOTU(2*NT+2),LABEL(10),BLANK(7)JJJJUN(14)
C         BEST
C            STATE(NN+1,NT+1),INFO(NT,NN),OTUNO(2*NT+2),COUNT(2*NT+2),
C            ODATA(NN+1,NT+1),OLD(2*NT+2)
C         CLUSTR
C            STATE(NN+1,NT+1),INFO(NT,NN),OTUNO(2*NT+2),CCC(NN),DD(2*NT+2),
C            C(NN),ITP(2*NT+2),EXTRA(1100),EXT(NN),NAME(4)
C         TREE
C            STATE(NN+1,NT+1),INFO(NT,NN),OTUNO(2*NT+2),IBLANK(8),IBLA(3),
C            OUTPUT(7*NT+1),TEMP(7*NT+1),GP(2*NT+2)
C
C     THE PROGRAM USES THE FOLLOWING LIBRARY FUNCTIONS-
C          IABS(J)   (INTEGER ABSOLUTE VALUE)
C          MAXO(J,K)     (INTEGER MAXIMUM OF J,K )
C          MINO(J,K)     (INTEGER MINIMUM OF J,K )
C          EXIT          (TERMINATES THE PROGRAM)
C                CLOCK(I)     (PRINTS TIME SINCE LAST CALL)
C
C     THE FORM OF INPUT IS AS FOLLOWS-
C
C     FIRST CARD       NUMBER OF SEPARATE DATA SETS ON WHICH CLADN2 SHOULD
C                      BE PERFORMED   FORMAT(I2)
C
C                      NN=NUMBER OF CHARACTERS AND NT=NUMBER OF OTU-S
C     SECOND CARD      THE NEXT 60 COLUMNS ARE FOR THE NAME OF THIS SET OF
C                      DATA AND FOR ANY ADDITIONAL COMMENTS     FORMAT(2I2,10A6)
C
C     THIRD CARD       NUMBER OF *BAD* CHARACTERS AND THE LIST OF *BAD*
C                      CHARACTER NAMES   FORMAT(I2,(13A6))
C
C                      EACH DATA CARD CONTAINS THE CHARACTER NAME AND THE
C     DATA             CHARACTER STATES OF ALL OTU-S FOR THAT PARTICULAR
C     CARDS            CHARACTER    FORMAT(A6,(30I2)). THERE MUST BE NN DATA
C                      CARDS.
C
C     OTU NAME         THESE CONTAIN THE OTU NAMES IN PROPER SEQUENCE
C       CARDS          FORMAT(13A6)
C
C       TREE           THESE CARDS CONTAIN THE TREE REPRESENTATION
C       CARDS          (FORMAT(10(A6,I2))). (WHERE OTU NAMES GO INTO THE A6
```

```
C                     AND FURCATION LEVELS INTO THE I2)
C
C     PARAMETER       THESE CARDS CONTAIN THE VARIOUS VALUES OF
C       CARDS         N AND K IN FORMAT(2I2).  (SEE EXPLANATION ABOVE)
C
C     CARDS FOR NEXT DATA SET IF  MORE THAN ONE
C
C       ZERO, ONE, TWO, AND THREE ARE SCRATCH TAPES
C       FIVE IS INPUT     SIX IS OUTPUT
      CALL CLOCK(0)
      ZERO=0
      ONE=1
      TWO=2
      THREE=3
      FIVE=5
      SIX=6
C       READ NO. OF DATA SETS
      READ(FIVE,205)IJJJJ
      DO 298 IKKKK=1,IJJJJ
C       READ MATRIX SIZE, LABEL, AND BAD CHAR.
      READ(FIVE,205)NNNN,NT,(LABEL(I),I=1,10),NOBAD,(BAD(I),I=1,NOBAD)
  205 FORMAT(2I2,10A6/I2,(13A6))
      NN=NNNN-NOBAD
      NNNN1=NNNN+1
      NN1=NN+1
      NT1=NT+1
      NZ=2*NT-1
      NZ1=NZ+1
C       READ MATRIX AND CHAR. NAMES
      DO 204 I=1,NNNN
  204 READ(FIVE,227)STATE(I,NT1),(STATE(I,J),J=1,NT)
  227 FORMAT(A6,37I2)
C       READ OTU NAMES
      READ(FIVE,282)(STATE(NNNN1,J),J=1,NT)
  282 FORMAT(13A6)
      REWIND TWO
      WRITE(TWO)(STATE(NNNN1,J),J=1,NT)
      REWIND TWO
C       READ TREE
      READ(FIVE,281)(OTUNO(I),I=1,NZ)
  281 FORMAT(10(A6,I2))
C       SET UP CHAR. AND OTU NO.
      I=0
  279 IF(I.GE.NOBAD)GO TO 280
      I=I+1
      DO 276 J=1,NNNN
  276 IF(STATE(J,NT1) .EQ.BAD(I))GO TO 277
      WRITE(SIX,278)
  278 FORMAT(13H ERROR IN BAD)
      GO TO 299
  277 BAD(I)=J
      GO TO 279
  280 WRITE(SIX,270)(LABEL(I),I=1,10)
  270 FORMAT(26H1NAME OF THIS SET OF DATA=,10A6////30X,12HSYMBOL TABLE /
     1//31X,10HCHARACTERS//25X,6HNUMBER,11X,4HNAME)
  272 FORMAT(27X,I2,12X,A6)
      DO 290 I=1,NZ,2
      DO 291 J=1,NT
  291 IF(STATE(NNNN1,J).EQ.OTUNO(I))GO TO 290
      WRITE(SIX,292)
```

```
  292 FORMAT(18H ERROR IN THE TREE )
      GO TO 299
  290 OTUNO(I)=J
  293 DO 273 I=1,NNNN
      WRITE(SIX,272)I,STATE(I,NT1)
  273 STATE(I,NT1)=I
      DO 274 I=1,NT
  274 STATE(NNNN1,I)=I
      WRITE(SIX,275)
  275 FORMAT(1H1)
      IF(NOBAD.LE.0)GO TO 230
C      REMOVE BAD CHAR.
      REWIND ONE
      DO 231 I=1,NNNN1
  231 WRITE(ONE)(STATE(I,J),J=1,NT1)
      REWIND ONE
      K=1
      DO 232 I=1,NNNN
      READ(ONE)(STATE(K,J),J=1,NT1)
      DO 233 J=1,NOBAD
  233 IF(STATE(I,NT1).EQ.BAD(J))GO TO 232
      K=K+1
  232 CONTINUE
      READ(ONE)(STATE(NN1,J),J=1,NT1)
      WRITE(SIX,238)(BAD(I),I=1,NOBAD)
  238 FORMAT(33HOTREE FOLLOWS--BAD CHARACTERS ARE  20I4)
      WRITE(SIX,243)
  243 FORMAT(////)
      GO TO 239
  230 WRITE(SIX,240)
  240 FORMAT(34HOTREE FOLLOWS -- NO BAD CHARACTERS ////)
  239 OTUNO(NZ1)=0
C      SET PARAMETERS AND PRINT THE INPUTED TREE
      STATE(NN1,NT1)=-1
  283 FORMAT(27H INITIAL TREE--NO PARSIMONY )
      WRITE(SIX,283)
      RITE=.TRUE.
      CALL BEST
C      PRACTICE PARSIMONY BUT DON-T MOVE BRANCHES , THEN PRINT NEW TREE
      RITE=.FALSE.
      CALL BEST
      RITE=.TRUE.
      WRITE(SIX,260)
  260 FORMAT(//1H1,50HFIRST TREE--PARSIMONY PRACTICED--NO BRANCHES MOVED
     1 //)
      CALL BEST
      JCNT=ITCNT
C      BEGIN BRANCH MOVING PROCEDURE
      DO 263 I=1,NZ
  263 IOTU(I)=OTUNO(I)
  259 READ(FIVE,205)N1,K1
      IF(N1.LE.0.OR.K1.LE.0)GO TO 250
      DO 258 II=1,K1
      IJ=JCNT
      WRITE(SIX,257)N1,II
  257 FORMAT(1H1,I2,18H LEVEL MOVE---PASS ,I3////)
      DO 261 I=1,NZ
  261 OLD(I)=OTUNO(I)
      RITE=.FALSE.
      IZ=OTUNO(2)
```

```
      DO 206 I=4,NZ,2
  206 IZ=MAX0(IZ,OTUNO(I))
      IZ=IZ-N1+1
  208 DO 207 JJ=1,N1
      DO 207 I=2,NZ,2
  207 IF(OTUNO(I).GE.IZ)OTUNO(I)=OTUNO(I)-1
      STATE(NN1,NT1)= 1
      CALL BEST
      IF(ITCNT.GE.IJ)GO TO 226
      IJ=ITCNT
      DO 212 I=1,NZ
  212 OLD(I)=OTUNO(I)
  226 IZ=IZ-1
      DO 210 J=1,NZ
  210 OTUNO(J)=OLD(J)
      IF(IZ.GT.N1)GO TO 208
  211 RITE=.TRUE.
      STATE(NN1,NT1)=-1
      CALL BEST
      IF(ITCNT.GE.JCNT)GO TO 265
      JCNT=ITCNT
      DO 258 I=1,NZ
  258 IOTU(I)=OTUNO(I)
      GO TO 259
  265 DO 266 I=1,NZ
  266 OTUNO(I)=IOTU(I)
      GO TO 259
C      BRANCH MOVING FINISHED - PRINT BEST TREE WITH ALL CHAR.
C      (IF THERE ARE ANY BAD CHAR.)
  250 IF(NOBAD.LE.0)GO TO 202
      REWIND ONE
      DO 234 I=1,NNNN1
  234 READ(ONE)(STATE(I,J),J=1,NT1)
      RITE=.TRUE.
      NN=NNNN
      NN1=NN+1
  236 WRITE(SIX,235)
  235 FORMAT(40H1TREE FOLLOWS WITH ALL CHARACTERS PUT ON ////)
      STATE(NN1,NT1)=-1
      CALL BEST
      REWIND TWO
      READ(TWO )(OLD   (I),I=1,NT)
      J=1
      DO 604 I=1,NT
      K=OTUNO(J)
      OTUNO(J)=OLD(K)
  604 J=J+2
      WRITE(SIX,605)(OTUNO(I),I=1,NZ)
  605 FORMAT(///(10(1H ,A6,I2)))
  202 CONTINUE
      CALL CLOCK(1)
  298 CONTINUE
  299 CONTINUE
      CALL EXIT
      END
$IBFTC DATBLK  NODECK
      BLOCK DATA
      COMMON /ALWAYS/BLANK,MINUS,EX,NUM(11)
      INTEGER BLANK,EX
      DATA BLANK,MINUS,EX,NUM/1H ,1H-,1H*,1H0,1H1,1H2,1H3,1H4,1H5,1H6,
```

```
     11H7,1H8,1H9,4HJUNK/
      END
$IBFTC BEST    NODECK
      SUBROUTINE BEST
      COMMON /TAPES/ZERO,ONE,TWO,THREE,FIVE,SIX
      INTEGER ZERO,ONE,TWO,THREE,FIVE,SIX
      COMMON /AL1/STATE(41,31),INFO(30,40),OTUNO(62),NN,NT,NZ,NT1,NN1,NZ
     11
      COMMON /AL2/NEWA,IC1,A,A2,B,L,L1,LKLKO,LEV,ITCNT,RITE,GP2
      INTEGER COUNT(62)
      COMMON ODATA(41,31)
      INTEGER STATE,OTUNO,A,A2,B,COUNT,ODATA          ,C,GP,C2,TIME,GP1,
     1    GP3,B2,GROUP,ST,FIN,ROW,TOP,SEC ,OLD(64) ,ST5131
      LOGICAL RITE,OTUFIN ,GP2 ,TST ,GP22
      ITCNT=0
      TIME=0
      LKLK=LKLKO-1
      ST5131=STATE(NN1,NT1)
C      FIND HIGHEST LEVEL IN TREE
   61 IZ=OTUNO(2)
      DO 26 I=4,NZ,2
   26 IF(OTUNO(I).GT.IZ)IZ=OTUNO(I)
      TIME=TIME+1
C      REARRANGE MATRIX TO AGREE WITH TREE
      II=0
      I=-1
    1 I=I+2
      II=II+1
      J=II-1
    2 J=J+1
      IF(J.GE.NT1)GO TO 186
      IF(STATE(NN1,J).NE.OTUNO(I))GO TO 2
      DO 3 K=1,NN1
      IT=STATE(K,J)
      STATE(K,J)=STATE(K,II)
    3 STATE(K,II)=IT
      IF(II.LT.NT   )GO TO 1
      REWIND ZERO
      TOP=IZ
      IF(ST5131      .GT.0)TOP=LKLK
      DO 301 J=1, NT1
      DO 301 I=1,NN1
  301 ODATA(I,J)=STATE(I,J)
      DO 30 I=1,NN
   30 COUNT(I)=0
      TST=.FALSE.
C      PUT INTO INFO AND THEN ONTO TAPE INFORMATION ABOUT HOW MANY
C      STEPS THERE ARE FOR EACH BRANCH AT THIS LEVEL
  303 SEC=-1
      TOP=TOP-1
      LEV=TOP-1
      DO309J=1,NT1
      DO309I=1,NN1
  309 STATE(I,J)=ODATA(I,J)
      IF(.NOT.(ST5131      .GT.0.AND.TST))GO TO 350
      LEV=LKLK
      GP3=GROUP
      GO TO 27
  350 DO 5 J=1,NN
      DO 5 I=1,NT
```

```
    5 INFO(I,J)=0
      TST=.TRUE.
      IF(TOP+1)306,305,20
  305 SEC=1
      LEV=-1
   20 GROUP=0
      SEC=SEC+1
      LEV=LEV+1
      OTUFIN=.FALSE.
      FIN=0
   21 GROUP=GROUP+1
      IF(LEV.NE.0)GO TO 23
      ST=1
      FIN=NT
      OTUFIN=.TRUE.
      GO TO 24
   23 ST=FIN+1
      J=2*ST-2
   25 J=J+2
      IF(OTUNO(J).GT.LEV)GO TO 25
      FIN=J/2
      IF(OTUNO(J).EQ.0)OTUFIN=.TRUE.
   24 DO 10 ROW=1,NN
      MAXNO=STATE(ROW,ST)
      MINNO=MAXNO
      DO81 I=ST,FIN
      IF(STATE(ROW,I).GT.MAXNO)MAXNO=STATE(ROW,I)
   81 IF(STATE(ROW,I).LT.MINNO)MINNO=STATE(ROW,I)
      IF(MAXNO*MINNO.LE.0)GO TO 10
      NUM=MINNO
      IF (MAXNO.LT.0)NUM=MAXNO
      DO 13 J=ST,FIN
   13 STATE(ROW,J)=STATE(ROW,J)-NUM
      IF(SEC.LT.1)GO TO 10
      INFO(GROUP,ROW)=NUM
      COUNT(ROW)=COUNT(ROW)+IABS(NUM)
   10 CONTINUE
 3232 IF(.NOT.OTUFIN)GO TO 21
      IF(SEC)20,20,302
  302 IF(ST5131       .EQ.0)GO TO 303
      GP1=1
      DO 307 I=2,NZ,2
  307 IF(OTUNO(I).LE.TOP)GP1=GP1+1
      IF(ST5131       .NE.-1)GO TO 303
  308 WRITE(ZERO)GP1,GROUP,((INFO(I,J),I=1,GROUP),J=1,NN)
      GOTO 303
  306 LEV=IZ
      IPCNT=ITCNT
C      SUM COUNTS FOR EACH CHAR.
      ITCNT=COUNT(1)
      DO 40 I=2,NN
   40 ITCNT=ITCNT+COUNT(I)
      IF(.NOT.RITE)IF(TIME-2)62,63,89
      CALL TREE
      WRITE(SIX,660)
  660 FORMAT(1H1)
      WRITE(SIX,31)(STATE(I,NT1),COUNT(I),I=1,NN)
   31 FORMAT(/20H COUNT FOR CHARACTER,I3,3H IS,I3)
      WRITE(SIX,41)ITCNT
   41 FORMAT(/15H TOTAL COUNT IS,I4//)
```

```
      GO TO 99
C      REMOVE EMPTY INTERNODES
   62 L=LEV
      REWIND ZERO
      READ(ZERO)GP1,GROUP,((INFO(I,J),I=1,GROUP),J=1,NN)
   54 IF(L.LE.1)GO TO 61
      L=L-1
      READ(ZERO)GP1,GROUP,((INFO(I,J),I=1,GROUP),J=1,NN)
      GP=0
      C=0
   58 IF(GP.GE.GROUP)GO TO 54
      B=C+2
      DO 56 I=B,NZ1,2
      C=I
   56 IF(OTUNO(I).LE.L)GO TO 57
   57 GP=GP+1
      IF(C.LE.B)GO TO 58
      DO 52 I=1,NN
   52 IF(INFO(GP,I).NE.0)GO TO 58
      C2=C-2
      DO 59 I=B,C2,2
   59 OTUNO(I)=OTUNO(I)-1
      GO TO 58
C      RESOLVE CLUSTERS OF THREE OR MORE OTU
C      (THIS SECTION LOCATES THEM - CLUSTR ROUTINE RESOLVES THEM)
   63 REWIND ZERO
   27 DO 77 I=1,NZ1
   77 OLD(I)=OTUNO(I)
      GP22=.FALSE.
      L=LEV
      JJKK=1
   64 IF(.NOT.(JJKK.EQ.2.AND.ST5131       .GT.0))GO TO 364
      ST5131=0
      TIME=3
      GO TO 61
  364 IF(L.LE.0)GO TO 61
      L=L-1
      IF(ST5131       .EQ.-1)
     1READ(ZERO)GP1,GP3   ,((INFO(I,J),I=1,GP3   ),J=1,NN)
      JJKK=2
      GP=0
      B=0
   68 IF(GP.GE.GP1)GO TO 64
      GP=GP+1
      A=B+2
      DO 66 I=A,64,2
      B=I
   66 IF( OLD (I).LE.L)GO TO 67
   67 IF((B-A)/2+1.LT.3)GO TO 68
      L1=L+1
      B2=B-2
      IC=0
      DO 73 I=A,B2,2
   73 IF( OLD (I).EQ.L1)IC=IC+1
      IF(IC.LT.2)GO TO 68
      IC1=IC+1
      A2=A-2
      NEWA=0
      DO 69 J=2,A2,2
   69 IF( OLD (J).LE.L1)NEWA=NEWA+1
```

```
      NEWA=NEWA+1
      IF(A2.LE.0)NEWA=1
      CALL CLUSTR
      IF(GP2)GP22=.TRUE.
      GO TO 68
   89 IF(GP22.AND.ST5131       .EQ.-1)GO TO 63
      IF(.NOT.GP22)GO TO 29
      ST5131=1
      LKLK=LKLK+1
      GO TO 61
   29 CONTINUE
   99 CONTINUE
   72 RETURN
  186 WRITE(SIX,187)OTUNO(I)
  187 FORMAT(/I10,5HERRRR//////)
      WRITE(SIX,185)(OTUNO(I),I=1,NZ)
  185 FORMAT(1X,25I3)
      CALL EXIT
      END
$IBFTC CLUSTR  NODECK
      SUBROUTINE CLUSTR
      COMMON /TAPES/ZERO,ONE,TWO,THREE,FIVE,SIX
      INTEGER ZERO,ONE,TWO,THREE,FIVE,SIX
      COMMON /AL1/STATE(41,31),INFO(30,40),OTUNO(62),NN,NT,NZZ,NT1,NN1,
     1NZZ1
      COMMON /AL2/NEWA,IC1,A,A2,BB,LLL,L1,NAME(4),REPEAT
      COMMON CCC(40),DD(62),C(40),ITP(62),EXTRA(1100),EXT(40)
      INTEGER H
      INTEGER STATE,OTUNO,A,A2,BB,DD,C,CCC,B,DDD,D,E,F,G,CC    ,EXTRA,EXT
      LOGICAL REPEAT
      REPEAT=.FALSE.
      IAR=1
C      THIS ROUTINE RESOLVES LARGE CLUSTERS AND PRINTS OUT ANY EQUALLY
C      PARSIMONIOUS SOLUTIONS AND THEN SELECTS THE FIRST OF THESE
   41 B=NEWA-1+IC1
      IMAX=0
      IC=IC1-1
      MM=IC1
      K=1
   67 MAX=0
      KONT=0
      K=K+1
      MM=MM*(IC1-K+1)/K
      DO61 I=1,K
   61 DD(I)=I
      DO 6 M=1,MM
      CC=0
      IF(K.EQ.2)GO TO 301
      DO 302 I=1,K
      H=DD(I)
      DO 303 J=1,JONT
  303 IF(H.EQ.EXTRA(J+1000))GO TO 302
      GO TO 69
  302 CONTINUE
  301 DO 3 J=1,NN
      D=10
      E=-10
      DO 1 KK=NEWA,B
      DO 2 I=1,K
    2 IF(KK.EQ.NEWA-1+DD(I))GO TO 44
```

```
      GO TO 1
   44 IF(E.LT.INFO(KK,J))E=INFO(KK,J)
      IF(D.GT.INFO(KK,J))D=INFO(KK,J)
    1 CONTINUE
    3 IF(D*E.GT.0)CC=CC+MIN0(IABS(E),IABS(D))
      IF(CC.LE.0)GO TO 69
      IF(CC-MAX)69,91,92
   92 KONT=0
   91 DO 62 I=1,K
      KONT=KONT+1
      EXTRA(KONT)=DD(I)
   62 IF(CC.NE.MAX)CCC(I)=DD(I)
      MAX=CC
   69 IF(DD(K).GE.IC1) GO TO 63
      DD(K)=DD(K)+1
      GO TO 6
   63 K1=K-1
      DO64 J=1,K1
      II=K-J
   64 IF(DD(II).LT.IC1-J) GO TO 65
      GO TO 200
   65 DD(II)=DD(II)+1
      I1=II+1
      DO66 J=I1,K
      IK=J-I1+1
   66 DD(J)=DD(II)+IK
  200 IF(MAX.EQ.0.AND.K.EQ.2)RETURN
    6 CONTINUE
      IF(MAX.LT.IMAX)GO TO 77
      DO 197 I=1,K
  197 ITP(I)=CCC(I)
      IMAX=MAX
      K2=K
      JONT=KONT
      DO 90 I=1,KONT
   90 EXTRA(I+1000)=EXTRA(I)
      IF(K.LT.IC)GO TO 67
   77 K1=K2
      IF(IMAX.LE.0)RETURN
      IF(K1.GE.3)REPEAT=.TRUE.
      K=JONT/K1
      IF(K.LE.1)GO TO 40
      WRITE(SIX,49)L1,K,IAR
   49 FORMAT(9H-AT LEVEL ,I4,10H THERE ARE ,I4,27H EQUALLY PARSIMONIOUS
     1CASES  ,11H FOR STAGE  ,I2/)
      READ(TWO)(DD(I),I=1,NT)
      REWIND TWO
      JJ=1
      I=A-1
   17 J=I-1
   12 J=J+2
      IF(J-BB)16,13,15
   16 IF(OTUNO(J).GT.L1)GO TO 12
   13 II=0
      DO 225 IJ=I,J,2
      I1=OTUNO(IJ)
      II=II+1
  225 C(II)=DD(I1)
      WRITE(SIX,14)JJ,(C(IJ),IJ=1,II)
   14 FORMAT(/7H GROUP ,I2,4H IS ,17(1H*,A6)/12X,17(1H*,A6))
```

```
      JJ=JJ+1
      I=J+1
      GO TO 17
   15 J=1001
      DO 23 I=1,K
      JJ=J+K1-1
      WRITE(SIX,25)(EXTRA(II),II=J,JJ)
   23 J=JJ+1
   25 FORMAT(10X,20I5)
   40 IAR=IAR+1
      DO 68 I=1,K1
   68 DD(I)=ITP(I)
      IK=NEWA
      DO 9 K=1,K1
      M=DD(K)-1+IK
      IF(M     .EQ.NEWA)GO TO 9
      DO 8 I=1,NN
    8 C(I)=INFO(M,I)
      L=M-1
      DO 10 I=NEWA,L
      J=M-I+NEWA
      DO 10II=1,NN
   10 INFO (J,II)=INFO(J-1,II)
      DO 11 I=1,NN
   11 INFO(NEWA,I)=C(I)
    9 NEWA=NEWA+1
      IK1=IK+K1-1
      DO 70 I=1,NN
      D=INFO(IK,I)
      E=D
      DO 71 J=IK,IK1
      H=INFO(J,I)
      IF(H-E)50,71,51
   51 E=H
      GO TO 71
   50 IF(H.LT.D)D=H
   71 CONTINUE
      C(I)=0
      H=1
      IF(E.LT.0)H=-1
   70 IF(D*E.GT.0)C(I)=MIN0(IABS(D),IABS(E))*H
      NEWA=IK
      I4=NEWA+K1
      I5=NEWA+IC1-1
      DO 72 I=I4,I5
      JJ=NEWA+I-I4
      DO 72 J=1,NN
   72 INFO(JJ,J)=INFO(I,J)
      JJ=NEWA+IC1-K1
      DO 73 I=1,NN
   73 INFO(JJ,I)=C(I)
      NW=OTUNO(BB)
      JJ=0
      J=1
      KJ=0
      JK=A
      IA=A-1
      DO 30 I=JK,BB,2
      IF (OTUNO(I).GT.L1) GO TO 30
      KJ=KJ+1
```

```
      IF(KJ.NE.DD(J)) GO TO 31
      J=J+1
      DO 32 M=IA,I,2
      JJ=JJ+1
      C(JJ)=OTUNO(M)
      JJ=JJ+1
      C(JJ)=OTUNO(M+1)+1
      OTUNO(M)=-1
   32 OTUNO(M+1)=-1
      C(JJ)=L1+1
      IF(J.GT.K1) GO TO 33
   31 IA=I+1
   30 CONTINUE
   33 C(JJ)=NW
      IX=A2
      IF(A2.LE.0)IX=1
      DO 38 I=IX,BB
      IF(OTUNO(I).NE.-1)GO TO 38
      DO 52 J=I,BB
      IF(OTUNO(J).EQ.-1)GO TO 52
      IF(J.EQ.BB)OTUNO(J)=L1
      IF(J.EQ.NZZ1)OTUNO(J)=L1
      OTUNO(I)=OTUNO(J)
      OTUNO(J)=-1
      GO TO 38
   52 CONTINUE
      GO TO 56
   38 CONTINUE
   56 DO 53 I=IX,BB
   53 IF(OTUNO(I).EQ.-1)GO TO 54
   54 DO 55 J=1,JJ
      OTUNO(I)=C(J)
   55 I=I+1
      IC1=IC1-K1+1
      IF(IC1.GT.2)GO TO 41
      RETURN
      END
$IBFTC TREE     NODECK
      SUBROUTINE TREE
      COMMON /TAPES/ZERO,ONE,TWO,THREE,FIVE,SIX
      INTEGER ZERO,ONE,TWO,THREE,FIVE,SIX
      COMMON /AL1/STATE(41,31),INFO(30,40),OTUNO(62),NN,NT,NZ,NT1,NN1,NZ
     11
      COMMON /AL2/IBLANK(8),LEV,IBLA(3)
      COMMON OUTPUT(211),TEMP(211),GP(62)
      INTEGER STATE,OTUNO,OUTPUT,TEMP,GP,EX,GROUP,GP1,BLANK ,Q,R,S
      COMMON /ALWAYS/BLANK,MINUS,EX,NUM(11)
      JST=LEV
      LOGICAL TTSS
C      THIS ROUTINE IS SIMILAR TO THE TREE PRINTING ROUTINE OF CLADN2
C      BUT INFORMATION FOR EACH LEVEL IS CALCULATED BY BEST ROUTINE
C      AND STORED ON TAPE
      TTSS=.TRUE.
      IF(NT.GT.18)GO TO 631
      LEND=-2
      GO TO 632
  631 J=36
      MIN=OTUNO(J)
      N=NT-1
      DO 633 I=19,N
```

```
      MIN=MINO(MIN,OTUNO(J))
  633 J=J+2
      LEND=MIN-2
  632 WRITE(SIX,498)
  498 FORMAT(6H1LEVEL/5H  NO./)
      REWIND ZERO
      REWIND THREE
      NG=7*NT
      JEND=MINO(126,NG)
      DO 450 I=1,NG
      OUTPUT(I)=BLANK
  450 TEMP(I)=BLANK
      L=LEV+1
  420 L=L-1
      IF(L.LT.LEND)TTSS=.FALSE.
      IF(L.LT.0)GO TO 452
      READ(ZERO)GP1,GROUP,((INFO(I,J),I=1,GROUP),J=1,NN)
      IF(L.EQ.LEV)GO TO 403
      K=1
      DO440 I=2,NZ1,2
      IF(OTUNO(I).GT.L+1)GO TO440
      GP(K)=OTUNO(I)
      K=K+1
  440 CONTINUE
      I=1
      J=1
  443 IF(I.GT.NG)GO TO441
      IF(OUTPUT(I).EQ.EX)GO TO442
      I=I+1
      GO TO443
  442 IF(GP(J).EQ.L+1)GO TO444
      J=J+1
      I=I+1
      GO TO443
  444 I=I+1
  445 OUTPUT(I)=EX
      I=I+1
      IF(OUTPUT(I).NE.EX.AND.I.LE.NG)GO TO445
      IF(I.GT.NG)GO TO441
      IF(OUTPUT(I).EQ.EX)I=I-1
      I=I+1
      J=J+1
      GO TO443
  441 WRITE(SIX,7)JST,(OUTPUT(I),I=1,JEND)
      I=JST/10
      J=JST-10*I
      LV=NUM(I+1)
      IF(I.LE.0)LV=BLANK
      I=NUM(J+1)
      IF(NG.GT.126.AND.TTSS)
     1WRITE(THREE)LV,I,(OUTPUT(I),I=127,NG)
      JST=JST-1
    7 FORMAT(1X,I2,2X,127A1)
      DO492 I=1,NG
  492 TEMP(I)=OUTPUT(I)
  403 IF(L.LT.LEV)GO TO 405
      READ(TWO)(OUTPUT(I),I=1,NT)
      J=1
      DO 641 I=1,NT
      K=OTUNO(J)
```

```
      TEMP(I)=OUTPUT(K)
  641 J=J+2
      WRITE(THREE,610)(TEMP(I),I=1,NT)
  610 FORMAT(18(A6,1X))
      REWIND THREE
      READ(THREE,611)(OUTPUT(I),I=1,NG)
  611 FORMAT(126A1)
      REWIND TWO
      REWIND THREE
      WRITE(SIX,407)(OUTPUT(I),I=1,JEND)
      IF(NG.GT.126.AND.TTSS)
     1WRITE(THREE)BLANK,BLANK,(OUTPUT(I),I=127,NG)
      DO 404 I=1,NG
  404 TEMP(I)=NUM(11)
      DO 451 I=1,NG,7
      TEMP(I+6)=BLANK
      TEMP(I)=BLANK
  451 TEMP(I+5)=BLANK
  407 FORMAT(5X,127A1)
      GO TO434
  405 N=1
  436 IF(N.GE.NG)GO TO434
      IF(TEMP(N).EQ.EX.AND.TEMP(N+1).EQ.EX)GO TO430
      N=N+1
      GO TO436
  430 I=N+2
  437 IF(I.GT.NG)GO TO435
      IF(TEMP(I).NE.EX)GO TO435
      I=I+1
      GO TO437
  435 J=(I-N)/2
      DO480 KK=1,J
      TEMP(N)=BLANK
  480 N=N+1
      TEMP(N)=EX
      DO481 KK=1,J
      N=N+1
  481 TEMP(N)=BLANK
      GO TO436
  434 ASSIGN 422 TO KK
      I=1
  482 IF(I.GT.NG)GO TO421
      IF(TEMP(I).EQ.EX)GO TO483
      I=I+1
      GO TO482
  483 TEMP(I-2)=NUM(11)
      TEMP(I-1)=NUM(11)
      TEMP(I  )=NUM(11)
      TEMP(I+1)=NUM(11)
      I=I+2
      GO TO482
  421 DO 408 I=1,NG
  408 OUTPUT(I)=TEMP(I)
      I=1
  410 IF(I.GT.NG)GO TO411
      IF(OUTPUT(I).EQ.NUM(11))GO TO 409
      I=I+1
      GO TO410
  409 OUTPUT(I)=BLANK
      OUTPUT(I+1)=BLANK
```

```
      OUTPUT(I+2)=EX
      OUTPUT(I+3)=BLANK
      I=I+4
      GO TO410
  411 WRITE(SIX,407)(OUTPUT(I),I=1,JEND)
      IF(NG.GT.126.AND.TTSS)
     1WRITE(THREE)BLANK,BLANK,(OUTPUT(I),I=127,NG)
      GO TO KK,(422,420)
  422 J=0
  412 J=J+1
      IF(J.LE.NN)GO TO416
      ASSIGN 420 TO KK
      GO TO421
  416 IND=0
      JI=0
      DO413 I=1,NG
  413 OUTPUT(I)=TEMP(I)
      I=1
  414 IF(I.GT.NG)GO TO415
      IF(OUTPUT(I).EQ.NUM(11))GO TO425
      I=I+1
      GO TO414
  425 JI=JI+1
      JJ=INFO(JI,J)
      IF(JJ.NE.0)GO TO470
      OUTPUT(I   )=BLANK
      OUTPUT(I+1)=BLANK
      OUTPUT(I+2)=EX
      OUTPUT(I+3)=BLANK
      I=I+4
      GO TO414
  470 IND=1
      IF(JJ.GT.0)GO TO426
      JJ=-JJ
      OUTPUT(I+2)=MINUS
      GO TO427
  426 OUTPUT(I+2)=BLANK
  427 JK=JJ/10
      JJ=JJ-JK*10
      OUTPUT(I+3)=NUM(JJ+1)
      IF(JK.EQ.0)GO TO 110
      OUTPUT(I+3)=NUM(JK+1)
      OUTPUT(I+4)=NUM(JJ+1)
  110 JJ=STATE(J,NT1)/10
      JK=STATE(J,NT1)-JJ*10
      IF(JJ.EQ.0)GO TO428
      OUTPUT(I)=NUM(JJ+1)
      GO TO429
  428 OUTPUT(I)=BLANK
  429 OUTPUT(I+1)=NUM(JK+1)
      I=I+4
      GO TO414
  415 IF(IND.EQ.0)GO TO 412
      WRITE(SIX,407)(OUTPUT(JJ),JJ=1,JEND)
      IF(NG.GT.126.AND.TTSS)
     1WRITE(THREE)BLANK,BLANK,(OUTPUT(JJ),JJ=127,NG)
      GO TO412
  452 OUTPUT(127)=-100
      WRITE(THREE)BLANK,BLANK,(OUTPUT(I),I=127,NG)
      IF(NG.LE.126)RETURN
```

```
      WRITE(SIX,457)
  457 FORMAT(1H1)
      REWIND THREE
  455 READ(THREE)II,JJ,(OUTPUT(I),I=127,NG)
      IF(OUTPUT(127).EQ.-100)RETURN
  454 WRITE(SIX,500)II,JJ,(OUTPUT(I),I=127,NG)
  500 FORMAT(1X,2A1,2X,127A1)
      GO TO 455
      END
$ENTRY          CLADN3
 2
 9 6 HORSE DATA
 0
HORSE1 0 1 2 2 2 3
HORSE2 2 0 1 2 1 2
HORSE3 2 0 1 2 1 2
HORSE4 0 0 1 1 1 1
HORSE5 0 0 1 1 1 1
HORSE6 0 0 1 1-1-1
HORSE7 0 0 1 1 1 1
HORSE8 0 1 2 2 2 2
HORSE9 0 0 0 0 0 0
     2      3      7      8      9     10
     2 1      3 2      7 4      9 3      8 4     10 0
 1 5
 2 5
 0 0
1224   FUSULINIDS DATA
 1FUS 11
FUS   1 1 2 2 3 2 3 3 3 3 3 1 2 1 2 4 4 4 4 4 4 4 3 3 3
FUS   2 0 0 0 1 0 4 4 3 2 4 2 2 2 2 4 4 4 4 4 4 4 3 2 2
FUS   3 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 0 0
FUS   4 0 0 0 1 3 2 2 4 2 2 1 2 3 4 2 2 2 2 2 2 3 0 0 0
FUS   5 0 0 1 0 2 2 2 3 1 2 0 2 2 2 2 2 3 3 4 4 4 0 0 0
FUS   6 1 2 0 1 1 3 1 2 1 4 1 2 2 2 3 3 2 3 3 2 3 0 1 1
FUS   7 0 0 1 1 3 2 1 4 2 2 1 4 3 4 1 1 2 1 2 2 2 0 0 0
FUS   8 0 0 1 1 4 4 3 4 4 2 4 4 5 5 2 2 2 2 4 1 1 0 1 0
FUS   9 0 0 1 1 1 1 1 4 3 3 1 2 2 3 2 2 2 3 3 3 4 0 0 0
FUS  10 1 2 0 0 1 4 1 2 1 2 0 1 1 0 2 2 1 2 2 2 2 1 0 1
FUS  11 0 0 2 3 4 2 3 4 4 1 3 4 4 3 2 2 1 1 2 1 0 0 0 0
FUS  12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0
     1      2      3      4      5      6      7      8      9     10     11     12     13
    14     15     16     17     18     19     20     21     22     23     24
     1 2      2 2     22 2     24 1     23 2      3 3      4 5     11 4      5 5      9 6
     7 7     13 8     14 9     1210      811      612     1013     1714     1516     1615
    1816     2017     1918     21 0

 1 5
 2 5
 0 0
```

COMPATIBILITY MATRIX FOR HORSE DATA

SYMBOL TABLE

CHARACTERS

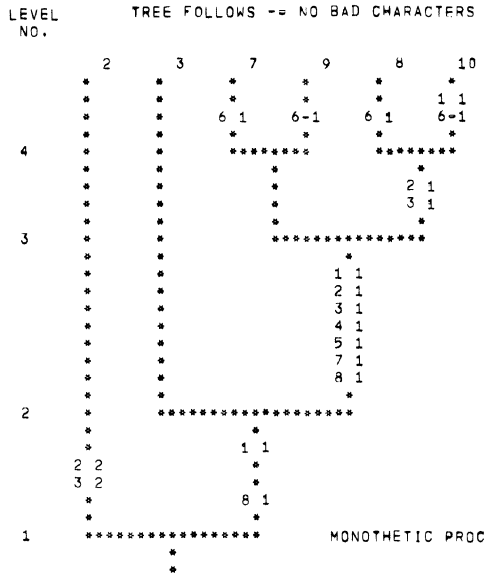
| NUMBER | NAME |
|---|---|
| 1 | HORSE1 |
| 2 | HORSE2 |
| 3 | HORSE3 |
| 4 | HORSE4 |
| 5 | HORSE5 |
| 6 | HORSE6 |
| 7 | HORSE7 |
| 8 | HORSE8 |
| 9 | HORSE9 |

THIS IS THE COMPATIBILITY MATRIX

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 3 | 3 | 1 | 1 | 1 | 1 | 0 | 0 | 2 | 10 |
| 2 | 3 | 0 | 0 | 2 | 2 | 3 | 2 | 2 | 0 | 2 | 14 |
| 3 | 3 | 0 | 0 | 2 | 2 | 3 | 2 | 2 | 0 | 2 | 14 |
| 4 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 2 |
| 5 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 2 |
| 6 | 1 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 5 |
| 7 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 2 |
| 8 | 0 | 3 | 3 | 1 | 1 | 1 | 1 | 0 | 0 | 2 | 10 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8 | 0 |
| | 5 | 2 | 2 | 4 | 4 | 4 | 4 | 6 | 8 | | |
| | 7 | 11 | 11 | 6 | 6 | 8 | 6 | 4 | 0 | | |

1 TIME 0.0311 MIN

Figure 1. - Output of Recent horses from CLADN1.




COUNT FOR CHARACTER 1 IS 3

COUNT FOR CHARACTER 2 IS 4

COUNT FOR CHARACTER 3 IS 4

COUNT FOR CHARACTER 4 IS 1

COUNT FOR CHARACTER 5 IS 1

COUNT FOR CHARACTER 6 IS 4

COUNT FOR CHARACTER 7 IS 1

COUNT FOR CHARACTER 8 IS 2

COUNT FOR CHARACTER 9 IS 0

TOTAL COUNT IS 20

2 1 3 2 7 4 9 3 8 4 10

1 TIME 0.0936 MIN

SYMBOL TABLE

CHARACTERS

| NUMBER | NAME |
|---|---|
| 1 | HORSE1 |
| 2 | HORSE2 |
| 3 | HORSE3 |
| 4 | HORSE4 |
| 5 | HORSE5 |
| 6 | HORSE6 |
| 7 | HORSE7 |
| 8 | HORSE8 |
| 9 | HORSE9 |

Figure 2. - Output of Recent horses from CLADN2.

```
TREE FOLLOWS -= NO BAD CHARACTERS

INITIAL TREE--NO PARSIMONY

LEVEL
 NO.
          2       3       7       9       8      10
        *       *       *       *       *       *
        *       *       *       *       *     1 1
        *       *     6 1     6-1     6 1     6=1
        *       *       *       *       *       *
 4      *       *       ********        ********
        *       *           *               *
        *       *           *             2 1
        *       *           *             3 1
        *       *           *               *
 3      *       *           ****************
        *       *                   *
        *       *                 1 1
        *       *                 2 1
        *       *                 3 1
        *       *                 4 1
        *       *                 5 1
        *       *                 7 1
        *       *                 8 1
        *       *                   *
 2      *       ********************
        *               *
        *             1 1
      2 2             *
      3 2             *
        *             8 1
        *               *
 1      ****************
              *
              *

COUNT FOR CHARACTER  1 IS  3

COUNT FOR CHARACTER  2 IS  4

COUNT FOR CHARACTER  3 IS  4

COUNT FOR CHARACTER  4 IS  1

COUNT FOR CHARACTER  5 IS  1

COUNT FOR CHARACTER  6 IS  4

COUNT FOR CHARACTER  7 IS  1

COUNT FOR CHARACTER  8 IS  2

COUNT FOR CHARACTER  9 IS  0

TOTAL COUNT IS  20
```

NAME OF THIS SET OF DATA= HORSE DATA

SYMBOL TABLE

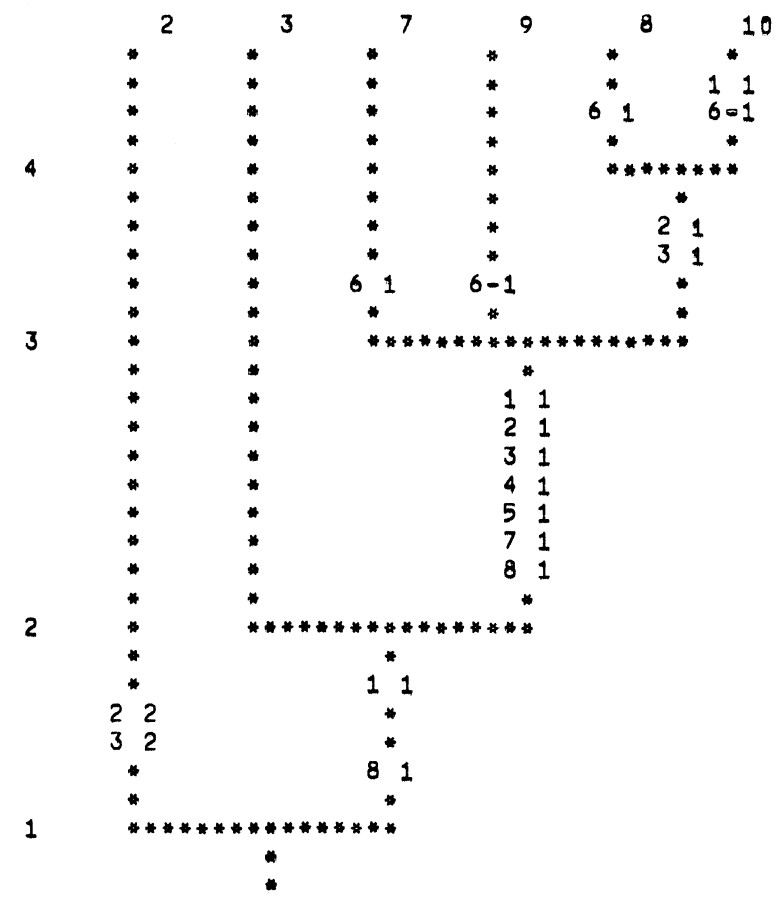
CHARACTERS

| NUMBER | NAME |
|---|---|
| 1 | HORSE1 |
| 2 | HORSE2 |
| 3 | HORSE3 |
| 4 | HORSE4 |
| 5 | HORSE5 |
| 6 | HORSE6 |
| 7 | HORSE7 |
| 8 | HORSE8 |
| 9 | HORSE9 |

A

Figure 3. - Output of Recent horses from CLADN3.

FIRST TREE--PARSIMONY PRACTICED--NO BRANCHES MOVED




COUNT FOR CHARACTER 1 IS 3

COUNT FOR CHARACTER 2 IS 4

COUNT FOR CHARACTER 3 IS 4

COUNT FOR CHARACTER 4 IS 1

COUNT FOR CHARACTER 5 IS 1

COUNT FOR CHARACTER 6 IS 4

COUNT FOR CHARACTER 7 IS 1

COUNT FOR CHARACTER 8 IS 2

COUNT FOR CHARACTER 9 IS 0

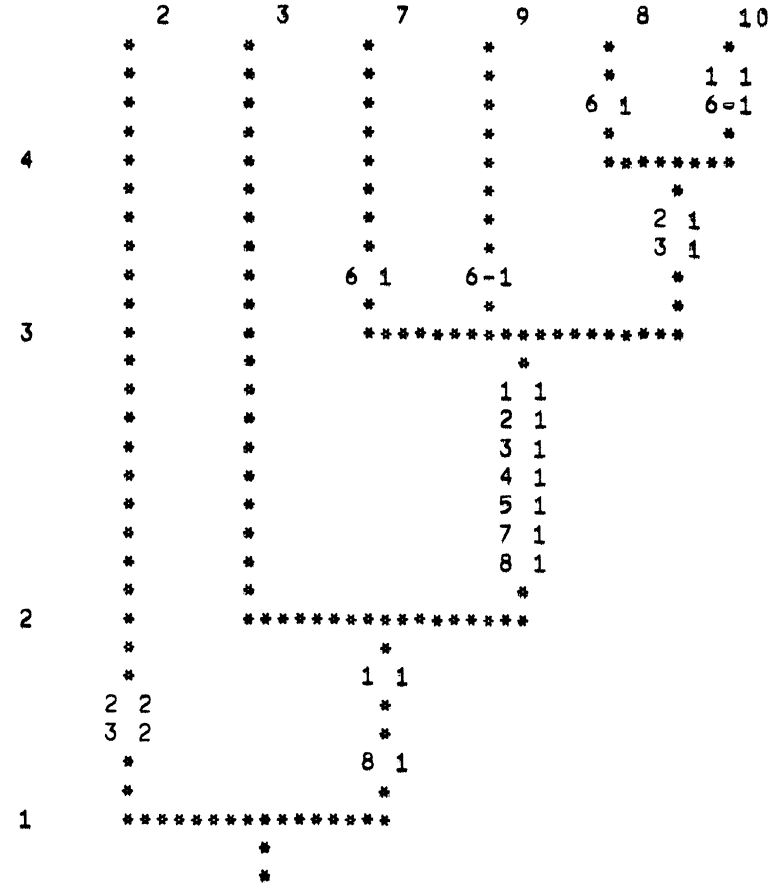
TOTAL COUNT IS 20

B

```
 1 LEVEL MOVE-==PASS  1
AT LEVEL   1 THERE ARE   2 EQUALLY PARSIMONIOUS CASES FOR STAGE  1
GROUP  1 IS *      2*

GROUP  2 IS *      3*

GROUP  3 IS *      7*      9*      8*     10*
             1     3
             2     3

LEVEL
 NO.
          2       3       7       9       8      10
        *       *       *       *       *       *
        *       *       *       *       *     1 1
        *       *       *       *     6 1     6=1
        *       *       *       *       *       *
 4      *       *       *       *       ********
        *       *       *       *           *
        *       *       *       *         2 1
        *       *       *       *         3 1
        *       *     6 1     6-1           *
        *       *       *       *           *
 3      *       *       ********************
        *       *                 *
        *       *               1 1
        *       *               2 1
        *       *               3 1
        *       *               4 1
        *       *               5 1
        *       *               7 1
        *       *               8 1
        *       *                 *
 2      *       ******************
        *               *
        *             1 1
      2 2             *
      3 2             *
        *             8 1
        *             *
 1      ****************
               *
               *
COUNT FOR CHARACTER  1 IS  3

COUNT FOR CHARACTER  2 IS  4

COUNT FOR CHARACTER  3 IS  4

COUNT FOR CHARACTER  4 IS  1

COUNT FOR CHARACTER  5 IS  1

COUNT FOR CHARACTER  6 IS  4

COUNT FOR CHARACTER  7 IS  1

COUNT FOR CHARACTER  8 IS  2

COUNT FOR CHARACTER  9 IS  0

TOTAL COUNT IS  20
```

C

```
 2 LEVEL MOVE-=-PASS  1

LEVEL
 NO.
```




```
COUNT FOR CHARACTER  1 IS  3

COUNT FOR CHARACTER  2 IS  4

COUNT FOR CHARACTER  3 IS  4

COUNT FOR CHARACTER  4 IS  1

COUNT FOR CHARACTER  5 IS  1

COUNT FOR CHARACTER  6 IS  4

COUNT FOR CHARACTER  7 IS  1

COUNT FOR CHARACTER  8 IS  2

COUNT FOR CHARACTER  9 IS  0

TOTAL COUNT IS  20
```

D

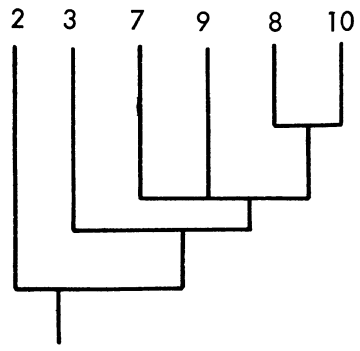



Figure 4. - Representation of output of Recent horses from CLADN3 as shown in Figure 3b.

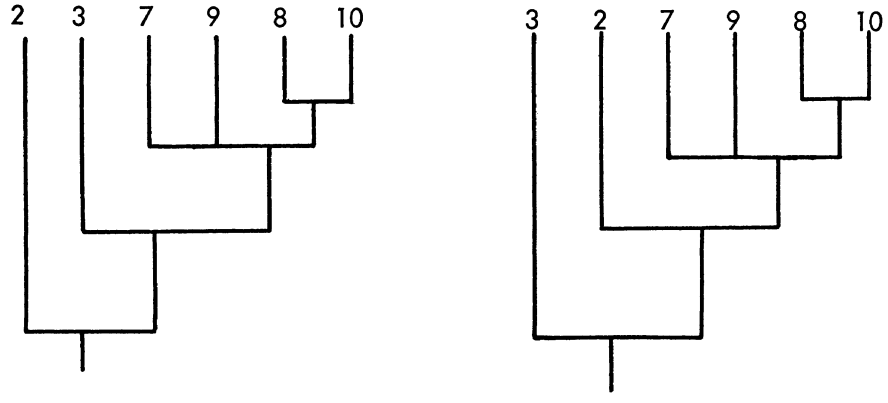



Figure 5. - Two possible interpretations on information shown in Figure 4.

COMPATIBILITY MATRIX FOR FUSULINIDS DATA THIS IS THE COMPATIBILITY MATRIX

SYMBOL TABLE

CHARACTERS

| NUMBER | NAME |
|---|---|
| 1 | FUS 1 |
| 2 | FUS 2 |
| 3 | FUS 3 |
| 4 | FUS 4 |
| 5 | FUS 5 |
| 6 | FUS 6 |
| 7 | FUS 7 |
| 8 | FUS 8 |
| 9 | FUS 9 |
| 10 | FUS 10 |
| 11 | FUS 11 |
| 12 | FUS 12 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 5 | 3 | 9 | 8 | 7 | 8 | 11 | 9 | 6 | 10 | 3 | 0 79 |
| 2 | 7 | 0 | 4 | 12 | 10 | 11 | 12 | 17 | 11 | 10 | 15 | 4 | 0113 |
| 3 | 1 | 1 | 0 | 2 | 3 | 3 | 2 | 4 | 4 | 2 | 3 | 1 | 0 26 |
| 4 | 10 | 11 | 3 | 0 | 6 | 6 | 4 | 8 | 6 | 9 | 10 | 3 | 0 76 |
| 5 | 7 | 7 | 3 | 5 | 0 | 7 | 5 | 11 | 7 | 7 | 13 | 4 | 0 76 |
| 6 | 7 | 7 | 3 | 5 | 6 | 0 | 7 | 8 | 9 | 5 | 9 | 3 | 0 69 |
| 7 | 9 | 10 | 2 | 5 | 6 | 7 | 0 | 6 | 8 | 10 | 7 | 2 | 0 72 |
| 8 | 13 | 13 | 4 | 11 | 13 | 10 | 8 | 0 | 11 | 13 | 9 | 1 | 0106 |
| 9 | 9 | 9 | 4 | 6 | 7 | 10 | 6 | 8 | 0 | 7 | 10 | 4 | 0 80 |
| 10 | 5 | 4 | 2 | 6 | 6 | 4 | 7 | 8 | 8 | 0 | 7 | 2 | 0 59 |
| 11 | 10 | 14 | 3 | 13 | 13 | 9 | 8 | 7 | 12 | 9 | 0 | 1 | 0 99 |
| 12 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 7 4 |
| | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | |
| | 78 | 81 | 31 | 75 | 78 | 75 | 67 | 88 | 86 | 78 | 94 | 28 | |

Figure 6. - Output of fusulinids from CLADN1.

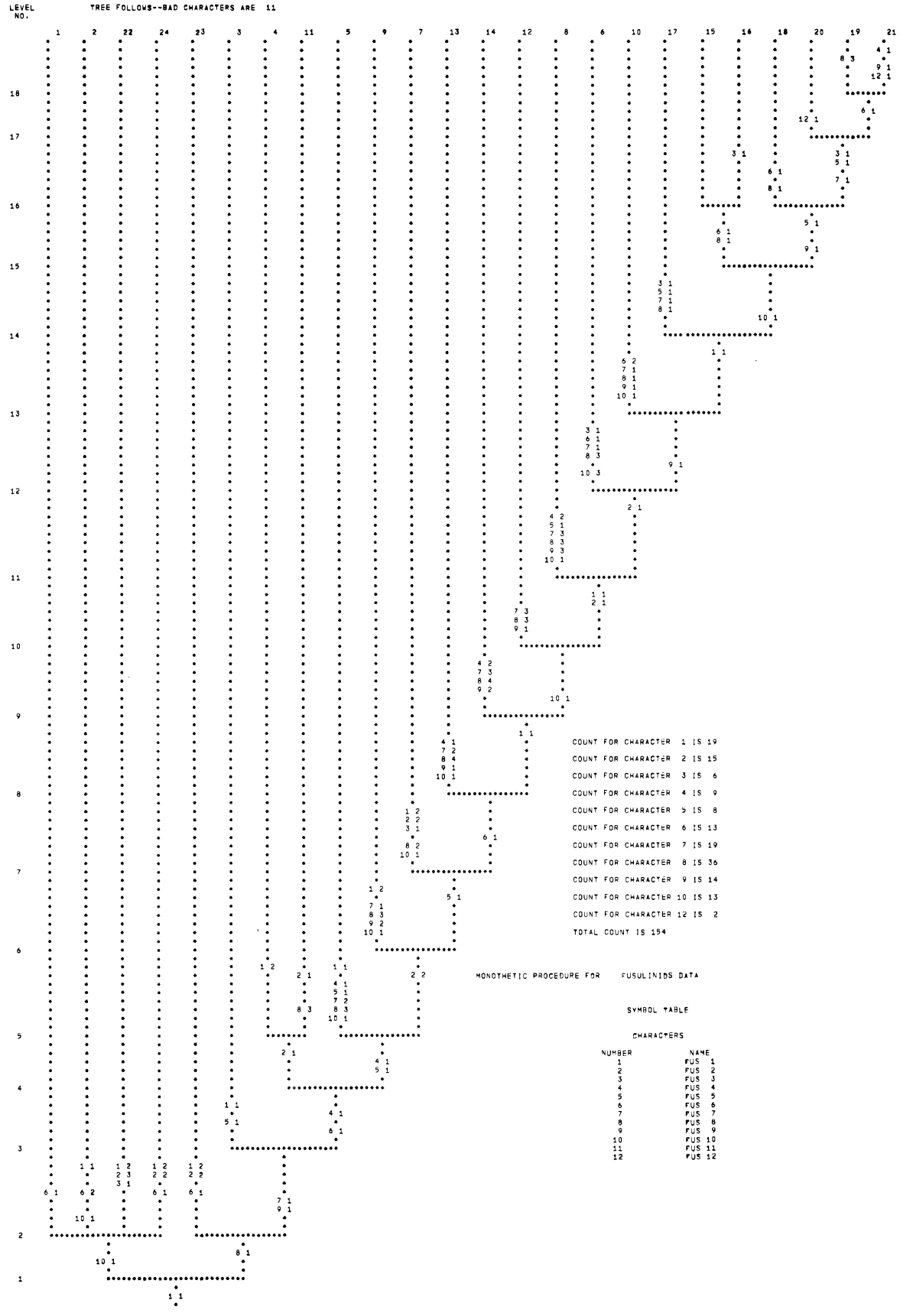
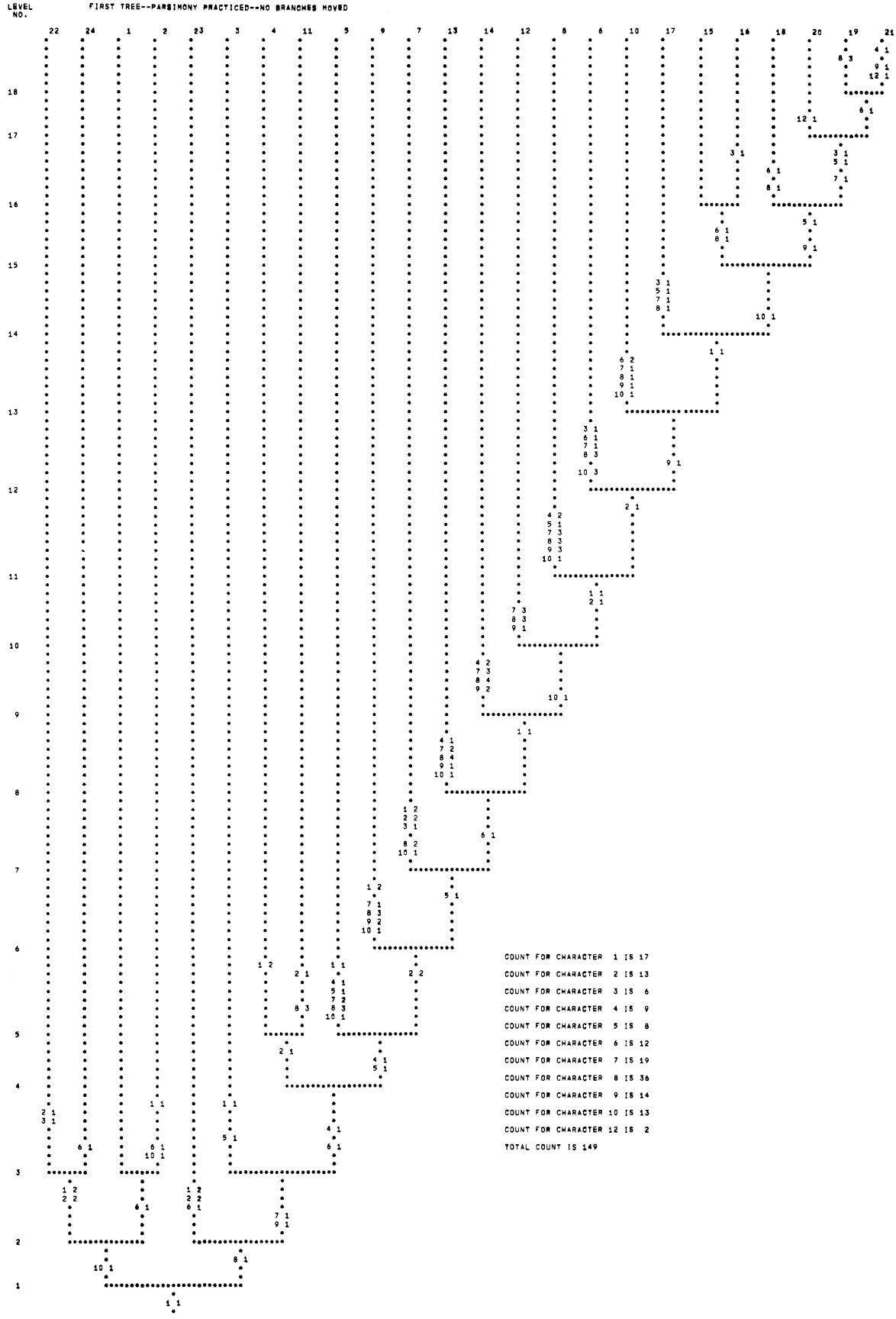



Figure 7. - Output of fusulinids from CLADN2.

```
NAME OF THIS SET OF DATA=   FUSULINIDS DATA

                        SYMBOL TABLE
                         CHARACTERS

                    NUMBER          NAME
                       1            FUS  1
                       2            FUS  2
                       3            FUS  3
                       4            FUS  4
                       5            FUS  5
                       6            FUS  6
                       7            FUS  7
                       8            FUS  8
                       9            FUS  9
                      10            FUS 10
                      11            FUS 11
                      12            FUS 12
 TREE FOLLOWS--BAD CHARACTERS ARE  11

 INITIAL TREE--NO PARSIMONY
 COUNT FOR CHARACTER  1 IS 19

 COUNT FOR CHARACTER  2 IS 15

 COUNT FOR CHARACTER  3 IS  6

 COUNT FOR CHARACTER  4 IS  9

 COUNT FOR CHARACTER  5 IS  8

 COUNT FOR CHARACTER  6 IS 13

 COUNT FOR CHARACTER  7 IS 19

 COUNT FOR CHARACTER  8 IS 36

 COUNT FOR CHARACTER  9 IS 14

 COUNT FOR CHARACTER 10 IS 13

 COUNT FOR CHARACTER 12 IS  2

 TOTAL COUNT IS 154

 AT LEVEL   2 THERE ARE   2 EQUALLY PARSIMONIOUS CASES FOR STAGE  2

 GROUP  1 IS *     1*

 GROUP  2 IS *     2*

 GROUP  3 IS *    22*    24*
              1    2
              2    3
```

(Note: This tree is the same as that in Figure 7.)

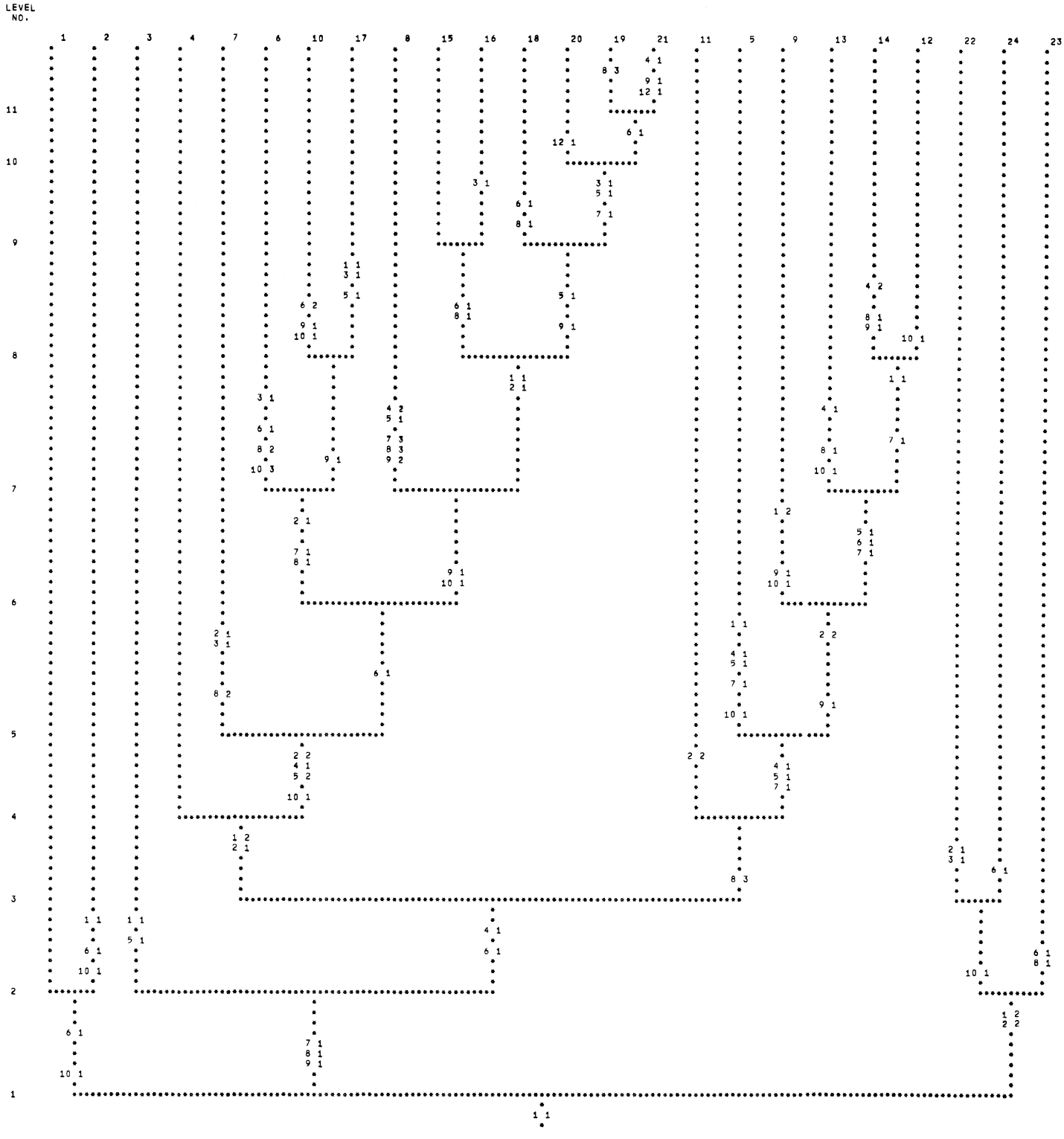
Figure 8. - Output of fusulinids from CLADN3.


LEVEL NO.
FIRST TREE--PARSIMONY PRACTICED--NO BRANCHES MOVED
22 24 1 2 23 3 4 11 5 9 7 13 14 12 8 6 10 17 15 16 18 20 19 21
COUNT FOR CHARACTER 1 IS 17
COUNT FOR CHARACTER 2 IS 13
COUNT FOR CHARACTER 3 IS 6
COUNT FOR CHARACTER 4 IS 9
COUNT FOR CHARACTER 5 IS 8
COUNT FOR CHARACTER 6 IS 12
COUNT FOR CHARACTER 7 IS 19
COUNT FOR CHARACTER 8 IS 36
COUNT FOR CHARACTER 9 IS 14
COUNT FOR CHARACTER 10 IS 13
COUNT FOR CHARACTER 12 IS 2
TOTAL COUNT IS 149

B

```
 1 LEVEL MOVE---PASS  1
AT LEVEL  17 THERE ARE   2 EQUALLY PARSIMONIOUS CASES FOR STAGE  1
GROUP  1 IS *    20*

GROUP  2 IS *    19*

GROUP  3 IS *    21*
             1    3
             2    3
AT LEVEL  14 THERE ARE   3 EQUALLY PARSIMONIOUS CASES FOR STAGE  1
GROUP  1 IS *    17*

GROUP  2 IS *    15*    16*

GROUP  3 IS *    18*    20*    19*    21*
             1    2
             1    3
             2    3
AT LEVEL  11 THERE ARE   2 EQUALLY PARSIMONIOUS CASES FOR STAGE  1
GROUP  1 IS *     8*

GROUP  2 IS *    15*    16*    18*    20*    19*    21*

GROUP  3 IS *     6*    10*    17*
             1    2
             1    3
AT LEVEL  10 THERE ARE   2 EQUALLY PARSIMONIOUS CASES FOR STAGE  1
GROUP  1 IS *    12*

GROUP  2 IS *     6*    10*    17*

GROUP  3 IS *     8*    15*    16*    18*    20*    19*    21*
             1    2
             2    3
AT LEVEL   4 THERE ARE   2 EQUALLY PARSIMONIOUS CASES FOR STAGE  1
GROUP  1 IS *     4*

GROUP  2 IS *    11*

GROUP  3 IS *     7*     6*    10*    17*     8*    15*    16*    18*    20*

                 19*    21*

GROUP  4 IS *     5*     9*    13*    14*    12*
             1    3
             2    4
AT LEVEL   2 THERE ARE   2 EQUALLY PARSIMONIOUS CASES FOR STAGE  2
GROUP  1 IS *     1*

GROUP  2 IS *     2*

GROUP  3 IS *    22*    24*
             1    2
             2    3
```

C

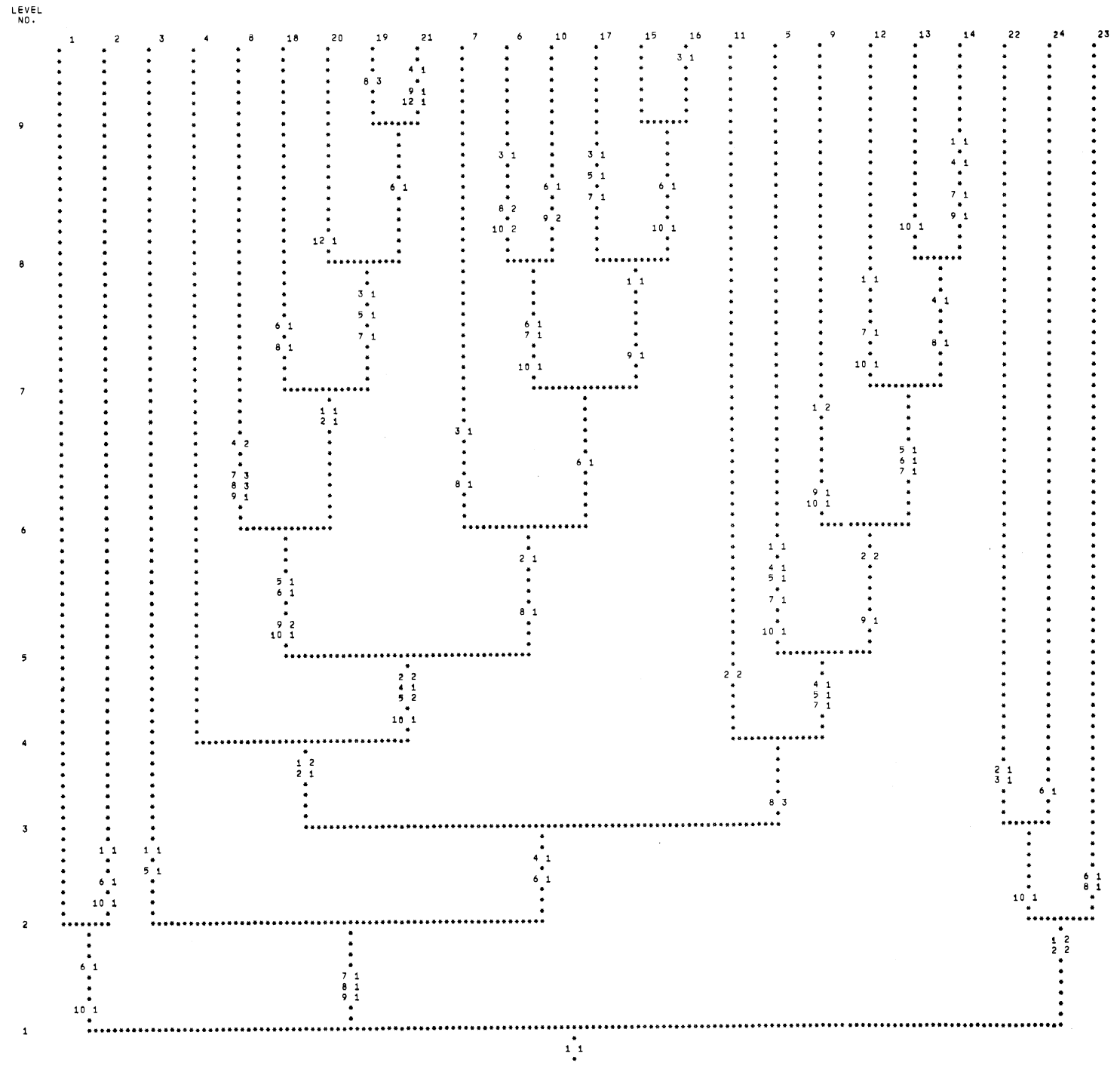



COUNT FOR CHARACTER 1 IS 13
COUNT FOR CHARACTER 2 IS 13
COUNT FOR CHARACTER 3 IS 6
COUNT FOR CHARACTER 4 IS 10
COUNT FOR CHARACTER 5 IS 10
COUNT FOR CHARACTER 6 IS 13
COUNT FOR CHARACTER 7 IS 10
COUNT FOR CHARACTER 8 IS 20
COUNT FOR CHARACTER 9 IS 11
COUNT FOR CHARACTER 10 IS 13
COUNT FOR CHARACTER 12 IS 2
TOTAL COUNT IS 121

D

```
 1 LEVEL MOVE-==PASS  2
AT LEVEL  10 THERE ARE   2 EQUALLY PARSIMONIOUS CASES FOR STAGE  1
GROUP  1 IS *    20*

GROUP  2 IS *    19*

GROUP  3 IS *    21*
             1    3
             2    3
AT LEVEL   7 THERE ARE   2 EQUALLY PARSIMONIOUS CASES FOR STAGE  1
GROUP  1 IS *     8*

GROUP  2 IS *    15*    16*

GROUP  3 IS *    18*    20*    19*    21*
            1    3
            2    3
AT LEVEL   7 THERE ARE   2 EQUALLY PARSIMONIOUS CASES FOR STAGE  1
GROUP  1 IS *    13*

GROUP  2 IS *    14*

GROUP  3 IS *    12*
            1    2
            2    3
AT LEVEL   6 THERE ARE   2 EQUALLY PARSIMONIOUS CASES FOR STAGE  1
GROUP  1 IS *    17*

GROUP  2 IS *     6*    10*

GROUP  3 IS *    15*    16*

GROUP  4 IS *     8*    18*    20*    19*    21*
            1    3
            2    3
AT LEVEL   3 THERE ARE   2 EQUALLY PARSIMONIOUS CASES FOR STAGE  1
GROUP  1 IS *     4*

GROUP  2 IS *     8*    18*    20*    19*    21*     7*     6*    10*    17*

                 15*    16*

GROUP  3 IS *    11*

GROUP  4 IS *     5*     9*    12*    13*    14*
            1    2
            3    4
AT LEVEL   2 THERE ARE   2 EQUALLY PARSIMONIOUS CASES FOR STAGE  1
GROUP  1 IS *    22*

GROUP  2 IS *    24*

GROUP  3 IS *    23*
            1    2
            2    3
```

E


LEVEL NO.
1 2 3 4 8 18 20 19 21 7 6 10 17 15 16 11 5 9 12 13 14 22 24 23
9
8
7
6
5
4
3
2
1
COUNT FOR CHARACTER 1 IS 14
COUNT FOR CHARACTER 2 IS 12
COUNT FOR CHARACTER 3 IS 6
COUNT FOR CHARACTER 4 IS 9
COUNT FOR CHARACTER 5 IS 9
COUNT FOR CHARACTER 6 IS 13
COUNT FOR CHARACTER 7 IS 12
COUNT FOR CHARACTER 8 IS 17
COUNT FOR CHARACTER 9 IS 11
COUNT FOR CHARACTER 10 IS 13
COUNT FOR CHARACTER 12 IS 2
TOTAL COUNT IS 118



F

```
2 LEVEL MOVE-=PASS  1
AT LEVEL    7 THERE ARE    2 EQUALLY PARSIMONIOUS CASES FOR STAGE  1

GROUP  1 IS *    18*

GROUP  2 IS *    20*

GROUP  3 IS *    19*

GROUP  4 IS *    21*
             2    4
             3    4
AT LEVEL    7 THERE ARE    2 EQUALLY PARSIMONIOUS CASES FOR STAGE  2

GROUP  1 IS *     6*

GROUP  2 IS *    10*

GROUP  3 IS *    17*

GROUP  4 IS *    15*    16*
             1    2
             2    4
AT LEVEL    7 THERE ARE    2 EQUALLY PARSIMONIOUS CASES FOR STAGE  3

GROUP  1 IS *    17*

GROUP  2 IS *    15*    16*

GROUP  3 IS *     6*    10*
             1    2
             2    3
AT LEVEL    7 THERE ARE    2 EQUALLY PARSIMONIOUS CASES FOR STAGE  1

GROUP  1 IS *    12*

GROUP  2 IS *    13*

GROUP  3 IS *    14*
             1    3
             2    3
AT LEVEL    6 THERE ARE    2 EQUALLY PARSIMONIOUS CASES FOR STAGE  1

GROUP  1 IS *     7*

GROUP  2 IS *     6*

GROUP  3 IS *    10*

GROUP  4 IS *    17*

GROUP  5 IS *    15*    16*
             2    3
             3    5
AT LEVEL    6 THERE ARE    2 EQUALLY PARSIMONIOUS CASES FOR STAGE  2

GROUP  1 IS *     7*

GROUP  2 IS *    17*

GROUP  3 IS *    15*    16*

GROUP  4 IS *     6*    10*
             2    3
             3    4
AT LEVEL    6 THERE ARE    2 EQUALLY PARSIMONIOUS CASES FOR STAGE  1

GROUP  1 IS *     9*

GROUP  2 IS *    12*

GROUP  3 IS *    13*

GROUP  4 IS *    14*
             2    4
             3    4
AT LEVEL    3 THERE ARE    2 EQUALLY PARSIMONIOUS CASES FOR STAGE  3

GROUP  1 IS *     4*

GROUP  2 IS *    11*

GROUP  3 IS *     8*    18*    20*    19*    21*     7*     6*    10*    17*    15*    16*

GROUP  4 IS *     5*     9*    12*    13*    14*
             1    3
             2    4
AT LEVEL    2 THERE ARE    2 EQUALLY PARSIMONIOUS CASES FOR STAGE  1

GROUP  1 IS *     3*

GROUP  2 IS *     4*

GROUP  3 IS *     8*    18*    20*    19*    21*     7*     6*    10*    17*    15*    16*

GROUP  4 IS *    11*

GROUP  5 IS *     5*     9*    12*    13*    14*
             2    3
             4    5
AT LEVEL    2 THERE ARE    2 EQUALLY PARSIMONIOUS CASES FOR STAGE  1

GROUP  1 IS *    22*

GROUP  2 IS *    24*

GROUP  3 IS *    23*
             1    2
             2    3
```

G

```
TREE FOLLOWS WITH ALL CHARACTERS PUT ON
```

(Note: Tree is omitted here. Character count is given below.)

```
COUNT FOR CHARACTER  1 IS 14
COUNT FOR CHARACTER  2 IS 12
COUNT FOR CHARACTER  3 IS  6
COUNT FOR CHARACTER  4 IS  9
COUNT FOR CHARACTER  5 IS  9
COUNT FOR CHARACTER  6 IS 13
COUNT FOR CHARACTER  7 IS 12
COUNT FOR CHARACTER  8 IS 17
COUNT FOR CHARACTER  9 IS 11
COUNT FOR CHARACTER 10 IS 13
COUNT FOR CHARACTER 11 IS 25
COUNT FOR CHARACTER 12 IS  2
TOTAL COUNT IS 143
```

(Note: This is the numerical representation of the final tree.)

```
 1 2     2 1     3 2     4 4     8 6     18 7    20 8    19 9
21 5     7 6     6 8    10 7    17 8    15 9    16 3    11 4
 5 5     9 6    12 7    13 8    14 1    22 3    24 2    23
```

H

KANSAS GEOLOGICAL SURVEY COMPUTER PROGRAM
THE UNIVERSITY OF KANSAS, LAWRENCE

PROGRAM ABSTRACT

Title (If subroutine state in title):

FORTRAN IV PROGRAM FOR ESTIMATION OF CLADISTIC RELATIONSHIPS USING THE IBM 7040

Computer: IBM 7040 Date: August, 1966

Programming language: FORTRAN IV

Author, organization: Ronald L. Bartcher
Department of Entomology, University of Kansas, Lawrence

Direct inquiries to: Author, or

Name: D. F. Merriam Address: Kansas Geological Survey
University of Kansas, Lawrence

Purpose/description: This program is designed to estimate cladistic relationships among taxa, i.e., the evolutionary branching sequence among taxonomic units without regard to phenetic similarities among them or to an absolute time scale.

Mathematical method: The reconstruction of cladistic relationships proceeds on the hypothesis that the minimum number of evolutionary steps yields the correct cladogram.

Restrictions, range:

Storage requirements:

Equipment specifications: Memory 20K 40K 60K 16 K X

Automatic divide: Yes X No Indirect addressing Yes X No

Other special features required Four utility tape units

Additional remarks (include at author's discretion: fixed/float, relocatability; optional: running time, approximate number of times run successfully, programming hours) The program uses fixed-point arithmetic and has been tested successfully twenty-five times. Running time depends primarily upon the number of OTU's. Nine characters and 6 OTU's required 1 minute to run; 12 characters and 24 OTU's required 12 minutes to run.

Computer Contribution

1. Mathematical simulation of marine sedimentation with IBM 7090/7094 computers, by J. W. Harbaugh, 1966. . . . . $1.00
2. A generalized two-dimensional regression procedure, by J. R. Dempsey, 1966 . . . . . $0.50
3. FORTRAN IV and MAP program for computation and plotting of trend surfaces for degress 1 through 6, by Mont O'Leary, R. H. Lippert, and O. T. Spitz, 1966 . . . . . $0.75
4. FORTRAN II program for multivariate discriminant analysis using an IBM 1620 computer, by J. C. Davis and R. J. Sampson, 1966 . . . . . $0.50
5. FORTRAN IV program using double Fourier series for surface fitting of irregularly spaced data, by William R. James, 1966 . . . . . $0.75
6. FORTRAN IV program for estimation of cladistic relationships using the IBM 7040, by Ronald L. Bartcher, 1966. . . . . $1.00

Special Distribution Publication

3. BALGOL program for trend-surface mapping using an IBM 7090 computer, by J. W. Harbaugh, 1963 . . $0.50
4. FORTRAN II program for coefficient of association (Match-Coeff) using an IBM 1620 computer, by R. L. Kaesler, F. W. Preston, and D. I. Good, 1963 . . . . . $0.25
9. BALGOL programs for calculation of distance coefficients and correlation coefficients using an IBM 7090 computer, by J. W. Harbaugh, 1964 . . . . . $0.75
11. Trend-surface analysis of regional and residual components of geologic structure in Kansas, by D. F. Merriam and J. W. Harbaugh, 1964. . . . . $0.75
12. FORTRAN and FAP program for calculating and plotting time-trend curves using an IBM 7090 or 7094/1401 computer system, by W. T. Fox, 1964 . . . . . $0.75
13. FORTRAN program for factor and vector analysis of geologic data using an IBM 7090 or 7094/1401 computer system, by Vincent Manson and John Imbrie, 1964. . . . . $1.00
14. FORTRAN II trend-surface program for the IBM 1620, by D. I. Good, 1964 . . . . . $1.00
15. Application of factor analysis to petrologic variations of Americus Limestone (lower Permian), Kansas and Oklahoma, by J. W. Harbaugh and Ferruh Demirmen, 1964 . . . . . $1.00
23. ALGOL program for cross-association of nonnumeric sequences using a medium-size computer, by M. J. Sackin, P. H. A. Sneath, and D. F. Merriam, 1965 . . . . . $0.75
24. BALGOL program and geologic application for single and double Fourier series using IBM 7090/7094 computers, by F. W. Preston and J. W. Harbaugh, 1965 . . . . . $1.00
25. Final report of the Kansas Geological Society Basement Rock Committee and list of Kansas wells drilled into Precambrian rocks, by V. B. Cole, D. F. Merriam, and W. W. Hambleton, 1965 . . . . . $0.75
26. FORTRAN II trend-surface program with unrestricted input for the IBM 1620 computer, by R. J. Sampson and J. C. Davis, 1966 . . . . . $0.50
27. Application of factor analysis to a facies study of the Leavenworth Limestone (Pennsylvanian-Virgilian) of Kansas and environs, by D. F. Toomey, 1966 . . . . . $0.75
28. FORTRAN II program for standard-size analysis of unconsolidated sediments, by J. W. Pierce and D. I. Good, 1966 . . . . . $0.75

Reprints (available for limited time)

Mathematical conversion of section, township, and range notation to Cartesian Coordinates, by D. I. Good (Kansas Geological Survey Bulletin 170, pt. 3, 1964) . . . . . $0.50

A computer method for four-variable trend analysis illustrated by a study of oil-gravity variations in southeastern Kansas, by J. W. Harbaugh (Kansas Geological Survey Bulletin 171, 1964) . . . . $1.00

Finding the ideal cyclothem, by W. C. Pearn (reprinted from Symposium on cyclic sedimentation, D. F. Merriam, editor, Kansas Geological Survey Bulletin 169, v. 2, 1964). . . . . no charge

Fourier series characterization of cyclic sediments for stratigraphic correlation, by F. W. Preston and J. H. Henderson (reprinted from Symposium on cyclic sedimentation, D. F. Merriam, editor, Kansas Geological Survey Bulletin 169, v. 2, 1964) . . . . . no charge

Geology and the computer, by D. F. Merriam (reprinted from New Scientist, v. 26, no. 444, 1965) . . . no charge

Quantitative comparison of contour maps, by D. F. Merriam and P. H. A. Sneath (reprinted from Journal of Geophysical Research, v. 71, no. 4, 1966) . . . . . no charge

Trend-surface analysis of stratigraphic thickness data from some Namurian rocks east of Sterling, Scotland, by W. A. Read and D. F. Merriam (reprinted from Scottish Journal of Geology, v. 2, pt. 1, 1966) . no charge

Generation of orthogonal polynomials for trend surfacing with a digital computer, by O. T. Spitz (reprinted from Computers and operations research in mineral industries, Pennsylvania State University, 6th Annual Symposium, 1966) . . . . . no charge

The use of statistical communication theory for characterization of porous media, by F. W. Preston, D. W. Green, and W. D. Aldenderfer (reprinted from Computers and operations research in mineral industries, Pennsylvania State University, 6th Annual Symposium, 1966) . . . . . no charge

Geologic model studies using trend-surface analysis, by D. F. Merriam and R. H. Lippert (reprinted from Journal of Geology, v. 74, no. 5, 1966) . . . . . no charge

Geologic use of the computer, by D. F. Merriam (reprinted from Wyoming Geological Association, 20th Field Conference, 1966) . . . . . no charge