1975

Corrections to Some Errors in a Version of TSP: (Time Series Processor)

Robert E. Lynch
Purdue University, rel@cs.purdue.edu

Report Number:
75-153
CORRECTIONS TO SOME ERRORS IN A VERSION OF TSP
(TIME SERIES PROCESSOR)

Robert E. Lynch
Department of Computer Science
Purdue University
Lafayette, Indiana 47907

CSD TR 153
CORRECTIONS TO SOME SEVERE ERRORS IN A VERSION OF TSP (Time Series Processor)

Robert E. Lynch
Department of Computer Sciences
Purdue University

1. Introduction. TSP (Time Series Processor) is a problem-oriented computer system designed to carry out automatically the computations which occur routinely in econometric research. It has some of the same procedures as SPSS and it has some which SPSS does not have. The simplicity of its language makes TSP suitable for users who have only a casual acquaintance with the use of a computer, for example, the TSP statement PLOTS$ generates informative line printer graphical displays.

Several faculty members at Purdue wanted TSP available for their research activities as well as for student use in courses. A version of TSP, written in IBM360/370 Fortran, was obtained and converted to CDC6500 Fortran.

During the conversion, results from test cases showed that there were some severe errors in the IBM version. Reasonable looking output was produced which is incorrect. No TSP error message was printed. We do not see how a user could tell that these results were incorrect unless he had correct results to compare with or unless he made a detailed examination of the output together with some hand calculation.

The purpose of this report is to bring these errors, and their corrections, to the attention of installations which support TSP.

2. Errors In TSP. Errors occur in the IBM Fortran version of the TSP subroutines PASS1, LOAD (three errors), GRAPH, and CAPITL. These errors involve the way that the TSP processor handles blank common.
If the IBM Fortran compiler handles double precision assignment statements in a way similar to that of the CDC6500 Fortran compiler, then there is an error in the TSP inner produce routine, INPROD.

Section 4 contains listings of the CDC6500 version of PASSL, LOAD, GRAPH, CAPITL, and INPROD. The locations of the errors in the IBM version, and their corrections, are indicated.

Parts of a user's TSP program statements and intermediate results are stored in blank common; blank common is also used for scratch space. Errors include the omission of an appropriate test to see how much scratch space is available as well as incorrect tests.

The errors in the test cases mentioned in Section 1 occurred because the processor used scratch space which overlapped the other part of blank common.

If [1] is used as the TSP manual for this IBM Fortran version of TSP, then there is a misprint on page 3 of [1]. In the display of the size limitation, NOBS*VARS should be replaced with $301 + NOBS*(2*VARS + 1) + 2*VARS$.

Remark: The TSP manual [1] does not mention the useful TSP feature which allows a comment to be inserted into the first statement, $\$NAME$ program name$, of a TSP program. The processor stores the name immediately following NAME and everything else up to the terminator, $\$, is ignored. Consequently, instead of using something like $\$NAME CASE1$\$, the user may give a more detailed program identification, for example:

\$\$NAME CASE1, GROSS NATIONAL PRODUCT MODELS OF I.M. JONES. JUNE 19, 1984.\$

MODEL 1 GNP = CONSTANT + A*EXP(TIME)
MODEL 2 GNP = CONSTANT + A*EXP(TIME) + B*EXP(2*TIME) $

Any name in the identification, such as CASE1, GROSS, etc., may be used as a variable name in the program. The only limitation is a maximum of 8 characters
for a name; the use of SEPTEMBER would result in the generation of the TSP error message "name has too many characters."

3. Some differences between the IBM and CDC Fortran versions. In order to locate and correct the errors, detailed knowledge was required about the workings and functions of about half the 119 TSP subroutine which we had. As the purposes of various subroutines became clear, numerous comments were inserted. Consequently, the CDC6500 version is better documented than the IBM Fortran version, which is almost devoid of comments.

A number of inefficiencies were eliminated. Most of the linear searches of lists were replaced by more efficient search techniques.

In the CDC6500 version, a comment can be inserted at any place that a blank or comma separator is required or can occur. /* in columns 1 and 2 of a card denote that columns 3 to 80 contain a comment.

The user can use longer than 8 character names in order to help document his TSP program, for example

GENR NEWCASHBALANCE = OLD CASHBALANCE*(1 + (INTEREST RATE/100)*(MONTHS/12))$

However, the processor only uses the first 8 characters internally and it prints a record of the more-than-8-character names so the user can check for uniqueness.

The input routine, INPT, was completely rewritten. The new version is much more efficient than the older one. Free format data input takes only about 30% longer than formatted data input. The new version examines each non-blank, non-comma character about twice instead of the up to 20 times of the old version.

The user has more flexibility in the choice of storage requirements. He may choose blank common sizes of 3K, 5K, 10K, 15K, 20K, 25K. The 3K size is suitable for student jobs and the 25K size is capable of doing TSP programs which require the 30K IBM Fortran version. At the end of the execution output
of each TSP program, the amount of blank common which was used by the program is printed. With this information, the user can, perhaps, decrease the blank common size request on subsequent runs in order to decrease turnaround time and cost.

The most significant improvement in efficiency involves the number of words (NWORD) used to store a TSP keyword or user defined variable name (this improvement is directly transferable to the IBM Fortran version). The IBM version uses NWORD = 2 and two 8-byte words are used for keywords and names; floating point numerical values are stored in single 8-byte words. When a name is moved, two words must be moved and when a comparison or search of a list is done, pairs of comparisons are made (there is a lot of moving and comparisons of names during the execution of a TSP program). The CDC version uses NWORD = 1 and a keyword or name is stored in a single 60-bit word. This change to NWORD = 1 accomplishes the following:

1. Execution time for the movement and comparison of names is decreased by more than a factor of 2.
2. Numerous calls of short subroutines and functions are replaced by single assignment statements or by single IF statements.
3. Memory requirement is reduced (on the CDC6500).

Remark 1: If the change to NWORD = 1 were made in the IBM version, then the increased efficiencies of 1 and 2 are achieved by declaring names to be REAL DOUBLE PRECISION (or whatever the IBM Fortran declaration is). Accuracy is improved by using REAL DOUBLE PRECISION declarations for numerical values so that 16-bytes are used for floating point values; this improvement is made at the sacrifice of memory space. It is well-known that one must use as much precision as economically feasible in matrix computations--especially in least
squares computations. Moreover, in some cases, users might not know the importance of the choice of a basis for least squares computations and, perhaps, try to do a regression with a polynomial approximator of the form \( a_0 + a_1 x + \ldots + a_n x^n \); use of low precision numerical values might result in meaningless results. Thus, the change to NWORD = 1 should also be accompanied by REAL DOUBLE PRECISION declaration for just about all the variables in the TSP IBM Fortran program.

Remark 2: The CDC version uses NWORD = 1 and all changes from the IBM version are noted with comments. To change the IBM version to NWORD = 1, one can make all the appropriate changes indicated on the complete listing of the CDC version plus inserting the required REAL DOUBLE PRECISION declarations (there are a lot of these).

4. Listing of PASS1, LOAD, GRAPH, CAPITL, and INPROD. Listing of the CDC versions of the subroutines in which we found errors follows. On these listings, changes from the IBM Fortran version to CDC Fortran version are bracketed by a pair of comments: CIBMTOCDC CCDCTOIBM. Locations of errors in the IBM code and their corrections are bracketed with pairs of lines of dollar signs; notes about efficiencies are also bracketed. The corrections to the errors are CDC independent.

Since we have not seen machine language code generated by the IBM Fortran compiler, we are not sure if there is an error in the inner product routine. The original IBM program contains

\[
\text{DOUBLE PRECISION XPROD}
\]

\[
J = 1
\]

\[
XPROD = 0.
\]

\[
\text{DO 100 } I = \text{NN}, \text{JSA}
\]

\[
XPROD = XPROD + A(I) * B(J)
\]

\[
100 J = J + \text{JSB}
\]
The CDC compiler generates code which computes the single precision product
\[ A(i) \times B(j), \]
then makes it double precision with zero lower half, and finally
adds this to the double precision \( X_{\text{PROD}} \).

One way to carry out this calculation correctly on the CDC6500 is to
introduce two more double precision variables, \( \text{TEMPA} \), \( \text{TEMPS} \), and to replace the
\DO loop with

\[
\begin{align*}
\text{DO } 100 \text{ I = NN, JSA} \\
\text{TEMPA = A(I)} \\
\text{TEMPS = B(J)} \\
\text{XPROD = XPROD + TEMPA*TEMPS} \\
100 \text{ J = J + JSB}
\end{align*}
\]

Remark: Some of the DATA statement on the listings use the CDC character
string specification R, for example DATA NAMES/ BRNOPRINT, .../. This causes
the BCD representation of the character string to be stored right justified
with zero left fill; these can be used as variables declared INTEGER in arith-
metical and logical expressions.
CIBMTecc

SUBROUTINE PASS1
C
C THIS PROGRAM SUPERVISES READING AND PARSING THE INPUT
C
C REPLACED JX(400) (NHORD = 1), ADDED LINTYP COMMON
*     TYPE(200), JX(200), C0DE(600), IX(200), ITYPE(200)
INTEGER TYPE
DATA LIT'T,1200 I
C
DIMENSION X(1)
EQUVALENCE (X(1),JX(1))
C
COMMON /TSPC0M/
*     MEMSIZ, NOB, NSPARG, NHORD, LENGTH,
*     NTYPE, IFDBUG, IFTITL, NCHAR, NSUP,
*     MEMST, NOREG, IFPLAT, IFFAST, NPAGE,
*     NUMLIN, IREPL, PROFF, SKIP(11), JPHAS,
*     LIMARG, LINE, NJARG, NARG, NAME,
*     JARG(4)
LENGTH OF JARG SET IN MAIN OVERLAY
C
C DELETED NAME2 BETWEEN NAME AND JARG IN /TSPC0M/ (NHORD = 1)
C
LOGICAL IFDBUG, IFTITL, IFPLAT, IFFAST,
*     IREPL, PROFF
C
C NEW COMMON BLOCK ADDED
COMMON /PENCOM/ INMSZ(?), USRNAM, PSSVL(20)
INTEGER USRNAM
EQUIVALENCE
*     (INMSZ(1), INMSZ(2), USRNAM, PSSVL(20))
C
DIMENSION IPSSVL(20)
EQUIVALENCE (IPSSVL(1), PSSVL(1))
C
PSSVL AND IPSSVL USED TO PASS VARIABLES BETWEEN SUBROUTINES
IN DIFFERENT OVERLAYS
C
IPSSVL(1) IS USED TO KEEP RECORD OF USE OF COMMON IN LOAD
AND IN GENR
C
IPSSVL(2) PASSES COMPUTED GO TO INDEX FROM EXEC TO MATRIX
C
IPSSVL(3) PASSES COMPUTED GO TO INDEX FROM EXEC TO OVERLAYS
C
IPSSVL(4) IS USED TO KEEP RECORD OF USE OF JARG-COMMON
C
IPSSVL(5) IS USED TO KEEP RECORD OF USE OF DOT-COMMON
C
IPSSVL(6) IS USED TO KEEP RECORD OF USE OF BUFFER-COMMON
C
IPSSVL(7) IS USED TO KEEP RECORD OF USE OF JARG-COMMON
C
IPSSVL(8) IS USED TO KEEP RECORD OF USE OF COMMON IN LOAD
AND IN GENR
C
INTEGER D0USE, BUFUSE, SNPUSE, OBFUSE
C
CHANGED DIMENSION OF NAMES TO 5 FROM 20
C
CHANGED DIMENSION OF NAMES TO 4 FROM 20
C
DIMENSION N4AMES(2), JNAME(5), NNAMES(4), NAMES(5), NAME(2)
C
COMMON /LPAR/
LOGICAL LAGGED, IFFPAR
NEXT ARE OPERATOR CODES FOR TSP FUNCTIONS

11 EXP, 12 ABS, 13 LOG AND ALOG

CCDCTOIBM
DATA JCODE /0, 11, 12, 13, 14/

C

DATA G / 1RG /

C

DATA NAMES /
* 5REXP, 6RABS, 7RALOG, 8RLOG /

C

DATA NAMES /
* 9SREND, 8RGENR, 9NAME, 9RBCD, 9REBCDIC /

C

CCDCTOIBM
LAGGED = .FALSE.
IRPAR = .FALSE.

C

CIBMT0DCDC

C

REPLACED 37 WITH 10 TO PRINT THE WORD LINE ON TSP PROGRAM OUTPUT

C

1 CALL OUTDET (10, 1, LINE+1 )

C

READ NEXT TSP STATEMENT

C

CCDCTOIBM

CALL INPT(JX, N, TYPE)

C

CIBMT0DCDC

C

NAMES, NUMBERS, OPERATORS ON TSP STATEMENT ARE NOW STORED
IN SUCCESSIVE WORDS IN ARRAY JX(I), I=1,N IN THE ORDER THAT
THEY APPEAR IN THE TSP STATEMENT
STATEMENT KEYWORD IS IN JX(1)
TYPE OF ELEMENT IN JX(I) IS GIVEN BY VALUE OF TYPE(I)
1 NUMERICAL
2 OPERATOR
3 NAME
4 DOTNAME

C

*********** MAJOR ERROR IN IBM VERSION. INPT CAN DESTROY DATA IN
BLANK COMMON SUCH AS PROGRAM STATEMENTS, DATA, VALUES
GENERATED WITH GENR
A TEST WAS ADDED AT RETURN OF CALL OF INPUT
IF( N . LE. LIMTYP )
THEN ARRAYS ARE LARGE ENOUGH FOR INPUT

C

ELSE TOO MUCH DATA
IF( LIMTYP + N . LT. MEMSIZ )
THEN HAVE NOT DESTROYED STORED VALUES IN BLANK COMMON

C

ELSE PART OF BLANK COMMON OVER-Written
IRUIN = LIMTYP + N - MEMSIZ
WRITE (6, 10010) IRUIN

10010 FORMAT (2HO*** ERROR. DURING READ OF STATEMENTS .
* 16H DESTROYED LAST, 11O, 14H WORDS OF DATA .
* 28H IN BLANK COMMON. ABORT RUN .
* )

C

10020 CALL ABORT PAS51

10030 FORMAT (3HO*** WARNING. READ IN TOO MUCH .
* 27H DATA FOR ARRAYS IN PASS1 .
* 3OH OPERATIONS CONTINUE CORRECTLY .
* )
THE FOLLOWING USES THE COLLATING SEQUENCE OF THE CDC6500
THE 8-BCD CHARACTER WORDS IN NAMES(I) SATISFY
CHAR. = 12345678 12345678 12345678 12345678 12345678
BCD . LT. EBCDIC .LT. END . LT. GENR . LT. NAME
I = 4 5 1 2 3

DETERMINE WHETHER TSP KEYWORD IS ONE OF
BCD, EBCDIC, END, GENR, NAME

TEST FOR KEYWORD END
IF( JX(1) - NAMES(1) ) 3, 100, 4
3 TEST FOR KEYWORD EBCDIC
IF( JX(1) - NAMES(5) ) 5, 410, 10
4 TEST FOR KEYWORD GENR
IF( JX(1) - NAMES(2) ) 10, 200, 6
5 TEST FOR KEYWORD BCD
IF( JX(1) - NAMES(4) ) 10, 400, 10
6 TEST FOR KEYWORD NAME
IF( JX(1) - NAMES(3) ) 10, 300, 10
END OF REPLACEMENT

KEYWORD IS NOT ONE OF BCD, EBCDIC, END, GENR, NAME

CCDCTIBM
10 NARG = N - 1
IF (TYPE(1) ,NE, 3) CALL ERG(1,JX)
C IGTOCDC
C MOVE KEYWORD TO NAME STORAGE
C REPLACED CALL NM0V(NAME,JX(1)) WITH FOLLOWING (NAME = JX(1))
NAME = JX(1)
35 CONTINUE
CCDCTIBM
IF (NARG . EQ. 0) GO TO 50
C IGTOCDC
C REPLACED KK = NWORD + 1 WITH THE FOLLOWING (NWORD = 1)
KK = 2
C PUT ARGUMENTS OF TSP STATEMENT INTO TEMPORARY STORAGE IN JARG
CCDCTIBM
DO 40 I=2, N
CALL ARGPUT(I-1,JX(KK),TYPE(I),0)
KK = KK + 1
C IGTOCDC
C DELETED THE FOLLOWING (NWORD = 1)
C IF(TYPE(1) .EQ. 3 OR. TYPE(I) .EQ. 4) KK = KK + NWORD + 1
CCDCTIBM
40 CONTINUE
CIBMT0CDD
C
50 CONTINUE
C
FROM STATEMENT FOLLOWING 35 OR 225
C
LINE = LINE + 1
C
HAVE NAMES AND ARGUMENTS OF COMPLETE TSP STATEMENT STORED
C IN NAME/JARG(1); MOVE THEM TO BLANK COMMON
C
CCDCT0IBM
CALL LINPUT
GO TO 1
C
CIBMT0CDD
C
HAVE TSP KEYWORD ****END****
C FINISHED READING TSP PROGRAM; STORE END STATEMENT AND
C RETURN TO SUPER TO EXECUTE TSP PROGRAM
C
REPLACED CALL NM0V(NAME, JX(1)) WITH FOLLOWING (NW0RD = 1)
100 NAME = JX(1)
CCDCT0IBM
MARG=0
LINE=LINE+1
CALL LINPUT
CALL OUTPUT
CIBMT0CDD
C
NEXT IS ENTRY TO SUPER
CCDCT0IBM
CALL RETURN
CIBMT0CDD
C
NO RETURN TO THIS POINT FROM ENTRY RETURN OF SUPER
CCDCT0IBM
C
CIBMT0CDD
C
HAVE TSP KEYWORD ****GENR****
C MOVE KEYWORD TO NAME STORAGE
C
REPLACED CALL NM0V(NAME, JX(1)) WITH FOLLOWING (NW0RD = 1)
200 NAME = JX(1)
C
CONSTRACT GENR-LINE-IDENTIFIER AND STORE IN JARG(1)
CCDCT0IBM
CALL INVNT(NAME,G,LINE)
CALL ARGPUT(1,NAME,6,0)
CIBMT0CDD
C
PUT LEFT SIDE OF = INTO TEMPORARY STORAGE IN JARG
C
REPLACED JX(NW0RD + 1) WITH JX(2) (NW0RD = 1)
CALL ARGPUT(2, JX(2), TYPE(2), 0 )
CCDCT0IBM
C
JJ IS LOCATION IN IX.
C
KK IS LOCATION IN JX.
C
LL IS LOCATION IN JARG.
C
IX(1) = 9
ITYPE(1) = 2
JJ = 2
LL = 3
CIBMT0CDD
C
REPLACED KK = NW0RD+2 + 2 WITH FOLLOWING (NW0RD = 1)
C
THE USE OF KK = 4 (OR 6 IF NW0RD=2) PICKS UP FIRST ITEM ON RIGHT
SIDE OF = IN GENR STATEMENT

\begin{verbatim}
KK = 4
CCDCT01BM
C
DO 220 I=4,N
NTYPE = TYPE(1)
    IF (<LAGGED>) GO T0 530
CIBMT0C0C
C TEST FOR NAME OR NUMBER
CCDCT01BM
    IF (NTYPE .NE. 2) GO T0 205
CIBMT0C0C
C TEST FOR LEFT PAREN
CCDCT01BM
        IF (JX(KK) .EQ. 9) GO T0 500
C
C OPERATOR
C
201 MMM = JX(KK)
CIBMT0C0C
C TEST TO SEE IF MINUS (CODE 3) SHOULD BE CHANGED TO CODE 8
C 9 = LEFT PAREN, 2 = OPERATOR). SWITCH MADE IF - IS PRECEDED
C BY < CODE 3 MINUS IS INFIX, CODE 8 MINUS IS PREFIX
CCDCT01BM
        IF (MMM .EQ. 3 .AND. IX(JJ-1) .EQ. 9 .AND. ITYPE(JJ-1) .EQ. 2)
           I MMM = 8
           IX(JJ) = MMM
           ITYPE(JJ) = 2
           JJ = JJ + 1
           KK = KK + 1
           GO T0 220
CIBMT0C0C
C HAVE LEFT PAREN. SEE IF AN OPERATOR (CODE 2) PRECEDES IT.
CCDCT01BM
      500 IF (ITYPE(JJ-1) .EQ. 2) GO T0 201
CIBMT0C0C
C LEFT PAREN PRECEDED BY NAME, GET NAME AND SEE IF IT IS
C TSP FUNCTION
CCDCT01BM
    LOOKUP = LL - 1
    CALL ARGGET(LOOKUP, NAMED, JTYPE, LAG)
CIBMT0C0C
C REPLACED MATCH = IUCOMP(NAMED, NNAMES) WITH THE FOLLOWING
C DETERMINE IF NAMED(1) IS ONE OF TSP FUNCTIONS EXP, ABS, LOG, ALOG
C MATCH = 1
C IF( NAMED(1) .EQ. NNAMES(1) ) MATCH = 2
C IF( NAMED(1) .EQ. NNAMES(2) ) MATCH = 3
C IF( NAMED(1) .EQ. NNAMES(3) ) MATCH = 4
C IF( NAMED(1) .EQ. NNAMES(4) ) MATCH = 5
C END OF REPLACEMENT
CCDCT01BM
IF (MATCH .EQ. 1) GO T0 525
C
CIBMT0C0C
C HAVE ONE OF THE FUNCTIONS ABS, EXP, ALOG, LOG
C INSERT ITS CODE AND SET TYPE TO OPERATOR (2)
CCDCT01BM
C
505 LL = LOOKUP
    IX(JJ-1) = JCODE(MATCH)
    ITYPE(JJ-1) = 2
    GO T0 201
C
CIBMT0C0C
C HAVE NONE OF THE FUNCTIONS ABS, EXP, ALOG, LOG
\end{verbatim}
LEFT PAREN PRECEDED BY NON-TSP-FUNCTION NAME, HAVE LAGGED

525 LAGGED = .TRUE.
  KK = KK + 1
  GO TO 220
  $30 IF (NTYPE .NE. 1) GO TO 540

PROCESSING SIGNED INTEGER ARGUMENT OF LAGGED VARIABLE, STORED AS REAL TYPE, CONVERT IT TO INTEGER TYPE

-MM = X(KK)
  CALL ARGPUT(LOOKUP, NAMED, JTYPE, MMM)
  IFRPAR = .TRUE.
  KK = KK + 1
  GO TO 220

TURN OF LAGGED-SWITCH WHEN GET TO RIGHT PAREN

-940 IF(NTYPE .NE. 1, IFRPAR) CALL ERG(69, JX(KK))
  IF (NTYPE .NE. 2, OR, JX(KK) .NE. 1) CALL ERG(69, JX(KK))
  LAGGED = .FALSE.
  KK = KK + 1
  GO TO 220

PUT NAME OR NUMBER INTO JAVA(.).

-205 CALL ARGPUT(LL, JX(KK), NTYPE, 0)
  IF (NTYPE .NE. 1, OR, JX(JJ-1) .NE. -1, AND, (NTYPE(JJ-1) .NE. 2, OR, IX(JJ) .NE. 1)) GO TO 210
  IX(JJ) = 2
  JY(JJ) = 2
  JJ = JJ + 1

210 IX(JJ) = LL - 2
  ITYPE(JJ) = -1
  JJ = JJ + 1
  KK = KK + 1

DELETED THE FOLLOWING (NW0RD = 1)

IF (NTYPE .EQ. 1, OR, NTYPE .EQ. 4) KK = KK + NW0RD - 1

HAVE FINISHED STORING CODED GENR STATEMENT, CONVERT IT TO STRING WHICH EVAL CAN PROCESS

IX(JJ) = 1
ITYPE(JJ) = 2
JJ = JJ + 1
IX(JJ) = 14
ITYPE(JJ) = 2
CALL COMPLR(IX, ITYPE, CODE, LENGTH, CODE(501), CODE(551))

STORE STRING IN BLANK COMMON

CALL VPUT(NAME, CODE, LENGTH)
NARG = LL - 1

CIBMT0CDCC
225 CONTINUE
CCDCT0IBM
GO TO 50
C
CIBMT0CDCC
300 CONTINUE
C HAVE TSP KEYWORD *****NAME*****
C STORE THE CASE PART OF TSP STATEMENT *****NAME,CASE*****
C IN USRNAME FOR USE IN END-OF-EXECUTION OUTPUT
CCDCT0IBM
GO TO 1
C
CIBMT0CDCC
C DELETED CALL NMOV(JARG(1),X) (NWORD+1)
C
C REPLACED CALL OF ONE LINE SUBROUTINE
C UNAME CALL NMOV(...,JARG) RETURN END
C WITH THE FOLLOWING (NWORD = 1)
C USRNAME = JX(2)
CCDCT0IBM
GO TO 1
C
C SET UP CHARACTER CODE
CIBMT0CDCC
400 CONTINUE
C FOLLOWING
C 400 CALL BCD
C GO TO 1
C 410 CALL EBCDIC
C GO TO 1
C REPLACED WITH
C
C HAVE TSP KEYWORD *****BCD*****
C
JUMP = 1
GO TO 415
C
410 CONTINUE
C HAVE TSP KEYWORD *****EBCDIC*****
C
JUMP = 2
415 CONTINUE
WRITE(6,420)
420 FORMAT(44HO TSP STATEMENTS BCD AND EBCDIC DISABLED , /
* 53H CDC6500 READS BCD. IF SOURCE TSP DECK IS IN EBCDIC, /
* 47H IT SHOULD BE PREPROCESSED TO CONVERT IT TO BCD )
GO TO (1,430), JUMP
430 CONTINUE
STOP
CCDCT0IBM
END

05.23.70 J0 16 EP30 8 FEET
CIIBMTOCDC

SUBROUTINE LOAD

CCDCTOIBM

C THIS IS A STRICTLY TEMPORARY VERSION BASED ON THE OLD 1620/7094

CIIBMTOCDC

C

COMMON /TSPC0M/
* MNSZ, NOB, NSPARG, NWORD, LENGTH,
* NTYPE, IFDBUG, IFITIL, NCHAR, NSUP,
* NEMST, NREC, IFPL0T, IFFAST, NPAGE,
* NUMLIN, IFREPL, PR0FF, SKIP(11), JPHAS,
* LINARG, LINE, NJARG, NARG, NAME,
* JARG(4)

LENGTH OF JARG SET IN MAIN OVERLAY

C DELETED NAME2 BETWEEN NAME AND JARG IN /TSPC0M/ (NWORD = 1)TSPC0M

C

LOGICAL IFDBUG, IFITIL, IFPL0T, IFFAST,
* IFREPL, PR0FF

C

NEW COMMON BLOCK ADDED

COMMON /MEMCOM/ IMNSZ(7), USRNAM, PSSY(20)
INTEGER USRNAM
EQUIVALENCE * (IMNSZ(1), MMNSIZ), (IMNSZ(2), LLMARG), (IMNSZ(3), LLMD0T),
* (IMNSZ(4), LLMBUF), (IMNSZ(5), LLMSYM), (IMNSZ(6), LLMSMP),
* (IMNSZ(7), LLM0UT)

C

IMNSZ USED TO KEEP TRACK OF MEMORY USE

DIMENSION IPSSY(20)
EQUIVALENCE (IPSSY(1), PSSY(1))
IPSSY AND PSSY USED TO PASS VARIABLES BETWEEN SUBROUTINES IN DIFFERENT OVERLAYS
IPSSY(1) IS USED TO KEEP RECORD OF USE OF COMMON IN LOAD AND IN GENR
IPSSY(2) PASSES COMPUTE GO TO INDEX FROM EXEC TO MATRIX
IPSSY(3) PASSES COMPUTE GO TO INDEX FROM EXEC TO OVERLAYS
IPSSY(4) IS USED TO KEEP RECORD OF USE OF JARG-C0M0M
IPSSY(5) IS USED TO KEEP RECORD OF USE OF DJT-C0M0M
IPSSY(6) IS USED TO KEEP RECORD OF USE OF BUFFER-C0M0M
IPSSY(7) IS USED TO KEEP RECORD OF USE OF JSML-COMMON
IPSSY(8) IS USED TO KEEP RECORD OF USE OF OUTBUF-COMMON
EQUIVALENCE
* (LOUSE, IPSSY(1)), (JT00VL, IPSSY(2)), (JT0MAT, IPSSY(3))
* (JARGUSE, IPSSY(4)), (DFITUSE, IPSSY(5)), (BUFUSE, IPSSY(6)),
* (MPUSE, IPSSY(7)), (GBUSE, IPSSY(8))
INTEGER D0TUSE, BUFUSE, SMPUSE, 0BFUSE

C REMOVED /SNPC0M/
C
C ADDED JX(1) EQUIVALENCED TO X(1)

DIMENSION JX(1)
EQUIVALENCE ( JX(1), X(1) )

CCDCTOIBM

INTEGER TYPE

CIIBMTOCDC

C

COMMON
* TYPE(100), MASK(200), X(1)

INTEGER Frm, ORPAC
DATED DATA: FOR 300, LIMITYPE: 100
FRONT IS LENGTH OF ARRAYS TYPE PLUS MASK

DELETED LOGICAL IF CCP

CHANGE DIMENSION OF NAMES TOF FROM 20
DII11ENSION NAMES(10)

DATA NAMES /
* BRLOAD , BREND , BRINAM , BRSMPL , BRNPRT /
* BRFORMAT /

BEGINNING OF LOAD /*

C CIBMT0CDC

C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++:
C MAJOR ERROR IN IBM VERSION. INPT CAN DESTROY DATA IN BLANK
C COMMON
C INSERTED TEST AT RETURN FROM INPT.
C REPLACED 1 CALL XNPT(XN, TYPE)
C IF(TYPE(1), EQ. 1) G0 T0 1
C WITH THE FOLLOWING
1 CONTINUE
C
C SEARCH FOR TSP STATEMENT KEYWORD OR OPERATOR
NMLEFT = MEMSIZ - FRONT
2 CONTINUE
* CALL INPT(XN, TYPE)
IF( N .LT. NMLEFT )
C THEN HAVE ENOUGH ROOM
* GO TO 10004 LOAD
ELSE HAVE STORED OVER DATA IN BLANK COMMON
RUINED = N - NMLEFT
WRITE( 6, 10003 ) RUINED
10003 FORMAT( 50D10*** ERROR. DURING LOAD SECTION, DESTROYED LAST, LOAD 67 )
C CALL ABORT LOAD
C
10004 CONTINUE
IF( TYPE(1), EQ. 1 )
C THEN CONTINUE READING INPUT
* ELSE HAVE FOUND A NON-NUMERIC INPUT VALUE
C
C END OF CORRECTION
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++:
C INSERTED SLIGHTLY MORE EFFICIENT SEARCH
REPLACED 2 JJ = IUCOMP(XN NAMES)
C 00 T0 (10, 100, 200, 300, 400, 500, 10), JJ
C WITH THE FOLLOWING
C
C THE FOLLOWING USES THE COLLATING SEQUENCE OF THE CDC6500
C THE 8-BCD CHARACTER WORDS IN NAMES(I) SATISFY THE FOLLOWING
CHAR. = 12345678 12345678 12345678 12345678 12345678
C END . LT. LOAD . LT. MASK . LT. NOPRINT LT. SMPL LOAD
C  TEST FOR MASK
  IF( JX(1) - NAMES(3) ) 3,300, 4
C  TEST FOR LOAD
  3 IF( JX(1) - NAMES(1) ) 5,100, 10
C  TEST FOR NOPRINT
  4 IF( JX(1) - NAMES(5) ) 10,500, 6
C  TEST FOR END
  5 IF( JX(1) - NAMES(2) ) 10,200, 10
C  TEST FOR REPLACE
  6 IF( JX(1) - NAMES(4) ) 10,400, 10
C  END OF REPLACE

C  MAJOR ERROR IN IBM VERSION
  INPT CAN READ MORE VARIABLE NAMES THAN ALLOWED FOR BY THE SIZE OF ARRAY TYPE.
  IF( NVAR .LT. LIMTYP ) THEN THE ARRAYS ARE LARGE ENOUGH

C  ELSE TOO MANY VARIABLE NAMES
  WRITE( 6, 115 ) NVAR, LIMTYP
  115 FORMAT(20X0,°ERROR. LOADED , 15.
  25H VARIABLE NAMES. MAXIMUM IS , 15.
  17H PER CALL OF LOAD )
  CALL ERG(20, NVAR )

C  END OF CORRECTION

C  REPLACEMENT NLBUF=NLBUF+N+1 WITH (NWORD = 1)
  NLBUF = N + 1

C  MOVED NEXT STATEMENT UP A FEW LINES TO CUT OUT SOME MULTIPLICATION.
  NDAT = NOB*NVAR

C  MAJOR ERROR IN IBM VERSION
  NTTEST USED TO TEST THE SIZE OF AVAILABLE BLANK COMMON
  SPACE CONTAINS ERROR.

C  REPLACED NT = NLAT + NOB*NVAR WITH
```
NTEST = FRONIT + N + 1 + N08 + 2+NDATA
C BLANK COMMON USE
C ARRAYS TYPE, MASK = FRONT
C INPUT IN X = N
C TEMPORARY = N08
C WILL READ = N08+NOVAR + 1 (THE PLUS 1 IS FOR THE
C TYPE OF LAST DAtUM READ) LOAD 164
C WILL STORE = N08+NOVAR
C TOTAL IS ASSIGNED TO NTST ABOVE
C IF(NTST.LT. MEMSIZ)
C THEN HAVE ENOUGH ROOM
C ELSE WILL DESTROY DATA IN BLANK COMMON
C INEED = NTST - NDATA
C WRITE(6,118) N08, NOVAR, NDATA, NTST, MEMSIZ, INEED, NTST
I18 FORMAT(47H0*** ERROR. TRYING TO LOAD (NO. OBSERVATIONS)*, LOAD 175
* 10H(N08. VARIABLES) = ,16.1H*,16.3H = ,110.7H VALUES/
* 13K. 13THIS REQUIRES. 110. 23H WORDS IN BLANK LOMON. , LOAD 177
* 10H ONLY HAVE.110. 11H AVAILABLE. / 13K. INCREASE ,
* 53H BLANK COMMON SIZE OR REWRITE SUBROUTINE LOAD TO LOAD ,
* 12H VALUES WITH. 110. 17H WORDS INSTEAD OF . 110 / LOAD 180
* 10HABORT RUN )
C CALL ABORT LOAD 181
C (NOTE THAT A CALL OF ERR WHICH CALLS SUPER WHICH CALLS LOAD 182
C INPT IN IBM VERSION WOULD DESTROY THE DATA)
C LOAD 183
C LOAD 184
C LOAD 185
C 119 CONTINUE
C END OF CONNECTION
C $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ LOAD 189
C NOTE ON EFFICIENT USE OF STORAGE.
C THE IBM VERSION AND THIS VERSION USE NTST WORDS IN BLANK LOAD 192
C COMMON TO READ N08+NOVAR DATA. THIS AMOUNT CAN BE REDUCED LOAD 194
C BY. N08+NOVAR IF THE LOADING IS DONE IN A DO-LOOP.
C EACH ITERATION READS ONLY NOVAR DATA. THE VALUES OF THE LOAD 196
C NEXT OBSERVATION FOR EACH VARIABLE. THIS REQUIRES THE LOAD 197
C CONSTRUCTION OF A NEW ENTRY TO INPT TO READ A SPECIFIED LOAD 198
C NUMBER (NOVAR) OF VALUES. SUBROUTINE FORMAT HAS TO BE LOAD 199
C CHANGED SO THAT IT READS ONLY NOVAR VALUES.
C LOAD 200
C END OF NOTE
C $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C RECORD THE USE OF BLANK COMMON BY LOAD
C L0DUSE = MAX( L0DUSE. NTST - N08+NOVAR )
CCDC708BM
C NDT = 0 LOAD 203
C NPOINT = NLDAT LOAD 204
CIBMT0CDC
C C REPLACED NWORD+1, NWORD+2 WITH 2,3 (NWORDD = 1)
C JARG( 2 ) = 3 LOAD 212
C JARG( 3 ) = 0 LOAD 213
C LOAD 214
C 120 CALL XNPT( X(NPOINT), N, X(NPOINT+1) ) LOAD 215
C XNPT IS AN ENTRY TO NEW VERSION OF INPT LOAD 216
C LOAD 217
C C REPLACED
C IF(IFCNP( X(NPOINT), NAMES(13)) CALL FORMAT( X(NPOINT), N, NOVAR) LOAD 218
C LOAD 219
C C TEST FOR KEYWORD ***FORMAT***
C IF( JX(NPOINT) .EQ. NAMES(6) ) LOAD 220
C * CALL FORMAT( X(NPOINT), N, NBVAR )
CCDC708BM LOAD 221
C LOAD 222
C LOAD 223
C LOAD 224
C LOAD 225
```
IF (NOVAR .GT. 1) GO TO 121     
NLBUF = NLDAT
CIBMTOCDC
C
C REPLACED NWORD + 1 WITH 2 (NWORD = 1) 
   L = 2
CCDCT0IBM
GO TO 140
CIBMTOCDC
121 CONTINUE
C
C REMOVED FOLLOWING FROM INSIDE DO-130-LOOP     
   K = NLBUF + NOB - 1
C
DO 130 I = 1, NOVAR
C
C REPLACED I+NWORD + 1 WITH I + 1 (NWORD = 1) 
   L = I + 1
CCDCT0IBM
   NP0INT = NLDAT + I - 1
   DO 125 J=NLPBUF,K     
      X(J) = X(NP0INT)
   125 NP0INT = NP0INT + NOVAR
CIBMTOCDC
   LOAD 2
C
C REPLACED CALL NM0V(JARG(1),X(L)) (NWORD = 1) 
      JARG(1) = JX(L)
CCDCT0IBM
      130 CALL TSPUT(JARG(1),X(NLBUF))
      135 IF(N. NE. NDRTA) CALL ERGCiO.N)
GO Te 1
CIBMTOCDC
C REPLACED CALL NM0V(JARG(1),X(L)) (NWORD = 1) 
      JARG(1) = JXCL)
CCDCT0IBM
      CALL TSPUT(JARG(1),X(NLBUF))
      GO Te 135
C
CIBMTOCDC
C HAVE TSP KEYWORD *******END******
CCDCT0IBM
C
200 PROFF. = FALSE.
CIBMTOCDC
C NEXT IS ENTRY TO SUPER
CCDCT0IBM
   CALL RETURN
CIBMTOCDC
C NO RETURN TO THIS POINT FROM ENTRY RETURN OF SUPER
CCDCT0IBM
C
C SECTION FOR MASK
C
300 DO 310 I=1,N     
   310 MASK(I) = X(I)
GO Te 1
C
CIBMTOCDC
C HAVE TSP KEYWORD *******SMPL*******
CCDCT0IBM
C
400 NARG = N - 1
CIBMTOCDC
C
C REPLACE NWORD + 1 WITH 2 
      KK = 2
LOAD 226
LOAD 227
LOAD 228
LOAD 229
LOAD 230
LOAD 231
LOAD 232
LOAD 233
LOAD 234
LOAD 235
LOAD 236
LOAD 237
LOAD 238
LOAD 239
LOAD 240
LOAD 241
LOAD 242
LOAD 243
LOAD 244
LOAD 245
LOAD 246
LOAD 247
LOAD 248
LOAD 249
LOAD 250
LOAD 251
LOAD 252
LOAD 253
LOAD 254
LOAD 255
LOAD 256
LOAD 257
LOAD 258
LOAD 259
 LOAD 260
LOAD 261
LOAD 262
LOAD 263
LOAD 264
LOAD 265
LOAD 266
LOAD 267
LOAD 268
LOAD 269
LOAD 270
LOAD 271
LOAD 272
LOAD 273
LOAD 274
LOAD 275
LOAD 276
LOAD 277
LOAD 278
LOAD 279
LOAD 280
LOAD 281
LOAD 282
LOAD 283
LOAD 284
LOAD 285
LOAD 286
LOAD 287
LOAD 288
LOAD 289
LOAD 290
LOAD 291
LOAD 292
LOAD 293
LOAD 294
LOAD 295
LOAD 296
LOAD 297
LOAD 298
LOAD 299
LOAD 300
LOAD 301
LOAD 302
CCDCT0IBM
   DO 520 I=2,N
   CALL ARGPUT(I-1,X(KK),TYPE(I),O)
520  KK = KK + 1
   CALL SMPL
   GO TO 1
C
CI8MT0CDC
C HAVE TSP KEYWORD ******N0PRINT******
CCDCT0IBM
C TURN OFF PRINTING OF DATA AS LOADED
C
   500 PROF= TRUE.
   GO TO 1
END

05. 23. 33. J0 16 EP30 6 FEET
SUBROUTINE GRAPH

THIS IS AN ADAPTATION OF **PLOT-C**

C CIBMT0CDC

COMMON /TSPCOM/
* MNSZ, NARG, NSPARG, NWORD, LENGTH,
* NTYPE, IFDBG, IFITL, NCHAR, NSUP,
* NARG, NWORD, IFPLOT, IFFAST, NPAGE,
* NUMLIN, IFREPL, PROFF, SKIP(11), DPHAS,
* LIMARG, LINE, NJARG, NARG, NAME,
* JARG(4)

LENGTH OF JARG SET IN MAIN OVERLAY

DELETED NAME2 Between NAME AND JARG IN /TSPCOM/ (NWORD = 1) TSPCOM

LOGICAL IFDBG, IFITL, IFPLOT, IFFAST,
* IFREPL, PROFF

NEW COMMON BLOCK ADDED
COMMON /MEMCOM/ INNSZ(7). USRNAME, PSSVL(20)
INTEGER USRNAME
EQUIVALENCE
* (INNSZ(1), MNSZ), (INNSZ(2), LLMARG), (INNSZ(3), LLMOUT),
* (INNSZ(4), LLMBUF), (INNSZ(5), LLMSYM), (INNSZ(6), LLMSP),
* (INNSZ(7), LLMOUT)

INNSZ USED TO KEEP TRACK OF MEMORY USE
DIMENSION IPSSVL(20)
EQUIVALENCE (IPSSVL(1), PSSVL(1))

PSSVL AND IPSSVL USED TO PASS VARIABLES BETWEEN SUBROUTINES
IN DIFFERENT OVERLAYS

IPSSVL(1) IS USED TO KEEP RECORD OF USE OF COMMON IN LOAD
AND IN GEN

IPSSVL(2) PASSES COMPUTED GO TO INDEX FROM EXEC TO MATRIX
IPSSVL(3) PASSES COMPUTED GO TO INDEX FROM EXEC TO OVERLAYS
IPSSVL(4) IS USED TO KEEP RECORD OF USE OF JARG-COMMON
IPSSVL(5) IS USED TO KEEP RECORD OF USE OF DOT-COMMON
IPSSVL(6) IS USED TO KEEP RECORD OF USE OF BUFFER-COMMON
IPSSVL(7) IS USED TO KEEP RECORD OF USE OF JSML-COMMON
IPSSVL(8) IS USED TO KEEP RECORD OF USE OF OUTBU-COMMON
EQUIVALENCE
* (JARUSE, IPSSVL(1)), (JTABVL. IPSSVL(2)), (JTABMAT. IPSSVL(3))
* (JARUSE. IPSSVL(4)), (DOTUSE. IPSSVL(5)), (DOTUSE. IPSSVL(6)),
* (MPUSE. IPSSVL(7)), (MPUSE. IPSSVL(8))
INTEGER DOTUSE, BUFUSE, SMPUSE, OBUSE

INSERTED LIMITS ON ARRAY LENGTHS IN GRPALTHX (CALLED BELOW).
DATA LIMGRX / 5000 /

C CDDCT0IBM

COMMON SPACE(1)
IF(NARG, GT, 2) CALL ERG(130, NARG)
IF(NWORD, GT, 1500) CALL ERG(131, NWORD)
12=NWORD+1
13=NSPARG+1
14=NARG+1
IF(14, GT, MNSZ) CALL ERG(132, [4]

C CIBMT0CDC
C
C *********************************************************
C MAJOR ERROR IN IBM VERSION,
C
CALL OF GRAPHX CAN DESTROY DATA IN BLANK COMMON.

THE CORRECTION BELOW IS CONSERVATIVE--MAXIMUM POSSIBLE USE
OF BLANK COMMON BY GRAPHX IS TESTED FOR

IF(LIMGRX .LT. MEMSIZ)
THEN HAVE ENOUGH ROOM IN BLANK COMMON

* ELSE NO ENOUGH ROOM
READ(10,LIMGRX, LENSIZ
10 FORMAT(45HO*** ERROR. EXECUTION OF GRAPH MIGHT DESTROY ,
* 28H DATA IN BLANK COMMON. NEED. I1O, 6H WORDS, ,
* 10H ONLY HAVE. I1O, 10H AVAILABLE )
CALL ERG(112, ITEST)
RETURN TO THIS POINT FROM ERG

20 CONTINUE

C END OF CORRECTION
C
C ADDED A RECORD OF USE OF BLANK COMMON USE
C LOUSE = MAX(LOUSE, ITEST)
CCDCTOIBM
CALL TSMGET(NARG, JARG, SPACE)
CALL GRAPHX(SPACE(1), SPACE(12))
RETURN
END
SUBROUTINE GRAPHX(YVAR, XVAR)

THIS IS A REVISION OF **PL0T C**
22 JUNE 1968

NOTE THAT SPACE MUST BE DIMENSIONED .GE. 5000 IN FRESH
THIS ROUTINE IS GENERALLY DQPPED IN THE 7094 VERSION

CIBMT0CDC
COMMON /TSPC0M/
* MEMSIZ, NGB , NSPARG, NWORD, LENGTH,
* NTYPE, IFDEBUG, IFTITL, NCHAR, NSUP ,
* MEMT, NREG, IFPLOT, IFFAST, NPAGE ,
* NUMLIN, IFREPL, PR0FF , SKIP(11), JPHAS ,
* LIMARG, LINE , NJARG, NARG , NAME ,
* JARG(4)
LENGTH OF JARG SET IN MAIN OVERLAY
C
DELETED NAME2 BETWEEN NAME AND JARG IN /TSPC0M/ (NWORD = 1)TSPC0M
LOGICAL IFDEBUG, IFTITL, IFPLOT, IFFAST,
* IFREPL, PR0FF

CCDCTOIBM
COMMON SPACE(3000), IBEGIN(250), IEND(250), INDY(1500)
DIMENSION A(16), N0TIE(100), TAB(100), TA(100), TABLE(100), X(100), Y<10GRAPHX
D(1), XVAR(1), YVAR(1)
INTEGER IA
LOGICAL IQ'
DATA A, I1H, I1H0, 1H1, 1H2, 1H3, 1H4, 1H5, 1H6, 1H7, 1H8, 1H9, 1H10, 1H , 1H+ , 1H
G, 1H' ,
DO 7 I=1,250
IEND(1)=0
7 IBEGIN(1)=0

ARRANGE Y DATA BY DECREASING ORDER

5 DO 32 ID=1, NGB
DO 32 JD=1D, NGB
IF(YVAR(ID), GE, YVAR(JD))GO TO 32
DIAGS=VAR,ID
VAR,ID=VAR(JD)
VAR(JD)=DIAGS
DIAGS=VAR,ID
VAR,ID=VAR(JD)
VAR(JD)=DIAGS
32 CONTINUE

C FIND RANGES AND INCREMENTS
C
XMA=VAR(1)
XMIN=XMA
DO 18 JB=2,N0B
IF(XMA.LT.XVAR(JB))XMA=XVAR(JB)
IF(XMIN.GT.XVAR(JB))XMIN=XVAR(JB)
18 CONTINUE
IF(XMA.LE.0)130 TO 404
IF(XMIN.GE.0)G0 TO 404
RANGEX=XMA+ABS(XMIN)
60 TO 405

404 RANGEX=ABS(XMA-XMIN)
405 IF(YVAR(1).LE.0)G0 TO 408
IF(YVAR(N0B).GE.0)G0 TO 408
RANGEY=YVAR(1)+ABS(YVAR(N0B))
G0 TO 409

408 RANGEX=ABS(YVAR(1)-YVAR(N0B))
409 XINC=RANGEX/99.
YINC=RANGEY/103.
X(1)=XMIN
Y(1)=YVAR(1)
DO 27 IA=1,99
27 X(IA+1)=X(IA)+XINC
DO 28 IB=1,103
28 Y(IB+1)=Y(IB)-YINC
Y(104)=Y(104)-0.0049
J1=1
DO 102 II=1,N0B
DO 101 J1=J1,104
IF(YVAR(J1).LT.Y(J1))G0 TO 101
INDY(J1)=J1
J1=J1
G0 TO 102
101 CONTINUE
102 CONTINUE

C CHECK FOR TIES, INITIAL + FINAL INDICES STORED IN IBEGIN + IEND
C (IT= TIE NO.)
C
97 IT=0
NA1=1
NA2=N0B-1
34 CONTINUE
DO 38 NA=NA1,NA2
NB=NA+1
IF(INDY(NA).NE.INDY(NB))G0 TO 38
IT=IT+1
IF(IT.GT.250)CALL ERG(133,IT)
IBEGIN(IT)=NA
DO 37 MB=NB,N0B
IF(INDY(NA).EQ.INDY(NB))G0 TO 37
NB=NB
IEND(IT)=MB-1
G0 TO 34
37 CONTINUE
IEND(IT)=N0B
G0 TO 111
38 CONTINUE
111 CALL OUTPT
   IF(NUMLIN.GT.0) CALL HEDING
C
   OUTPUT LOOP (THROUGH 87), WRITES ONE LINE OF GRAPH EACH TIME THRU
C
   IM1=0
   JE1 = 1
   LUCK = 0
   DO 86 IO=1,104,2
    II=IO+1
   DO 87 NC=1,100
    TANC)=0
   123 TAB( NC)=A(1)
   40 JE2 = JE1 + 1
   JE3 = JE1
   DO 70 NC = 1,100
   70 TABLE(NC) = A(1)
       JG= TRUE.
       IF(INDY(JE1).NE.IO)GO TO 71
       JQ= FALSE.
       G0 TO 72
   71 IF(INDY(JE1).NE.II)GO TO 48
   72 IF(INDY(JE1).NE.INDY(JE2))GO TO 74
   IM1=IM1+1
   JE1 = IBEGIN(IM1)
   JE3 = IEND(IM1)
   74 DO 77 JG = JE1,JE3
       DO 113 IJ=1,100
       IF(XVAR(JG).LE.X(IJ))GO TO 75
       113 CONTINUE
       IJ=100
       75 IF(TABLE(IJ).NE.AC1))GO TO 170
       IF(TABLE(IJ).EQ.AC1))GO TO 69
       IF(TABLE(IJ).EQ.AC16))GO TO 45
       TA(IJ)=TAC(IJ)+1
       51 TAB(IJ)=AC1)
       GO TO 54
   170 TA(IJ)=TAC(IJ)+1
   54 IF(TA(IJ)<10)G0 TO 62,64,63
   62 LAZY=TAC(IJ)+2
       TABLE(IJ)=AC(LAZY)
       TAB(IJ)=A(1)
       G0 TO 77
   64 LUCK = LUCK+1
       NOTIE(LUCK)=10
       TABLE(IJ)=AC12)
       TAB(IJ)=A(1)
       GO TO 77
   63 NOTIE(LUCK)=NOTIE(LUCK)+1
       GO TO 77
   69 IF(JO)GO TO 45
       TABLE(IJ)=AC16)
       GO TO 65
   45 TABLE(IJ)=AC15)
   65 TA(IJ)=TAC(IJ)+1
   77 CONTINUE
       JE1=JE3 + 1
       IF(JG)GO TO 48
       DO 47 J=1,100
   47 TAB(J)=TABLE(J)
       GO TO 40
   48 IF(JG)GO TO 12
   12 WRITE(6,2017) (TABCM), CM=1,100
       2017 FORMAT(15X00001)
       CM=CM+1
       IF(CM).EQ.12)GO TO 48
       IF(CM).EQ.3)GO TO 2017
       2017 CONTINUE
C   DELETED IF(NWORD.EQ.1) GO TO 22  (NWORD = 1)
C   WRITE(6,2016) JARG(1),JARG(2)
C   2016 FORMAT(7X,2A4)
C   GO TO 13
C
C 12 CONTINUE
CDDCT0IBM
   22 WRITE(6,3016) JARG(1)
CIBM0CDC
C
C   REPLACED 9X,A6
   3016 FORMAT(9X, R8)
CDDCT0IBM
   GO TO 13
   84 WRITE(6,2008) Y(I1), (TABLE(ME), ME=1,100)
2008 FORMAT(1H+12. 3H=F10.1)
   GO TO 86
   85 WRITE(6,2009) (TABLE(ME), ME=1,100)
2009 FORMAT(1H+12X1H=F10.1)
   86 CONTINUE
13=1+HSFARG
CIBM0CDC
C
C   REMOVED IF(NWORD.EQ.1) GO TO 83  (NWORD = 1)
C   AND PRINT SIMILAR TO THE ONE AT 83
CDDCT0IBM
   83 WRITE(6,3010) JARG(13), X(1), X(20), X(40), X(60), X(80), X(100), X(10),
      W(X30), X(50), X(70), X(90)
3010 FORMAT(15X, 100(1H*), R8 / 7X, 2(8X, 1H*), 9(9X), R8)
CDDCT0IBM
   F19. 3, 5(F19. 3, 1X)*F28. 3, 4F20. 3)
   89 CALL HEDING
      IF(LUCK.EQ.0)GO TO 60
      WRITE(6,2004)
2004 FORMAT(10SH LIST OF TIED POINT COUNTS WHERE NUMBER OF TIES IS GREATER THAN 9 (READING DOWN Y-AXIS AND ACROSS X-AXIS))
   L0V1=0
   L0v2=0
   150 L0V2=L0V1+30
      L0V1=L0V2
      IF(L0V1.EQ.LUCK)GO TO 152
      L0V2=L0V2-(L0V1-LUCK)
152 L=L0V1-29
      WRITE(6,2015) (NBTIE(J), J=L, L0V2)
2015 FORMAT(1H, 3014)
      IF(L0V1.LT.LUCK)GO TO 150
60 RETURN
END

05 23. 49. J0 16 EP30       6 FEET
SUBROUTINE CAPITL
CIMT0CDC
C COMMON /TSPCOM/
  * MEMSIZ, NBB , NSPARG, NWORD , LENGTH, 
  * NTYPE, IFDEBUG, IFITLE, NCHAR, NSUP, 
  * NRESP, NREG , IFPLOT, IFFAST, NPAGE, 
  * NNUMIN, IFREPL, PR0FF, SKIP(11), SPHAS, 
  * LIMARG, LINE , NJARG, NARG , NAME , 
  * JARG(4), 
C LENGTH OF JARG SET IN MAIN OVERLAY
C DELETED NAME2 BETWEEN NAME AND JARG IN /TSPCOM/ (NWORD = 1) TSPCOM
C LOGICAL IFDEBUG, IFITLE, IFPLOT, IFFAST,
  * IFREPL, PR0FF
C CDDCT0IBM
C COMMON DINV(600), CAP(600), DNINV(600), REPL(600), XX(2)
CIMT0CDC
C NEXT IS BLANK COMMON SPACE NEEDED BY ARRAYS ABOVE 
DATA LIMSI2 / 2402 /
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C MAJ0R ERROR IN IBM VERSION, NO TEST ON AVAILABLE BLANK 
COMMON SIZE
C ADDED TEST 
IF( LIMSI2 .LT. MEMSIZ )
  THEN HAVE ENOUGH ROOM
  * 
  ELSE NOT ENOUGH ROOM 
WRITE(6,20) LIMSI2, MEMSIZ
20  FORMAT(25H0** ERR0R. CAPITL NEEDS , I10, 
* 25H WORDS IN BLANK COMMON ONLY , I10, 
* 42H AVAILABLE. PROCEDURE CAPITL NOT EXECUTED / ) 
RETURN
C 30 CONTINUE 
C END OF CORRECTION
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
CDDCT0IBM
CALL TSGET(JARG, DINV)
CALL ARGGET(5, XX, JJ, JJ)
NBENCH = XX(1)
CALL ARGGET(6, XX, JJ, JJ)
DELT = XX(1)
CALL ARGGET(7, XX, JJ, JJ)
BENCH = XX(1)
CAP(NBENCH) = BENCH
XDELT = 1. - DELTA
NN = NBENCH + 1
NNN = NBENCH - 1
IF (NN .GT. NBB) GO TO 101
DO 100 J=NN, NBB
100  CAP(J) = XDELT*CAP(J-1) + DINV(J)
101 IF(NNN .EQ. 0) GO TO 106
DO 105 J=1, NNN
105  CAP(K)=CAP(K)+DINV(K+1))/XDELT
106 DO 110 J=1, NBB
110  REPL(J) = DELTA*CAP(J)
110 DNINV(J) = DINV(J) - REPL(J)
CALL ARGGET(20, TYPE, 11)
IF(I'TYPE .EQ. 3) CALL TSPUT(JARG(NSPARG+1), CAP)
CALL ARGGET(3, XX, I'TYPE, JJ)
IF(I'TYPE .EQ. 3) CALL TSPUT(JARG(2+NSPARG+1), DNINV)
CALL ARGGET(4, XX, I'TYPE, JJ)
IF(I'TYPE .EQ. 3) CALL TSPUT(JARG(3+NSPARG+1), REPL)
RETURN
END
SUBROUTINE INPRED(N, JSA, JSB, A, B, PR0D)
CIBMTBDC
C CALLED BY GGMGT GWMGT GTGMLT G2VMLT ORTHQ
C TGGMLT T2VMLT TINY UNTRAN VGMLT
C YFACT
CCDCT01BM
C C THIS SUBROUTINE CALCULATES THE INNER PRODUCT, A@B, OF THE VECTORS
C A AND B.
CIBMTBDC
C SUM I = 1, N A(I, JSA)*A(I, JSB)
C C CORRECTED MARCH 75 WITH ADDITION OF TEMP
C CDDCT01BM
C C NOTE THAT THE INNER PRODUCT OF VECTORS OF ZERO LENGTH IS RETURNED
C AS ZERO
C C DIMENSION A(1), B(1)
C DOUBLE PRECISION XPR0D
CIBMTBDC
* , TEMPA, TEMPB
CCDCT01BM
   J = 1
   XPR0D = 0.
   IF (N) 150, 150, 50
50 NN = JSA*N
   DO 100 I = 1, NN, JSA
CIBMTBDC
   TEMPA = A(I)
   TEMPB = B(J)
   XPR0D = XPR0D + TEMPA*TEMPB
CCDCT01BM
100 J = J + JSB
150 PR0D = XPR0D
RETURN
END
References