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

https://docs.lib.purdue.edu/cstech/100

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.
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

I. 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 PASSI, 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 PASS1, 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 4 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, NOBSxVARS 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, $S$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 $S$NAME CASE1$, the user may give a more detailed program identification, for example:

$S$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 = OLDCASHBALANCE *( 1 + (INTERESTRATE/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 \( a_0 + a_1x + \ldots + a_nx^n \); 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 PASSI, 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: CIBMTOCADC 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

```
DOUBLE PRECISION XPROD
J = 1
XPROD = 0.
DO 100 I = NN, JSA
   XPROD = XPROD + A(I)*B(J)
100  J = J + JSB
```
The CDC compiler generates code which computes the single precision product $A(I) \cdot B(J)$, then makes it double precision with zero lower half, and finally adds this to the double precision XPROD.

One way to carry out this calculation correctly on the CDC6500 is to introduce two more double precision variables, TEMPA, TEMPB, and to replace the DO loop with

```
DO 100 I = NN, JSA
    TEMPA = A(I)
    TEMPB = B(J)
    XPROD = XPROD + TEMPA*TEMPB
100 J = J + JSB
```

Remark: Some of the DATA statement on the listings use the CDC character string specification R, for example DATA NAMES/ 8RNOPRINT /.../. 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 arithmetic and logical expressions.
CIBMTDCOC
SUBROUTINE PASS1
C
C THIS PROGRAM SUPERVISES READING AND PARSING THE INPUT
C
C REPLACED JX(400) (NHORD = 1), ADDED LIM_TYP
C
COMM
* TYPE(200), JX(200), C0DE(600), IX(200), ITYPE(200)
INTEGER TYPE
DATA LX1X1, (NIMNSZ, 2), (NIMNSZ, 3), (NIMNSZ, 4), (NIMNSZ, 5), (NIMNSZ, 6), (NIMNSZ, 7)
INTEGER NIMNSZ(7), USRNAM, PSSVU20)
EQUIVALENCE (LI3DUSE, IPSSVL(1)), (JARGUSE, IPSSVL(2)), (DIMG, IPSSVL(3)),
** (SMUSE, IPSSVL(4)), (BUFUSE, IPSSVL(5)), (BUFUSE, IPSSVL(6)),
** INTEGER DUSE, BUFUSE, SMUSE, OUTUSE
C
C NEW COMMON BLOCK ADDED
COMM (MEMCOM, IMNSZ(7), USRNAM, PSSVU20)
INTEGER USRNAM
EQUIVALENCE
* (IMNSZ(1), IMNSZ(2), IMNSZ(3), LLMARG), (IMNSZ(4), LLMSYM), (IMNSZ(5), LLMSMP), (IMNSZ(6), LLMSMP),
* (IMNSZ(7), LLMOUT)
C
C IMNSZ USED TO KEEP TRACK OF MEMORY USE
DIMENSION IPSSVL(20)
EQUIVALENCE (IPSSVL(1), IPSSVL(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 HELP RECORD OF USE OF JARG-COMMON
C
IPSSVL(5) IS USED TO HELP RECORD OF USE OF DDT-COMMON
C
IPSSVL(6) IS USED TO HELP RECORD OF USE OF BUFFER-COMMON
C
IPSSVL(7) IS USED TO HELP RECORD OF USE OF JSM-COMMON
C
IPSSVL(8) IS USED TO HELP RECORD OF USE OF OUTBUF-COMMON
EQUIVALENCE
* (LADUSE, IPSSVL(1)), (JADUSE, IPSSVL(2)), (JADUSE, IPSSVL(3))
** (JARUSE, IPSSVL(4)), (DURUSE, IPSSVL(5)), (BUFUSE, IPSSVL(6)),
** (SMUSE, IPSSVL(7)), (OUTUSE, IPSSVL(8))
INTEGER DUSE, BUFUSE, SMUSE, OUTUSE
C
CHANGED DIMENSION OF NAMES TO 5 FROM 20
C
CHANGED DIMENSION OF NAMES TO 4 FROM 20
C
NAME5(2)}
C    NEXT ARE OPERATOR CODES FOR TSP FUNCTIONS
C    11 EXP, 12 ABS, 13 LOG AND ALOG
CCDCTOIBM
   DATA JCODE /0, 11, 12, 13, 13/
CIHMTOIBM
   DATA 6 / IRG /
C    DATA NAMES /
C      * GREXPS , GRABS , GRALOG , GRLOG /
C    DATA NAMES /
C      * GREND , 8RGENR , ORNAME , ORBCD , 8REBCDIC /
CCDCTOIBM
   LAGGED = .FALSE.
   IFRPAR = .FALSE.
CIHMTOIBM
   REPLACED 37 WITH 10 TO PRINT THE WORD LINES ON TSP PROGRAM OUTPUT
C    READ NEXT TSP STATEMENT
CCDCTOIBM
   CALL INPT(JX, N, TYPE)
CIHMTOIBM
   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 DSTYPE
C
C ***************************************************************
C MAJOR ERROR IN IBM VERSION. INPT CAN DESTROY DATA IN
C BLANK COMMON SUCH AS PROGRAM STATEMENTS, DATA VALUES
C GENERATED WITH GENR
C A TEST WAS ADDED AT RETURN OF CALL OF INPUT
C IF( N .LE. LIMTYP )
C THEN ARRAYS ARE LARGE ENOUGH FOR INPUT
C * ELSE TOO MUCH DATA
C IF( LIMTYP + N .LT. MEMSIZ )
C THEN HAVE NOT DESTROYED STORED VALUES IN BLANK COMMON
C * ELSE PART OF BLANK COMMON OVER-WRITTEN
C   IRUIN = LIMTYP + N - MEMSIZ
C   WRITE(6, 10010) IRUIN
C 10010  FORMAT( 3HO*** ERROR: DURING READ OF STATEMENTS ,
C 16H DESTROYED LAST , 110. 14H WORDS OF DATA ,
C 23H IN BLANK COMMON. ABORT RUN").
C 10020  CONTINUE
C 10030  WRITE( 6, 10030)
C    *  FORMAT (3HO*** WARNING: READ IN TOO MUCH ,
C   27H DATA FOR ARRAYS IN PAS1 ,
C   30H OPERATIONS CONTINUE CONNECTION .
C
THE FOLLOWING USES THE COLLATING SEQUENCE OF THE CDC 6500
THE Θ-BCD CHARACTER WORDS IN NAMES(1) Satisfy
CHAR. = 12345678 12345678 12345678 12345678 12345678
BCD, EBCDIC, END, GENR, NAME

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

TEST FOR KEYWORD END
IF( JX(1) - NAMES(1) ) 3.100. 4
TEST FOR KEYWORD EBCDIC
3 IF( JX(1) - NAMES(5) ) 5.410.10
TEST FOR KEYWORD GENR
4 IF( JX(1) - NAMES(2) ) 10.200. 6
TEST FOR KEYWORD BCD
5 IF( JX(1) - NAMES(4) ) 10.400.10
TEST FOR KEYWORD NAME
6 IF( JX(1) - NAMES(3) ) 10.300.10
END OF REPLACEMENT

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

10 NARG = N - 1
IF (TYPE(1).NE. 3) CALL ERG(1, JX)
CIDT0CC

MOVE KEYWORD TO NAME STORAGE

REPLACED CALL NM0V(NAME, JX(1)) WITH FOLLOWING (NW0RD = 1)
NAME = JX(1)
35 CONTINUE
CIDT0BM
IF (NARG .EQ. 0) GO TO 50
CIDT0CC

REPLACED KK = NW0RD + 1 WITH THE FOLLOWING (NW0RD = 1)
KK = 2

PUT ARGUMENTS OF TSP STATEMENT INTO TEMPORARY STORAGE IN JARG

DO 40 I=2, N
CALL ARGPUT(I-1, JX(KK), TYPE(I), 0)
KK = KK + 1
CIDT0CC

DELETED THE FOLLOWING (NW0RD = 1)

IF(TYPE(I) .EQ. 3 OR TYPE(I) .EQ. 4) KK = KK + NW0RD + 1
CIDT0BM
40 CONTINUE
C
G0 CONTINUE
C
C FROM STATEMENT FOLLOWING 35 OR 225
C
LINE = LINE + 1
C
C HAVE NAMES AND ARGUMENTS OF COMPLETE TSP STATEMENT STORED
C IN NAME JARG(). MOVE THEM TO BLANK COMMON
C
CCDCT0IBM
CALL LINPUT
GO TO 1
C
CIBMT0CDC
C HAVE TSP KEYWORD *****END*****
C FINISHED READING TSP PROGRAM, STORE END STATEMENT AND
C RETURN TO SUPER TO EXECUTE TSP PROGRAM
C
C REPLACED CALL NM0V(NAME, JX(1)) WITH FOLLOWING (NW0RD = 1)
100 NAME = JX(1)
CCDCT0IBM
NARG=0
LINE=LINE+1
CALL LINPUT
CALL OUTPUT
CIBMT0CDC
C NEXT IS ENTRY TO SUPER
CCDCT0IBM
CALL RETURN
CIBMT0CDC
C NO RETURN TO THIS POINT FROM ENTRY RETURN OF SUPER
CCDCT0IBM
C
CIBMT0CDC
C HAVE TSP KEYWORD *****GENR*****
C MOVE KEYWORD TO NAME STORAGE
C
C REPLACED CALL NM0V(NAME, JX(1)) WITH FOLLOWING (NW0RD = 1)
200 NAME = JX(1)
C
C CONSTRUCT GENR-LINE-IDENTIFIER AND STORE IN JARG(1)
CCDCT0IBM
CALL INVNV(NAME, G.LINE)
CALL ARGPUT(1, NAME, 6, 0)
CIBMT0CDC
C
C PUT LEFT SIDE OF = INTO TEMPORARY STORAGE IN JARG
C
C REPLACED JX(NW0RD + 1) WITH JX(2) (NW0RD = 1)
C CALL ARGPUT(2, JX(2), TYPE(2), 0)
CCDCT0IBM
C
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
CIBMT0CDC
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

KK = 4

CCDCT0IBM

C
D0 220 I=1,N
NITYPE = TYPE(J)
IF(LAGED) GO TO 530

CIDMTCDC
C TEST FOR NAME OR NUMBER
CCDCT0IBM
IF (NITYPE .NE. 2) GO TO 205

CIDMTCDC
C TEST FOR LEFT PAREN
CCDCT0IBM
IF(JX(KK) .EQ. 9) GO TO 500
C OPERATOR
C
204 MMM = JX(KK)

CIDMTCDC
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)
CCDCT0IBM
IF (MMM .EQ. 3 .AND. IX(JJ-1) .EQ. 9 .AND. ITYPE(JJ-1) .EQ. 2)
I = 8
IX(JJ) = MMM
ITYPE(JJ) = 2
JJ = JJ + 1
KK = KK + 1
GO TO 220

CIDMTCDC
C HAVE LEFT PAREN. SEE IF AN OPERATOR (CODE 2) PRECEDES IT.
CCDCT0IBM
500 IF (ITYPE(JJ-1) .EQ. 2) GO TO 201

CIDMTCDC
C LEFT PAREN PRECEDED BY NAME. GET NAME AND SEE IF IT IS
C TSP FUNCTION
CCDCT0IBM
LOOKUP = LL - 1
CALL ARGGET(LOOKUP, NAMED, JTYPE, LAG)

CIDMTCDC
C REPLACED MATCH = IUCOMP(NAMED, NNAMES) WITH THE FOLLOWING
C
C DETERMINE IF NAMED(1) IS ONE OF TSP FUNCTIONS EXP, ABS, LOG, AL0G
MATCH = 1
IF( NAMED(1) .EQ. NNAMES(1) ) MATCH = 2
IF( NAMED(1) .EQ. NNAMES(2) ) MATCH = 3
IF( NAMED(1) .EQ. NNAMES(3) ) MATCH = 4
IF( NAMED(1) .EQ. NNAMES(4) ) MATCH = 5
END OF REPLACEMENT
CCDCT0IBM
IF(MATCH .EQ. 1) GO TO .525
C
CIDMTCDC
C HAVE ONE OF THE FUNCTIONS ABS, EXP, AL0G, LOG
C INSERT ITS CODE AND SET TYPE TO OPERATOR (2)
CCDCT0IBM
C
505 LL = LOOKUP
IX(JJ-1) = JCODE(MATCH)
ITYPE(JJ-1) = 2
GO TO 201
C
CIDMTCDC
C HAVE NONE OF THE FUNCTIONS ABS, EXP, AL0G, LOG
C LEFT PAREN PRECEDED BY NON-TSP-FUNCTION NAME, HAVE LAGGED
C VARIOUS
CCDOTIBM
C
525 LAGGED = .TRUE.
KK = KK + 1
G0 TO 220
530 IF(NTYPE .NE. 1) G0 TO 540
CIBMDCDC
C PROCESSING SIGNED INTEGRAL ARGUMENT OF LAGGED VARIABLE, STORED
C AS REAL TYPE, CONVERT IT TO INTEGER TYPE
CCDOTIBM
MMM = X(KK)
CALL ARGPUT(L00KUP, NAMED, JTYPE, MMM)
IFRPAR = .TRUE.
KK = KK + 1
G0 TO 220
CIBMDCDC
C TURN OFF LAGGED-SWITCH WHEN GET TO RIGHT PAREN
CCDOTIBM
540 IF(NTYPE .NE. 1) G0 TO 540
IF(NTYPE .EQ. 1 .OR. JX(KK) .NE. 1) CALL ERG(69, JX(KK))
KK = KK + 1
G0 TO 220
CIBMDCDC
C PUT NAME OR NUMBER INTO JARG(. )
CCDOTIBM
205 CALL ARGPUT(LL, JX(KK), NTYPE, 0)
IF(NTYPE .EQ. 1 .OR. NTYPE .EQ. 2, 0R. JX(KK) .EQ. 1 .OR. JX(KK) .EQ. 1)
G0 TO 210
IX(JJ) = 2
JTYPE(JJ) = 2
JJ = JJ + 1
210 IX(JJ) = LL + 2
JTYPE(JJ) = -1
JJ = JJ + 1
KK = KK + 1
CIBMDCDC
C DELETED THE FOLLOWING (NWORD = 1)
C IF(NTYPE .EQ. 3 .OR. NTYPE .EQ. 4) KK = KK + NWORD - 1
CCDOTIBM
LL = LL + 1
220 CONTINUE
CIBMDCDC
C HAVE FINISHED STORING CODED GENR STATEMENT, CONVERT IT TO
C STRING WHICH EVAL CAN PROCESS
CCDOTIBM
IX(JJ) = 1
JTYPE(JJ) = 2
JJ = JJ + 1
IX(JJ) = 14
JTYPE(JJ) = 2
CALL COMPLR(IX, JTYPE, CODE, LENGTH, CODE(501), CODE(551))
CIBMDCDC
C STORE STRING IN BLANK COMMON
CCDOTIBM
CALL VPUT(NAME, CODE, LENGTH)
NARG = LL - 1
CIBMTOCDC
225 CONTINUE
CCDCT0IBM
GO TO 50
C
CIBMTOCDC
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
C IBMTOCDC
C 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 *****EBCD*****
C
JUMP = 1
GO TO 415
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
CCDCT0IEN
END
05.23.17 JO 16 EP30 8 FEET
### COMMON /TSPC0M/

- MENSIZ, N0B, NSPARG, NWORD, LENGTH,
- TYPE, IFDBUG, IFTITL, NCHAR, NSUP,
- HENST, NREG, IFPLOT, IFFAST, NPAGE,
- NUMLIN, IFREPL, PROFF, SKIP(11), JPHAS,
- LIMARG, LIME, NJARG, NARG, NAME,
- JARG(4)

LENGTH OF JARG SET IN MAIN OVERLAY

### NEW COMMON BLOCK ADDED

- COMMON /MEMCIM/, IMNSZ(7), USRNAME, PSSVL(20)
- INTEGER USRNAME
- EQUIVALENCE
  - (IMNSZ(1), MMNSIZ), (IMNSZ(2), LIMARG), (IMNSZ(3), LIMMDOT),
  - (IMNSZ(4), LIMBUF), (IMNSZ(5), LMSYM), (IMNSZ(6), LIMSNP),
  - (IMNSZ(7), LIMMOUT)

IMNSZ 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 D0T-COMMON

IPSSVL(6) IS USED TO KEEP RECORD OF USE OF BUFFER-COMMON

IPSSVL(7) IS USED TO KEEP RECORD OF USE OF J5ML-COMMON

IPSSVL(8) IS USED TO KEEP RECORD OF USE OF BUFF-COMMON

EQUIVALENCE

- (LODUSE, IPSSVL(1)), (JT0VVL, IPSSVL(2)), (JT0MAT, IPSSVL(3))
- (JARUSE, IPSSVL(4)), (D0TUSE, IPSSVL(5)), (BUFFUSE, IPSSVL(6)),
- (SMPUSE, IPSSVL(7)), (0BFUSE, IPSSVL(8))

INTEGER D0TUSE, BUFFUSE, SMPUSE, 0BFUSE

### REMOVED /SMPCOM/

### ADDED JX(1) EQUIVALENCED TO X(1)

DIMENSION JX(1)

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

### INTEGER TYPE

### COMMON

- TYPE(100), MASK(200), X(1)

### INTEGER FRMT, DNUMC
DELETED LOGICAL IFCOMP
CHANGE DIMENSION OF NAMES TO 6 FROM 20
DIMENSION NAMES(6)
DATA NAMES/
* BRLOAD, BRENDE, ORMASK, BRSMPL, BANGPRINT, DATA FORMAT /
C
CDDCT0IBM
PROFF= F. ELSE.
CALL INPT
NMASK = 2
MASK(1) = 1
MASK(2) = 80
C
CIBM0CDC
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(X, N, TYPE)
C IF(TYPE(1), EQ. 1) GO TO 1
C WITH THE FOLLOWING
C 1 CONTINUE
C SEARCH FOR TSP STATEMENT KEYWORD OR OPERATOR
NMLEFT = MEMSIZ - FRONT
2 CONTINUE:
CALL INPT(X, N, TYPE)
IF( N . LT. NMLEFT )
C THEN HAVE ENOUGH ROOM
* ELSE HAVE STORED OVER DATA IN BLANK COMMON
RUINED = N - NMLEFT
WRITE( 6, 10003 ) RUINED
10003 FORMAT( 50H##### ERROR. DURING LOAD SECTION, DESTROYED LAST N
* 15, 30H WORDS OF DATA IN BLANK COMMON )
GO TO 10004
CALL ABORT
10004 CONTINUE
IF( TYPE(1), EQ. 1 )
C THEN CONTINUE READING INPUT
* ELSE HAVE FOUND A NON-NUMERIC INPUT VALUE
GO TO 2
C END OF CORRECTION
C*************************************************************************
C INSERTED SLIGHTLY MORE EFFICIENT SEARCH
C REPLACED 2 JJ = 100000(NAMES)
C GO TO (10, 100, 200, 300, 400, 500, 10), JJ
C WITH THE FOLLOWING
C THE FOLLOWING USES THE COLLATING SEQUENCE OF THE CDC6500
C THE 8-BCD CHARACTER WORDS IN NAMES(I) SATISFY THE FOLLOWING
C CHAR. = 12345678 12345678 12345678 12345678 12345678 12345678
C END . LT. LOAD . LT. MASK . LT. NP0RT LT. SMPL
C
C
C TEST FOR MASK
   IF( JX(1) .EQ. NAMES(3) )
C TEST FOR LOAD
3  IF( JX(1) .EQ. NAMES(1) )
C TEST FOR NOPRINT
4  IF( JX(1) .EQ. NAMES(5) )
C TEST FOR END
5  IF( JX(1) .EQ. NAMES(2) )
C TEST FOR SAMPLE
6  IF( JX(1) .EQ. NAMES(4) )
C END OF REPLACEMENT
C **********************************************
C
C none of the keywords were found
C
C others toIBM
   10 CALL ERF(11,X)

C IBM toCDC
C no return to this point from ERF

C
C IBM toCDC
C have this keyword. ********LOAD********
C have just read input, check to see if all are variable names
C
C others toIBM
C
C 100 DO 110 I=2,N
110 IF(TYPE(I) .NE. 3) CALL ERF(12,X)
   NOVAR = N - 1

C others toCDC
C noVAR is the number of variable names just read
C
C *************************************************
C major error in IBM version, input can read in more variable names
C than allowed for by the size of array type.
C if( NOVAR .LT. LIMTRP )
C then the arrays are large enough
C
C 116 CONTINUE
C
C ************
C
C ELSE TOO MANY VARIABLE NAMES
C WRITE( 6, 115 ) NOVAR, LIMTRP
C FORMAT(200,***.ERROR. LOADED , 15.
C 25h variable names. maximum is , 15.
C 17h per call of load )
C CALL ERF(20, NOVAR )
C no return to this point from ERF
C
C ************
C
C 116 CONTINUE
C
C END OF CORRECTION
C
C *************************************************
C
C replaced, nlbuf=nwrd*n+1 with (NWROD = 1)
C NLBUF = N + 1
C
C
C *************************************************
C
C moved next statement up a few lines to cut out some multiplication.
C NWAT = NWAT*NOVAR
C
C
C *************************************************
C
C major error in IBM version
C
C NTEST used to test the size of available blank common
C space contains error.
C
C replaced NTEST = NWAT+NROT*NOVAR with
NTES T = FRON T + N + 1 + NOB + 2*NDATA
C BLANK COMMON USE
C ARRAYS TYPE, MASK = FRON T
C INPUT IN X = N
C TEMPORARY = NOB
C WILL READ = NOB + NOVAR + 1 (THE PLUS 1 IS FOR THE
C TYPE OF LAST DATUM READ)
C WILL STORE = NOB + NOVAR
C TOTAL IS ASSIGNED TO NTES T ABOVE
C
C IF (NTES T .LT. ME MSZ )
C THEN HAVE ENOUGH ROOM
C *
C ELSE WILL DESTROY DATA IN BLANK COMMON
C INE ED = NTES T - ND T A
C WRITE(6,118) NOB, NOVAR, NDATA, NTES T, ME MS Z, INE ED, NTES T
C 118 FORMAT (47HO*** ERROR. TRYING TO LOAD (NO. OBSERVATIONS)*
C * 10H(N0. VARIABLES) = 1; 6.1H*, 16.3H = 110.7H VALUES/
C * 13H; 13TH THIS REQUIRES, 110. 23H WORDS IN BLANK COMMON. ,
C * 10H ONLY HAVE, 110. 11H AVAILABLE. / 13H, 8H INCREASE .
C * 53H BLANK COMMON SIZE OR REWRITE SUBROUTINE LOAD TO LOAD .
C * 12H VALUES WITH, 110. 17H WORDS INSTEAD OF, 110 /
C * 10HABORT RUN )
C CALL AB ORT LOAD
C
C (NOTE THAT A CALL OF ERG WHICH CALLS SUPER WHICH CALLS
C INPT IN IBM VERSION WOULD DESTROY THE DATA)
C
C 119 CONTINUE
C C END OF CONNECTION
C
C $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C NOTE ON EFFICIENT USE OF STORAGE.
C THE IBM VERSION AND THIS VERSION USE NTES T WORDS IN BLANK
C COMMON TO READ NOB + NOVAR DATA. THIS AMOUNT CAN BE REDUCED
C BY NOB + NOVAR IF THE LOADING IS DONE IN A DO-LOOP.
C EACH ITERATION READS ONLY NOVAR DATA, THE VALUES OF THE
C NEXT OBSERVATION FOR EACH VARIABLE. THIS REQUIRES THE
C CONSTRUCTION OF A NEW ENTRY TO INPT TO READ A SPECIFIED
C NUMBER (NOVAR) OF VALUES. SUBROUTINE FORMAT HAS TO BE
C CHANGED SO THAT IT READS ONLY NOVAR VALUES.
C
C C END OF NOTE
C
C $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C RECO RD THE USE OF BLANK COMMON BY LOAD
C LODUSE = MAX( LODUSE, NTES T - NOB + NOVAR )
C
C CDCT B1M
C NDT = 0
C NP0IN T = NLDAT
C
C IMTCDDC
C C REPLACED NW0RD+1, NW0RD+2 WITH 2, 3 (NW0R D = 1)
C JARG(2) = 3
C JARG(3) = 0
C
C 120 CALL XNPT( X(NP0IN T), N, X(NP0IN T+1) )
C XNPT IS AN ENTRY TO NEW VERSION OF INPT
C C REPLACED
C IF(IFCOMP( X(NP0IN T), NAMES(13)) CALL FORMAT(X(NP0IN T), N, NOVAR)
C C TEST FOR KEYWORD *****FORMAT*****
C IF( JX(NP0IN T) .EQ. NAMES(6) )
C * CALL FORMAT( X(NP0IN T), N, NOVAR )
C
C CDCT B1M
IF (N0VAR .GT. 1) GO TO 121
NLBUF = NLDAT

CIBMT0CDC
C
C REPLACED NW0RD + 1 WITH 2 (NW0RD = 1)
L = 2

CCDCT0IBM
GO TO 140
CIBMT0CDC
121 CONTINUE
C
C REMOVED FOLLOWING FROM INSIDE DO-130-LOOP
K = NLBUF + NOB - 1
C
DO 130 I = 1, N0VAR
C
C REPLACED I+NW0RD + 1 WITH I + 1 (NW0RD = 1)
L = I + 1

CCDCT0IBM
NP01NT = NLDAT + I - 1
DO 125 J=NLBUF.K
X(J) = X(NP01NT)
125 NP01NT = NP01NT + N0VAR
CIBMT0CDC
C
C REPLACED CALL NM0V(JARG(1),X(L)) (NW0RD = 1)
JARG(1) = JX(L)

CCDCT0IBM
130 CALL TSPUT(JARG(1),X(NLBUF))
135 IF(N. NE. NDRTA) CALL ERGCIO.N)
GO TO 1
CIBMT0CDC
C REPLACED CALL NM0V(JARG(1),X(L)) (NW0RD = 1)
140 JARG(1) = JXCL)

CCDCT0IBM
CALL TSPUT(JARG(1),X(NLBUF))
GO TO 135
C
CIBMT0CDC
C HAVE TSP KEYWORD *****END*******

CCDCT0IBM
C
200 PROFF=. FALSE.
CIBM0CDC
C
C NEXT IS ENTRY TO SUPER
CCDCT0IBM
CALL RETURN
CIBMT0CDC
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 TO 1
C
CIBMT0CDC
C HAVE TSP KEYWORD *****END*******

CCDCT0IBM
C
400 NARG = N - 1
CIBMT0CDC
C
C REPLACE NW0RD + 1 WITH 2
KK = 2
CCDCTEBM
  DO 520 I=2,N
    CALL ARGPUT(I-1,X(KK),TYPE(I),0)
  520 KK = KK + 1
    CALL SMPL
    GO TO 1
  C
CIBMTCDC
C HAVE TSP KEYWORD *****NOPRINT******
CCDCTEBM
C TURN OFF PRINTING OF DATA AS LOADED
C
  500 PROFF= TRUE.
    GO TO 1
END

LOAD  233
LOAD  234
LOAD  235
LOAD  236
LOAD  237
LOAD  238
LOAD  239
LOAD  300
LOAD  301
LOAD  302
LOAD  303
LOAD  304
LOAD  305
LOAD  306
LOAD  307

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

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

C
SMT0CD C

COMMON /TSPC0M/
* MNSIZ, NOB, NSPARG, NWORD, LENGTH,
* NTYPE, IFDEG, IFITIL, NCHAR, NSUP,
* MEMSIZ, NREG, IPFL0T, IFAST, NPAGE,
* NUMIN, IFREPL, PROFF, SKIP(11), DPHAS,
* LARG(4), LINE, RARG, NARG, NAME,
* LARG(4), NAME.

LENGTH OF JARG SET IN MAIN OVERLAY

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

LOGICAL IFDEG, IFITIL, IPFL0T, IFAST.
* IFREPL, PROFF

NEW COMMON BLOCK ADDED
COMMON /MEMCOM/ IN5Z(7), USRNAME, PSSV1(20)
INTEGER USRNAME
EQUIVALENCE
* (IN5Z(1), MNSIZ), (IN5Z(2), LLMARG), (IN5Z(3), LLM0FT),
* (IN5Z(4), LLMBUF), (IN5Z(5), LLMSYM), (IN5Z(6), LLMSP),
* (IN5Z(7), LLMOUT)

IN5Z USED TO KEEP TRACK OF MEMORY USE
DIMENSION IPSSV1(20)
EQUIVALENCE (IPSSV1(1), PSSV1(1))
PSSV1 AND IPSSV1 USED TO PASS VARIABLES BETWEEN SUBROUTINES IN DIFFERENT OVERLAYS
IPSSV1(1) IS USED TO KEEP RECORD OF USE OF COMMON IN LOAD AND IN GENR.
IPSSV1(2) PASSES COMPUTED GP TO INDEX FROM EXEC TO MATRIX
IPSSV1(2) PASSES COMPUTED GP TO INDEX FROM EXEC TO OVERLAYS
IPSSV1(4) IS USED TO KEEP RECORD OF USE OF JARG-COMMON
IPSSV1(5) IS USED TO KEEP RECORD OF USE OF DOT-COMMON
IPSSV1(6) IS USED TO KEEP RECORD OF USE OF BUFFER-COMMON
IPSSV1(7) IS USED TO KEEP RECORD OF USE OF JMSL-COMMON
IPSSV1(8) IS USED TO KEEP RECORD OF USE OF OUTBUF-COMMON
EQUIVALENCE
* (IDUSE, IPSSV1(1)), (IDAVL, IPSSV1(2)), (IDMAT, IPSSV1(3))
* (JARG5, IPSSV1(4)), (DOTUSE, IPSSV1(5)), (BUFFUSE, IPSSV1(6))
* (SMUSE, IPSSV1(7)), (OBUSE, IPSSV1(8))
INTEGER DOTUSE, BUFUSE, SMUSE, OBUSE

INSERTED LIMITS ON ARRAY LENGTHS IN GRAPHX (CALLED BELOW).

DATA LIMGX /5000

CCDCTRIBM

COMMON SPACE(1)
IF(NARG, GT, 2) CALL ERR(131, NARG)
IF(NOB, GT, 1500) CALL ERR(131, NOB)
12=NOB+1
13=NSPARG+1
14=12+NOB
IF(14, GT, MNSIZ1) CALL ERR(131, 14)

C
SMT0CD 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
GO TO 20

ELSE NO ENOUGH ROOM
WRITE(6,10) LIMGRX, MEMSIZ
10 FORMAT(45H0)*** ERROR. EXECUTION OF GRAPHX MIGHT DESTROY,
* 28H DATA IN BLANK COMMON. NEED, 110, 6H WORDS, 
* 11H ONLY HAVE, 110, 10H AVAILABLE 
CALL ERG(132, ITEST)

NO RETURN TO THIS POINT FROM ERG

CONTINUE

END OF CORRECTION

ADDED A RECORD OF USE OF BLANK COMMON USE
LOADUSE = MAX(LOADUSE, ITEST)

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 DROPPED IN THE 7094 VERSION

COMMON /TSPCOM/
* MEMSIZ, N0B, NSPARG, NWORD, LENGTH,
* NTYPE, IFDBG, IFTITL, NCHAR, NSUP,
* MEMST, NOREG, IFPLT, IFFAST, NFAGE,
* NUMLIN, IFREPL, PROFF, SKIP(11), JPHAS,
* 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, IFTITL, IFPLT, IFFAST,
* IFREPL, PROFF

COMMON SPACE(3000), IBEGIN(250), IEND(250), Indy(1500)
DIMENSION A(16), NOBLE(100), TAB(100), TA(100), TABLE(100), Nx(100), Y(10)
D4), XVAR(1), YVAR(1)
INTEGER TA
LOGICAL JQ
DATA A=1H, 1H0, 1H1, 1H2, 1H3, 1H4, 1H5, 1H6, 1H7, 1H8, 1H9, 1HX, 1H+, 1H-
C=1H+
D0 7 I=1,250
IEND(1)=0
7 IBEGIN(1)=0

ARRANGE Y DATA BY DECREASING ORDER

5 D0 32 ID=1,N0B
D0 32 JD=ID,N0B
IF(YVAR(ID),GE,YVAR(JD))GO TO 32
DIAGS=YVAR(ID)
YVAR(ID)=YVAR(JD)
YVAR(JD)=DIAGS
DIAGS=XVAR(ID)
XVAR(ID)=XVAR(JD)
XVAR(JD)=DIAGS
32 CONTINUE

C FIND RANGES AND INCREMENTS
C
XMA=XVAR(1)
XMIN=XMA
DO 18 JB=2,NDB
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)
G0 TO 405
RANGEY=ABS(YVAR(JB)-YVAR(NDB))
G0 TO 409
408 RANGEY=ABS(YVAR(1)-YVAR(NDB))
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)-0.00049
J1=1
DO 102 II=1,NDB
DO 101 JI=JI,104
IF(YVAR(JI).LT.Y(J1))G0 TO 101
INDY(JI)=JI
JI=JI
G0 TO 102
101 CONTINUE
102 CONTINUE

C CHECK FOR Y TIES, INITIAL + FINAL INDICES STORED IN IBEGIN + IEND
C (IT=TIE N0. )
C
97 IT=0
NA1=1
NA2=NDB-1
34 CONTINUE
DO 38 NA=NA1,NA2
NE=NA+1
IF(INDY(NA) NE INDY(NB))G0 TO 38
IT=IT+1
IF(IT.GT.250)CALL ERG(133,1T)
IBEGIN (IT) = NA
D6 37 NB=NB,NDB
IF(INDY(NA) EQ INDY(NB))G0 TO 37
NP1=MB
IEND(IT) = MB - 1
G0 TO 34
37 CONTINUE
IEND(IT)=NDB
G0 TO 111
38 CONTINUE
111 CALL OUTPT
IF(NUMLIN.GT.0) CALL HEADING
OUTPUT LOOP (THROUGH 87), WRITES ONE LINE OF GRAPH EACH TIME THRU GRAPHX 94

IM1=0
JE1 = 1
LUCK = 0
D0 86 II=1,104,2
D0 123 NC=1,100
TA(NC)=0
123 TAB(NC)=A(1)
40 JE2 = JE1 + 1
JE3 = JE1
D0 70 NC = 1,100
70 TABLE(NC) = A(JE1)
JQ= TRUE.
IF(INDY(JE1), NE. IO) G0 T0 71
JQ= FALSE.
G0 T0 72
71 IF(INDY(JE1), NE II) G0 T0 48
72 IF(INDY(JE1), NE. INDY(JE2) G0 T0 74
IM1=IM1+1
JE1 = IDBEGIN(IM1)
JE3 = IEND(IM1)
74 D0 77 JG = JEL.JE3
D0 113 IJ=1,100
IF(XVAR(JG), LE. X(IJ) G0 T0 75
113 CONTINUE
IJ=100
75 IF(TABLE(IJ), NE. A(I)) G0 T0 170
IF(TAB(IJ), EQ. A(I)) G0 T0 69
IF(TAB(IJ), EQ. A(I16)) G0 T0 45
TA(IJ)=TA(IJ)+1
51 TAB(IJ)=A(I)
G0 T0 54
170 TA(IJ)=TA(IJ)+1
54 IF(TAB(IJ)-10)62,64,63
62 LAY=Y+2
TABLE(IJ)=A(LAZY)
TAB(IJ)=A(I)
G0 T0 77
64 LUCK = LUCK+1
NOTIE(LUCK)=10
TABLE(IJ)=A(I12)
TAB(IJ)=A(I)
G0 T0 77
63 NOTIE(LUCK)=NOTIE(LUCK) + 1
G0 T0 77
69 IF(JO) G0 T0 45
TABLE(IJ)=A(I16)
G0 T0 65
45 TABLE(IJ)=A(I15)
65 TA(IJ)=TA(IJ)+1
77 CONTINUE
JE1=JE3 + 1
IF(JO) G0 T0 48
D0 47 J=1,100
47 TAB(J)=TABLE(J)
G0 T0 40
48 IF(J(IO), EQ. 2) G0 T0 12
12 WRITE(6,17) (TABCMX), ME=1, 100)
201 FORMAT(15X,0.0)
MEX=MEX: (I, 1, 2)-2
IF(MME)85,84,85
CONTINUE
C 12 CONTINUE

C CDDCT0IBM
C WRITE(6,3016) JARG(1)
C CONTINUE
C
C C CDDCT0IBM
C GO TO 13
C 84 WRITE(6,2008) Y(I), (TABLE(ME), ME=1, 100)
C 2008 FORMAT (1H+F12.3, 2H *100A1)
C 86 CONTINUE

C C CIBMTOCDC
C X
C C REFCOPT 9X, R6
C 3016 FORMAT (9X, R6)
C CDDCT0IBM
C GO TO 13
C 84 WRITE(6,2008) Y(I), (TABLE(ME), ME=1, 100)
C 2008 FORMAT (1H+F12.3, 2H *100A1)
C 86 CONTINUE

C C CIBMTOCDC
C X
C C REM0VED IF(NWORD. EQ. 1) GO Te 83 (NWORD = 1)
C AND A PRINT SIMILAR TO THE ONE AT 83
C CDDCT0IBM
C 83 WRITE(6,3010) JARG, X(1), X(20), X(40), X(60), X(80), X(100), X(10), X(30), X(50), X(70), X(90)
C 3010 FORMAT (15X, 100(IH*), R6)
C CIBMTOCDC
C X
C C CHANGED R6 TO R8 (CDC6500)
C 3016 FORMAT (9X, R8)
C CDDCT0IBM
C F1H*)/F19.3, 5(F19.3, 1X)/F28.3, 4F20.3
C 89 CALL HEDING
C IF (Lucky. EQ. 0) GO TO 60
C WRITE(6,2004)
C 2004 FORMAT (10SH LIST OF TIED POINT COUNTS WHERE NUMBER OF TIES IS GREATER THAN 9 (READING DOWN Y-AXIS AND ACROSS X-AXIS))
X GRAPHX 199
L0V1=0
L0V2=0
150 L0V2=L0V1+30
L0V1=L0V2
IF(L0V1 .EQ. Lucky) GO TO 152
L0V2=L0V2-(L0V1-Lucky)
152 L=L0V1-29
WRITE(6,2015) (N0TIE(J), J=L,L0V2)
2015 FORMAT (1H, 3014)
IF(L0V1 .LT. Lucky) GO TO 150
60 RETURN

END

05 23.49 J0 16 EP30 6 FEET
SUBROUTINE CAPITL
CIBM0CDC
C COMMON /TSPCOM/
C * MEMSIZ, NBB , NSPARG, NWORD, LENGTH,
C * NTYPE, IFDEBUG, IFTITL, NCHAR, NSUP,
C * MEMST, NREG, IFPL0T, IFFAST, NPAGE,
C * NUMIN, IFREPL, PROFF, SKIP(11), SJHAS,
C * LIMARG, LINE, NJARG, NARG, NAME,
C * 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, IFTITL, IFPL0T, IFFAST,
C * IFREPL, PROFF
C
C CI00CT0IBM
C COMMON DINV(600), CAP(600), DINV(600), REPL(600), XX(2)
CIBM0CDC
C NEXT IS BLANK COMMON SPACE NEEDED BY ARRAYS ABOVE
DATA LIMSI2 / 2402 /
C
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C MAJOR ERROR IN IBM VERSION: NO TEST ON AVAILABLE BLANK
C COMMON SIZE
C ADDED TEST
IC(LIMSI2 .LT. MEMSIZ )
C THEN HAVE ENOUGH ROOM
C ELSE NOT ENOUGH ROOM
GO TO 30
C
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C END OF CORRECTION
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

CCDCT0IBM
CALL TSGET(JARG,DINV)
CALL ARGGET(5,XX,JJ,JJ)
NBENCH = XX(1)
CALL ARGGET(6,XX,JJ,JJ)
DELTA = XX(1)
CALL ARGGET(7,XX,JJ,JJ)
BENCH = XX(1)
CAP(NBENCH) = BENCH
XDEL = 1. - DELTA
NN = NBENCH + 1
NN = NBENCH - 1
IF (NN .GT. NBB) GO TO 101
DO 100 J=NN,NBB
100 CAP(J) = XDEL*CAP(J-1) + DINV(J)
101 IF(NNN .EQ. 0) GO TO 106
DO 105 J=1,NNN
K = NBENCH - J
105 CAP(K) = (CAP(K+1)-DINV(K+1))/XDEL
106 DO 110 J=1,NBB
REPL(J) = DELTA*CAP(J)
110 DINV(J) = DINV(J) - REPL(J)
CALL ARGGET(6) TYPET 11}
IF(I TYPE .EQ. 3) CALL TSPUT(JARG(NSPARQ+1), CAP)
CALL ARGGET(3, XX, I TYPE, JJ)
IF(I TYPE .EQ. 3) CALL TSPUT(JARG(2+NSPARQ+1), DINV)
CALL ARGGET(4, XX, I TYPE, JJ)
IF(I TYPE .EQ. 3) CALL TSPUT(JARG(3+NSPARQ+1), REPL)
RETURN
END
SUBROUTINE INPR0D(N, JSA, JSB, A, B, PR0D)
C
C THIS SUBROUTINE CALCULATES THE INNER PRODUCT, A@B, OF THE VECTORS A AND B.
C
DIMENSION A(I), B(J)
DOUBLE PRECISION XPR0D
C
CALLED BY GGGMLT GTGMLT GTYMLT ORTHOS
TGGMLT T2YMLT TINY UNTRAN VGGMLT
YFACT
C
CIBMT0DC

SUM I = 1, N A(I, JSA)*A(I, JSB)

C C \text{CORRECTED MARCH 75 WITH ADDITION OF TEMP}
C
C
CIBMT0DC

NOTE THAT THE INNER PRODUCT OF VECTORS OF ZERO LENGTH IS RETURNED AS ZERO
C
C
DIMENSION A(I), B(J)
DOUBLE PRECISION XPR0D
C
CIBMT0DC

* , TEMPA, TEMPB
C
C
CIBMT0DC

J = 1
XPR0D = 0.
IF (N) 150, 150, 50
50 NN = JSA+N
DO 100 I=1, NN, JSA
C
CIBMT0DC

TEMPA = A(I)
TEMPB = B(J)
XPR0D = XPR0D + TEMPA*TEMPB
C
C
C
C
100 J = J + JSB
150 PR0D = XPR0D
RETURN
END

05.24.15. JO 16 EP30 2 FEET
References