A Computer Program Designed to Identify Behavior Patterns in Observational Data

James Shymansky
University of Missouri-St. Louis, jimshy@umsl.edu

John Penick

Jay Wortman

Follow this and additional works at: https://irl.umsl.edu/epir

Recommended Citation
Shymansky, James; Penick, John; and Wortman, Jay, "A Computer Program Designed to Identify Behavior Patterns in Observational Data" (1976). Educator Preparation & Leadership Faculty Works. 29. Available at: https://irl.umsl.edu/epir/29

This Article is brought to you for free and open access by the College of Education at IRL @ UMSL. It has been accepted for inclusion in Educator Preparation & Leadership Faculty Works by an authorized administrator of IRL @ UMSL. For more information, please contact marvinh@umsl.edu.
The technique of macroanalysis has been developed to facilitate the process of examining patterns of behavior. In this technique, sequentially recorded observational data are computer-analyzed in units of three or more codes. Behavior patterns that have been identified from observational data are collected so that the sequence of individual behaviors (codes) is preserved. The analyst decides the pattern length, which may vary from groups of one to five or more successive codes in the data. He/she also has the option of formulating patterns which include repetitive codes or of collapsing the repetitive codes. Collapsing codes reduces strings of repetitive codes into a single code. The following kinds of information are provided in the pattern analysis: pattern identification, listing options according to frequency or beginning character in the pattern, frequency and percentage of patterns, and raw data when the collapsing option is specified. User information, summary sheet of program options, and a sample printout are included. (CS)
technical report 12

A Computer Program Designed to Identify Behavior Patterns in Observational Data

by
James A. Shymensky
John E. Penick
Jay D. Wortman
A Computer Program Designed to Identify Behavior Patterns in Observational Data
Introduction

Macroanalysis is a technique by which sequentially recorded observational data are analyzed in units of three or more individual codes (Campbell, 1973). In dealing with chains of three or more behavior codes, macroanalysis facilitated the examination of "patterns of behaviors;" patterns which are totally ignored in conventional frequency measures and only slightly accounted for in the more sophisticated matrix analyses.

Early studies of classroom climate such as those by Withall (1949, 1951) and Flanders (1965) attempted to deal with behavior patterns in the classroom; however, due to inadequate techniques for positively identifying the exact patterns of behavior, gross descriptors such as "learner-centered" vs "teacher-centered" (Withall) and "direct" vs "indirect" (Flanders) were employed. These broad definitions of behavior have contributed to the inconsistent findings across studies of seemingly similar strategies and have lead to criticism of interaction analysis as a research tool by users in the field (Rosenshine, 1970). Even with such criticism, the need to study behavior patterns in the classroom, especially in terms of establishing teacher effectiveness, is a recognized fact (Smith, 1967). With the introduction of macroanalytic techniques in more recent years, interests in behavior patterns and strategies of teaching and learning in the classroom have been rekindled. Studies by Shymonsky, et al. (1975); Penick,
et al. (1976); and Campbell (1973) in which macroanalytic procedures were employed have been successful in identifying definite strategies or patterns of learning and teaching in various classroom settings.

Unfortunately, the search for behavior patterns in the classroom and elsewhere through the use of macroanalytic techniques has been limited to only a handful of researchers due to the complicated nature of the computer programming required in the analysis. The program described in the following pages is designed to identify the behavior patterns occurring in sequentially recorded observational data.

**General Program Description**

Behavior patterns can be identified from any observational data which are collected in a manner which preserves the sequence of the individual codes. Although the basic program function is the identification of patterns, several types of patterns as well as several characteristics of the patterns are built into the program or can be called at the user's option. Following is a list of the main functions and options contained in the program:

**Pattern Length:**

The behavior patterns are formed by grouping successive codes in the raw data. These groupings can be specified to contain from one to five characters (Note: a grouping size or pattern length of 1 amounts to a simple frequency report of individual codes).
while a pattern length of 2 is equivalent to a conventional pairs analysis characteristic of the matrix procedures).

**Pattern Type:**

Once the pattern length is decided, the user has the option of formulating patterns which include repetitive characters or collapsing repetitive characters within the sequence. The latter process of collapsing is referred to as MACROanalysis (Shymansky, et al., 1975). The collapsing function has the effect of reducing long strings of repetitive codes (e.g., 1) into a single code within behavior patterns. For example, without collapsing, the pattern AXXAB is possible from the data sequence AXXABAACX, etc. Incorporating the collapsing function prohibits the repetition of any code within the patterns generated. Thus, the first 5-code pattern identified in the above sequence is AXABA.

**Pattern Listing:**

Once all patterns within the data are identified, the user can specify the amount of data to be printed and the form of the listing. During the actual program execution, all patterns identified are stored and counted. Obviously, not all patterns will occur with the same frequency. Thus, the user may opt to have the top 100 patterns printed only. Furthermore, the user may opt to have these patterns listed according to frequency or beginning character in the pattern.
Miscellaneous Output:

In addition to the program options listed above, several other pieces of information are provided in the pattern analysis. The frequency of occurrence of all individual behavior codes is provided as well as the percentage of occurrence. For each pattern listed, the frequency and percentage of all patterns identified is provided. When the collapsing option is specified, the average length of the span of codes comprising the raw data from which the collapsed sequence was derived as well as the average number of each individual code within the collapsed span are listed for each collapsed pattern identified.

A sample printout is included in this report to clarify several of the user options and the output provided. Specifications for calling each of the options and other input data are also discussed.

User Information

Following is a list of "Keywords" used in the program to specify pattern parameters. A summary sheet of the program options is contained in Appendix I.

Input Options:

These commands specific where the data are located on the input cards.

(a) BEGIN_FIELD -- This specifies the position of the first character of data on the input card which will
initiate the first behavior pattern. For data beginning in column 21, the control card would contain the message:

BEGIN_FIELD = 21

(b) BEGIN_LENGTH -- This specifies the length of the data field on the input card. In the case of data contained in columns 21-80, the control card would contain the message:

BEGIN_LENGTH = 60

(c) FIELD_SPAN -- This command specifies which characters of the field are to be used in forming the patterns. For example, if the data were contained in columns 21-80 and only the first character of each group of 3 codes were to be used in forming the patterns, i.e., data in column 21, 24, 27, 30, etc. the control card message would be

FIELD_SPAN = 3

If every character were to be used in formulating the pattern, the message would be

FIELD_SPAN = 1

Pattern Options:

As was mentioned earlier, the user has the option of choosing pattern length and collapsing procedures.
(a) **PATLEN** -- To specify pattern length from 2-5, the following control message is used:

\[
\text{PATLEN} = 5
\]

(b) **COLLAPSE** -- Patterns generated from the raw observational data can be specified in one of two forms:

1. collapsed form in which repetitive codes in the raw data are contracted to a single unit code in the identification of patterns, and
2. repetitive form in which repetitive codes within the raw data are preserved and recognized in the patterns.

Collapsing reduces the raw code sequence **AAABBAADB** to a five code pattern of **ABADB** while retaining the repetitive codes results in the five code patterns **AAABB, AABBA, ABBAA, BBAAD, and BAADB**. To activate the collapsing option, the following control card message is used:

\[
\text{COLLAPSE} = 'YES'
\]

A default function in the program will specify that repetitive codes be preserved if the **COLLAPSE = 'YES'** command is deleted.

**Input Character Options:**

(a) **VALID_CHAR** -- This control card contains a listing of all the characters which will be used to create behavior patterns.
(b) INVALID_CHAR -- This card lists the characters which can occur in the data field but which are not to be used in formulating behavior patterns. For example, a data field may purposely contain blanks or a miscellaneous code character. Listing these characters as INVALID_CHARacters will signal the computer to note the occurrence of such characters but to ignore them in the formulation of the behavior patterns.

(c) BREAK_CHAR -- This character is used to denote the end of a data set. Whenever the BREAK_CHARacter is encountered within the data field, the pattern formation is ended and a new pattern is started with the next card. Note that the BREAK_CHARacter must be coded within the bounds specified by the BEGIN_FIELD and BEGIN_LENGTH commands.

(d) INVALID -- This specifies the maximum number of characters which can occur in the data other than the three character types (VALID_CHAR, INVALID_CHAR, and BREAK_CHAR) before the program is terminated. If an illegal character is encountered, an error message will be printed. The control card message which would allow 1000 such errors to be counted would read

INVALID = 1000
Output Options:

Once the program is executed and all patterns have been identified, counted, and stored, the patterns are printed according to the following user options:

(a) **MOST_FREQUENT** -- The user must specify the number of patterns to be listed in the printout. For example, the command

```
MOST_FREQUENT = 100
```

specifies that the 100 most frequently occurring patterns be listed. Although this number is very arbitrary, it has been observed that, beyond the top 50 patterns, the frequency of individual patterns drops off rapidly.

(b) **ALPHA_ORDER** -- Behavior patterns can be listed alphabetically by the first code (should that code be an alpha character) by using the control card message

```
ALPHA_ORDER = 'YES'
```

should the user not want an alphabetical listing, the message

```
ALPHA_ORDER = 'NO'
```

must be used.

As a final note, the program listed herein is efficient and economical for small amounts of data (less than 22,000 raw behavior codes). Beyond that point, the program becomes costly to run.
because all the data are stored in core. For larger amounts of raw data, an alternate program is available which will handle up to 80,000 raw behavior codes. Persons interested in the larger capability should contact the authors for further information.
REFERENCES


## APPENDIX I
### SUMMARY OF COMPUTER KEY WORDS AND USER OPTIONS

<table>
<thead>
<tr>
<th>COMPUTER KEYWORD/OPTION LIST</th>
<th>DEFAULT VALUE</th>
<th>RESTRICTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN_FIELD</td>
<td>1</td>
<td>1 ≤ BEGIN_FIELD ≤ 80</td>
</tr>
<tr>
<td>BEGIN_LENGTH</td>
<td>80</td>
<td>1 ≤ BEGIN_LENGTH ≤ 80</td>
</tr>
<tr>
<td>FIELD_SPAN</td>
<td>1</td>
<td>1 ≤ FIELD_SPAN ≤ 80</td>
</tr>
<tr>
<td>VALID_CHAR</td>
<td>null string</td>
<td>1 ≤ VALID_CHAR ≤ 40</td>
</tr>
<tr>
<td>INVALID_CHAR</td>
<td>null string</td>
<td>INVALID_CHAR ≤ 10</td>
</tr>
<tr>
<td>BREAK_CHAR</td>
<td>'/'</td>
<td>Only one BREAK_CHAR allowed</td>
</tr>
<tr>
<td>PATLEN</td>
<td>?</td>
<td>2 ≤ PATLEN ≤ 5</td>
</tr>
<tr>
<td>COLLAPSE</td>
<td>'NO'</td>
<td>Checks to see if it is YES -- otherwise it is NO</td>
</tr>
<tr>
<td>INVALID</td>
<td>90</td>
<td>INVALID ≤ 32,767</td>
</tr>
<tr>
<td>MOST_FREQUENT</td>
<td>?0</td>
<td>1 ≤ MOST_FREQUENT ≤ 100</td>
</tr>
<tr>
<td>ALPHA_ORDER</td>
<td>'NO'</td>
<td>Checks to see if it is YES -- otherwise it is NO</td>
</tr>
</tbody>
</table>
APPENDIX II

SAMPLE PRINTOUT
HASP SYSTEM LOG

13:32:44 JOB 184 -- MACRO -- BEGINNING EXEC - INIT 8 - CLASS D
13:34:06 JOB 184 END EXECUTION.

//MACRO JOB (---------,5,,1101),'SHYMANSKY'
// EXEC PL1CLG,PARM.PLIL='SM=(2,80,1)',
// REGION.GO=200K
//PL1L.SYSIN DD *
:EF142I - STEP WAS EXECUTED - COND CODE 0004
.CCTNG -- 17.05 SEC. CPU, 17.25 SEC. WAIT, DAC=85, HWM=100K
:EF142I - STEP WAS EXECUTED - COND CODE 0004
.CCTNG -- 2.85 SEC. CPU, 29.40 SEC. WAIT, DAC=468, HWM=98K
:GO.SEQOUT DD SYSOUT=A
:GO.ORDOUT DD SYSOUT=A
:GO.SYSIN DD *
:GO.CARD DD *
:EF142I - STEP WAS EXECUTED - COND CODE 0000
.CCTNG -- 5.55 SEC. CPU, 8.17 SEC. WAIT, DAC=0, HWM=110K
VERSION 5.5

PL/I F COMPILER OPTIONS SPECIFIED ARE AS FOLLOWS--

SM=(2,80,1)

THE COMPLETE LIST OF OPTIONS USED DURING THIS compilation IS--

- EBCDIC
- CHAR60
- NOMACRO
- SOURCE2
- NOMACDCK
- COMP
- SOURCE
- NOATR
- NOXREF
- NOEXTREF
- NOLIST
- LOAD
- NODECK
- FLAGW
- STMT
- SIZE=0096336
- LINECNT=060
- OPT=01
- SORMGIN=(002,080,001)
- NOEXTDIC
- NEST
- OPLIST
- SYNCHKT
*OPTIONS IN EFFECT*

EBCDIC, CHAR60, NOMACRO, SOURCE2, NOMACDCK, COMP, SOURCE, NOATR, NOXREF,
NOEXTREF, NOLIST, LOAD, NODECK, FLAGW, STMT, SIZE=0096336, LINECNT=060, OPT=01,
SORMGIN=(002,080,001), NOEXTDIC, NEST, OPLIST, SYNCHKT
PRNT PROC OPTIONS(MAIN)

STMT LEVEL NEST
1          PRNT PROC OPTIONS(MAIN)
1
2          DCL 1 NODE BASED (NLINKPT),
2          2 LLINK POINTER,
2          2 RLINK POINTER,
2          2 BALLANCE FIXED BIN(15),
2          2 KEY  CHAR(5),
2          2 COUNT FIXED BIN(15),
2          2 LENGT(5) FIXED BIN(15)
3          DCL 1 HEAD BASED (HEADPT),
3          2 NLLPT POINTER,
3          2 ROOTPT POINTER,
3          2 HIGHT FIXED BIN(15)
4          DCL A FIXED BIN(15)
5          DCL ALL CHAR(51) VARYING INIT('')
6          DCL ALPHA ORDER CHAR(3) INIT('NO')
7          DCL BEGIN FIELD FIXED BIN(15) INIT(1)
8          DCL BEGIN LENGTH FIXED BIN(15) INIT(80)
9          DCL BREAK CHAR CHAR(1) VARYING INIT('/*')
10         DCL BREAKBEGIN FIXED BIN(15)
11         DCL BREAKCNT FIXED BIN(15)
12         DCL CARD FILE
13         DCL CARDS FIXED BIN(15) INIT(0)
14         DCL CNT FIXED BIN(15)
15         DCL COLLAP FIXED BIN(15) INIT(0)
16         DCL COLLAPSE CHAR(3) INIT('NO ')
17         DCL DROT FIXED BIN(15) INIT(0)
18         DCL EOF FIXED BIN(15) INIT(0)
DCL FIELD SPAN FIXED BIN(15) INIT(1)
DCL FIRST(0 51) FIXED BIN(31) INIT((52)0)
DCL FIELDEND FIXED BIN(15)
DCL FIRST NODE FIXED BIN(15) INIT(1)
DCL FIRST TIME FIXED BIN(15) INIT(1)
DCL FIRSTCNT FIXED BIN(15) INIT(0)
DCL FREQUENT(50) POINTER
DCL HEADPT POINTER
DCL INERR FIXED BIN(15) INIT(0)
DCL INPT CHAR(80) INIT((80)' ')
DCL INVALID FIXED BIN(31) INIT(50)
DCL INVALID CHAR CHAR(10) VARYING INIT('')
DCL INVALIDBEGIN FIXED BIN(15)
DCL INVALIDCNT FIXED BIN(15) INIT(0)
DCL INVVLDCNT FIXED BIN(15)
DCL LASTCHAR CHAR(1)
DCL MOST FREQUENT FIXED BIN(15) INIT(20)
DCL VCHAR CHAR(1) INIT(' ')
DCL VCOMP FIXED BIN(15) INIT(0)
DCL VLINKPT POINTER
DCL VPOSIT FIXED BIN(15) INIT(81)
DCL VRDCTN FIXED BIN(15) INIT(0)
DCL VRDLSN(101) POINTER
DCL VRDOUT FILE PRINT
DCL P POINTER
DCL PAT CHAR(5) INIT('..........')
DCL PATCNT(5) FIXED BIN(15) INIT (0,0,0,0,0)
DCL PATFND FIXED BIN(31)
DCL PATLEN FIXED BIN(15) INIT(2)
```
PRNT PROC OPTIONS(MAIN)

STMT LEVEL NEST

<table>
<thead>
<tr>
<th></th>
<th>DCL</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>PERCNT</td>
<td>FLOAT</td>
<td>BIN(51)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>PERCNT1(5)</td>
<td>FLOAT</td>
<td>BIN(51)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>PTIGHT</td>
<td>FIXED</td>
<td>BIN(15)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>PTSTACK(30)</td>
<td>POINTER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>Q</td>
<td>POINTER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>R</td>
<td>POINTER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>S</td>
<td>POINTER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>SEQOUT</td>
<td>FILE</td>
<td>PRINT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>SLSH</td>
<td>FIXED</td>
<td>BIN(15)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>SROT</td>
<td>FIXED</td>
<td>BIN(15)</td>
<td>INIT(0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>START</td>
<td>FIXED</td>
<td>BIN(15)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>T</td>
<td>POINTER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>TITLES(5)</td>
<td>CHAR(6)</td>
<td>INIT('FIRST','SECOND','THIRD','FOURTH','FIFTH')</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>TMPORD1</td>
<td>POINTER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>TMPORD2</td>
<td>POINTER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>TOT</td>
<td>FIXED</td>
<td>BIN(15)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>TOTAL</td>
<td>FIXED</td>
<td>BIN(31)</td>
<td>INIT(0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>TOTCHAR</td>
<td>FLOAT</td>
<td>BIN(51)</td>
<td>INIT(0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>TOTPER</td>
<td>FLOAT</td>
<td>BIN(51)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>VALID</td>
<td>FIXED</td>
<td>BIN(15)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>68</td>
<td>VALID CHAR(40) VARYING INIT('')</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>VALIDBEGIN</td>
<td>FIXED</td>
<td>BIN(15)</td>
<td>INIT(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>VALIDCNT</td>
<td>FIXED</td>
<td>BIN(15)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>VALIDEND</td>
<td>FIXED</td>
<td>BIN(15)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
ON ERROR PUT DATA

ON ENDFILE(CARD) BEGIN
  EOF = 1
  GO TO CHNG
END

ON ENDPAGE(ORDOUT) BEGIN
  IF COLLAP = 0 THEN
    PUT FILE(ORDOUT) EDIT('NAME', 'COUNT', 'FREQUENCY',
      (SKIP, A(4), X(4), A(5), X(2), A(9)))
  ELSE
    Pjt FILE(ORDOUT) EDIT('NAME', 'COUNT', 'FREQUENCY',
      (TITLES(I) DO I = 1 TO PATLEN), 'TOTAL')
      (SKIP, A(4), X(4), A(5), X(2), A(9), X(5),
       (PATLEN)(X(2), A(6)), X(6), A(5))
  END

ON ENDPAGE(SEQOUT) BEGIN
  IF COLLAP = 0 THEN
    PUT FILE(SEQOUT) EDIT('NAME', 'COUNT', 'FREQUENCY',
      (SKIP, A(4), X(4), A(5), X(2), A(9)))
  ELSE
    PUT FILE(SEQOUT) EDIT('NAME', 'COUNT', 'FREQUENCY',
      (TITLES(I) DO I = 1 TO PATLEN), 'TOTAL')
      (SKIP, A(4), X(4), A(5), X(2), A(9), X(5),
       (PATLEN)(X(2), A(6)), X(6), A(5))
  END

PUT PAGE
PRNT PROC.OPTIONS(MAIN)

STMT LEVEL NEST

92 1 PUT EDIT('THE FOLLOWING ARE THE EXECUTION PARAMETERS ') (SKIP, A)
93 1 PUT SKIP(2)

94 1 GET DATA(VALID CHAR, INVALID CHAR, BREAK CHAR, COLLAPSE, INVALID, PATLEN, BEGIN FIELD, BEGIN LENGTH, FIELD SPAN, MOST FREQUENT, ALPHA ORDER)

95 1 PUT DATA(VALID CHAR, INVALID CHAR, BREAK CHAR, COLLAPSE, INVALID, PATLEN, BEGIN FIELD, BEGIN LENGTH, FIELD SPAN, MOST FREQUENT, ALPHA ORDER)

96 1 PUT SKIP(2)

97 1 VALIDCNT = LENGTH(VALID CHAR)
98 1 VALIDEND = VALIDCNT
99 1 IF VALIDCNT = 0 THEN DO
100 1 1 PUT EDIT('ERROR** NO VALID CHARACTERS SPECIFIED IN INPUT') (SKIP, A)
101 1 1 INERR = INERR + 1
102 1 1 END
103 1 1 END

104 1 ALL CHAR = VALID CHAR
105 1 BREAKBEGIN = VALIDCNT + 1
106 1 BREAKCNT = LENGTH(BREAK CHAR)
107 1 IF BREAKCNT = 0 THEN DO
108 1 1 ALL CHAR = ALL CHAR BREAK CHAR
109 1 1 END
110 1 1 END
111 1 1 ELSE DO
112 1 1 BREAKBEGIN = 0
113 1 1 PUT EDIT('WARNING ** NO BREAK CHARACTER SPECIFIED') (SKIP, A)
114 1 1 END
115 1 INVLOCNT = LENGTH(INVALID CHAR)
116 1 INVALIDBEGIN = VALIDCNT + BREAKCNT + 1
117 1 IF INVLOCNT = 0 THEN DO
119 1 ALL CHARS = ALL CHARS INVALID CHAR
120 1 ELSE PUT EDIT('WARNING** NO INVALID CHARACTERS, NOT EVEN A BLANK')
121 1 (SKIP,A)
122 1 FIRSTCNT = LENGTH(ALL CHARS)
123 1 IF PATLEN  2 PATLEN 5 THEN DO
125 1 PUT EDIT('ERROR - THIS PROGRAM IS SET FOR MAXIM PATTER LENGTH',
127 1 3 CHARACTERS, THE NUMBER INPUT IS ',PATLEN)
128 1 (SKIP,A,A,F(3))
129 1 END
130 1 IF SUBSTR(COLLAPSE,1,1) = 'Y' THEN COLLAP = 1
132 1 11 PUT EDIT('ERROR ** VALUE FOR INPUT VARIABLE BEGIN FIELD IS ',
133 1 'INCORRECT.') (SKIP,A,A)
134 1 11 INERR = INERR + 1
135 1 IF BEGIN LENGTH  1 BEGIN LENGTH 80 THEN DO
137 1 PUT EDIT('ERROR ** THE VALUE FOR BEGIN LENGTH IS INCORRECT.')
PRNT PROC OPTIONS(MAIN)

STMT LEVEL NEST

138   1   1
      (SKIP,A)
      INERR = INERR + 1
139   1   1
      END

140   1
      FIELDEND = BEGIN FIELD + BEGIN LENGTH - 1
141   1
      IF FIELDEND 80 THEN DO
143   1   1
      PUT EDIT('ERROR ** THE BEGINNING POSITION ON CARD PLUS THE FIELD
LENGTH IS GREATER THAN 80')(SKIP,A,A)
144   1   1
      INERR = INERR + 1
145   1   1
      END

146   1
      IF FIELD SPAN  1 FIELD SPAN  80 THEN DO
148   1   1
      PUT EDIT('THE VALUE FOR FIELD SPAN IS INCORRECT.')(SKIP,A)
149   1   1
      INERR = INERR + 1
150   1   1
      END

151   1
      IF MOST FREQUENT  1 MOST FREQUENT  100 THEN DO
153   1   1
      PUT EDIT('WARNING ** THE VALUE FOR VARIABLE MOST FREQUENT IS GREATER
THAN 100, IT IS RESET TO 100')(SKIP,A,A)
154   1   1
      MOST FREQUENT = 100
155   1   1
      END

156   1
      IF INERR  0 THEN DO
158   1   1
      PUT EDIT('ERROR ** EXECUTION IS TERMINATED BECAUSE OF
PARAMETER ERRORS')(SKIP,A,A)
159   1   1
      GO TO FINI
160   1   1
      END
PRNT PROC OPTIONS(MAIN)

STMT LEVEL NEST

161  1  BRK DO WHIL (EOF = 0)
162  1 1  IF FIRST TIME = 1 THEN DO
164  1 2  FIRST TIME = 0
165  1 2  CALL GETCHAR
166  1 2  DO I = 1 TO PATLEN
167  1 3  CALL GETCHAR
168  1 3  CNT = 1
169  1 3  IF COLLAP = 1 THEN DO
171  1 4  DO WHILE (NCHAR = LASTCHAR)
172  1 5  CNT = CNT + 1
173  1 5  CALL GETCHAR
174  1 5  END
175  1 4  END
176  1 3  SUBSTR(PAT,I,1) = LASTCHAR
177  1 3  PATCNT(I) = CNT
178  1 3  END
179  1 2  END
180  1 1  ELSE DO
181  1 2  CALL GETCHAR
182  1 2  CNT = 1
183  1 2  IF COLLAP = 1 THEN DO
185  1 3  DO WHILE (NCHAR = LASTCHAR)
186  1 4  CNT = CNT + 1
187  1 4  CALL GETCHAR
188  1 4  END
189  1 3  END
190  1 2  CHNG SUBSTR(PAT,1,PATLEN-1) = SUBSTR(PAT,2,PATLEN-1)
191  1 2  DJ I = 1 TO PATLEN-1
192  1 3  PATCNT(I) = PATCNT(I+1)
193  1 3  END
194  1 2  SJBSTR(PAT,PATLEN,1) = LASTCHAR
195  1 2  PATCNT(PATLEN) = CNT
196  1 2  END
PRINT PROC OPTIONS(MAIN)

/* THE FOLLOWING CODE CREATES AND SEARCHES A BALANCED
   BINARY TREE. INSERTING NODES WHEN NOT ALREADY IN TREE AND
   BALANCING TREE AFTER INSERTION, WHEN NECESSARY.
   THE ALGORITHM WAS ADAPTED FROM THE ART OF COMPUTER
   PROGRAMMING, VOLUME 3, SORTING AND SEARCHING BY DONALD
   E. KNUTH, ADDISON-WESLEY 1975 PAGES 455 - 457
*/

DO WHILE (FIRST NODE = 1)
   FIRST NODE = 0
   ALLOCATE HEAD
   ALLOCATE NODE
   NLINKPT = COUNT = 1
   NLINKPT = BALLANCE = 0
   NLINKPT = LLINK = NULL
   NLINKPT = RLINK = NULL
   NLINKPT = KEY = PAT
   HEADPT = HIGHT = 1
   HEADPT = ROOTPT = NLINKPT
   DI 1 = 1 TO PATLEN
   NLINKPT = LENGT(I) = PATCNT(I)
   END
   PATFND = 1
   TOTAL = 1
   GO TO ALL
   END
215 1 1 A1 /* INITIALIZE */
   I = HEADPT
216 1 1 S = HEADPT - ROOTPT
217 1 1 P = HEADPT - ROOTPT
218 1 1 A2 /* COMPARISON */
   NCOMP = NCOMP + 1
219 1 1 IF PAT > P - KEY THEN GO TO A3
220 1 1 IF PAT < P - KEY THEN GO TO A4
221 1 1 DO I = 1 TO PATLEN
222 1 2 P - LENGT(I) = P - LENGT(I) + PATCNT(I)
223 1 2 END
224 1 1 IF Q - BALLANCE = 0 THEN DO
225 1 2 T = P
226 1 2 S = Q
227 1 1 IF Q = NULL THEN DO
228 1 2 ALLOCATE NODE
229 1 2 Q = NLINKPT
230 1 2 P = LLINKPT
231 1 2 Q = NLINKPT
232 1 2 P - LLINK = Q
233 1 2 GO TO A5
234 1 2 END
235 1 2 END
236 1 2 T = P
237 1 1 IF Q = NULL THEN DO
238 1 2 ALLOCATE NODE
239 1 2 S = Q
240 1 2 GO TO A5
PRNT PROC OPTIONS(MAIN)

STMT LEVEL NEST

241  1  2   END
242  1  1   P = Q
243  1  1   GO TO A2

244  1  1   A4    /* MOVE RIGHT */
        Q = P - RLINK
245  1  1   IF Q = NULL THEN DO
247  1  2   ALLOCATE NODE
248  1  2   Q = NLINKPT
249  1  2   P - RLINK = Q
250  1  2   G3 TO A5
251  1  2   END
252  1  1   IF Q - BALLANCE = 0 THEN DO
254  1  2   T = P
255  1  2   S = Q
256  1  2   END
257  1  1   P = Q
258  1  1   GO TO A2

259  1  1   A5    /* INSERT */
        TOTAL = TOTAL + 1
260  1  1   PATFND = PATFND + 1
261  1  1   Q - LLINK = NULL
262  1  1   Q - RLINK = NULL
263  1  1   Q - BALLANCE = 0
264  1  1   Q - KEY = PAT
265  1  1   DO I = 1 TO PATLEN
266  1  2   Q - LENGTH(I) = PATCNT(I)
267  1  2   END
268  1  1   Q - COUNT = 1
269 1 1 A6 /* ADJUST BALANCE FACTORS */
270 1 1 IF PAT S - KEY THEN DO
271 1 2 R = S - LLINK
272 1 2 P = S - LLINK
273 1 2 END
274 1 1 ELSE DO
275 1 2 R = S - RLINK
276 1 2 P = S - RLINK
277 1 2 END
278 1 1 DO WHILE ( P = Q)
279 1 2 IF PAT P - KEY THEN DO
281 1 3 P - BALLANCE = -1
282 1 3 P = P - LLINK
283 1 3 END
284 1 2 ELSE IF PAT P - KEY THEN DO
286 1 3 P - BALLANCE = 1
287 1 3 P = P - RLINK
288 1 3 END
289 1 2 END

290 1 1 A7 /* BALANCE ACT */
291 1 1 IF PAT S - KEY THEN A = -1
292 1 1 ELSE A = 1
293 1 1 IF S - BALLANCE = 0 THEN DO
PRNT PROC OPTIONS(MAIN)

STMT LEVEL NEST

295  1  2  S - BALLANCE = A
296  1  2  HEADPT - HIGHT = HEADPT - HIGHT + 1
297  1  2  GO TO A11
298  1  2  END
299  1  1  ELSE DO
300  1  2  IF S - BALLANCE = -A THEN DO
301  1  3  S - BALLANCE = 0
302  1  3  GO TO A11
303  1  3  END
304  1  3  END
305  1  2  ELSE DO
306  1  3  IF S - BALLANCE = A THEN DO
307  1  4  IF R - BALLANCE = A THEN GO TO A8
308  1  4  ELSE IF R - BALLANCE = -A THEN GO TO A9
309  1  4  END
310  1  4  END
311  1  3  END
312  1  2  END
313  1  1  PUT EDIT (' ERROR IN TREE LOOKUP SECTION A7')(SKIP,A)
314  1  1  A8 /* SINGLE ROTATION */
315  1  1  SROT = SROT + 1
316  1  1  R = P
317  1  1  IF A = 1 THEN DO
318  1  2  S - RLINK = R - LLINK
319  1  2  R - LLINK = S
320  1  2  END
321  1  2  END
322  1  1  ELSE DO
323  1  1  S - LLINK = R - RLINK
324  1  2  R - RLINK = S
325  1  2  END
326  1  2  END
327  1  1  S - BALLANCE = 0
328  1  1  R - BALLANCE = 0
329  1  1  GO TO A10
/* DOUBLE ROTATION */

DROT = DROT + 1

IF A = 1 THEN DO
  P = R - LLINK
  R - LLINK = P - RLINK
  P - RLINK = R
  S - RLINK = P - LLINK
  P - LLINK = S
  END

ELSE DO
  P = R - RLINK
  R - RLINK = P - LLINK
  P - LLINK = R
  S - LLINK = P - RLINK
  P - RLINK = S
  END

IF P - BALLANCE = A THEN DO
  S - BALLANCE = -A
  R - BALLANCE = 0
  END

ELSE DO
  IF P - BALLANCE = 0 THEN DO

### PRNT PROC OPTIONS(MAIN)

<table>
<thead>
<tr>
<th>STMT</th>
<th>LEVEL</th>
<th>NEST</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>354</td>
<td>1</td>
<td>3</td>
<td>S - BALLANCE = 0</td>
<td></td>
</tr>
<tr>
<td>355</td>
<td>1</td>
<td>3</td>
<td>R - BALLANCE = 0</td>
<td></td>
</tr>
<tr>
<td>356</td>
<td>1</td>
<td>3</td>
<td>END</td>
<td></td>
</tr>
<tr>
<td>357</td>
<td>1</td>
<td>2</td>
<td>ELSE DO</td>
<td></td>
</tr>
<tr>
<td>358</td>
<td>1</td>
<td>3</td>
<td>IF P - BALLANCE = -A THEN DO</td>
<td></td>
</tr>
<tr>
<td>360</td>
<td>1</td>
<td>4</td>
<td>S - BALLANCE = 0</td>
<td></td>
</tr>
<tr>
<td>361</td>
<td>1</td>
<td>4</td>
<td>R - BALLANCE = A</td>
<td></td>
</tr>
<tr>
<td>362</td>
<td>1</td>
<td>4</td>
<td>END</td>
<td></td>
</tr>
<tr>
<td>363</td>
<td>1</td>
<td>3</td>
<td>END</td>
<td></td>
</tr>
<tr>
<td>364</td>
<td>1</td>
<td>2</td>
<td>END</td>
<td></td>
</tr>
<tr>
<td>365</td>
<td>1</td>
<td>1</td>
<td>P - BALLANCE = 0</td>
<td></td>
</tr>
<tr>
<td>366</td>
<td>1</td>
<td>1</td>
<td>A10 /* FINISHING TOUCH */</td>
<td></td>
</tr>
<tr>
<td>367</td>
<td>1</td>
<td>1</td>
<td>IF S = T - RLINK THEN T - RLINK = P</td>
<td></td>
</tr>
<tr>
<td>368</td>
<td>1</td>
<td>1</td>
<td>ELSE T - LLINK = P</td>
<td></td>
</tr>
<tr>
<td>369</td>
<td>1</td>
<td>1</td>
<td>A11 END BRK</td>
<td></td>
</tr>
</tbody>
</table>
PRNT PROC OPTIONS(MAIN)

STMT LEVEL NEST

370 1 PRNT

371 1 PUT PAGE

372 1 PUT EDIT(' THE TOTAL NUMBER OF PATTERNS FOUND IS ',PATFND)
         (SKIP,A,F(7))

373 1 PUT EDIT(' THE TOTAL NUMBER OF DIFFERENT PATTERNS FOUND IS ',TOTAL)
         (SKIP,A,F(7))

374 1 PUT EDIT(' CHARACTER','COUNT','FREQUENCY')
         (SKIP(3),A(10),X(10),A(5),X(3),A)

375 1 1 DO I = 1 TO VALIDEND

376 1 1 PERCNT = FIRST(I) / TOTCHAR

377 1 1 END

378 1 1 DO I = VALIDEND + 1 TO FIRSTCNT

379 1 1 PUT EDIT (SUBSTR(ALL CHARS,I,1),FIRST(I))
         (SKIP,X(5),A,X(10),F(7))

380 1 1 END

381 1 1 PUT EDIT('INVALID',FIRST(0))(SKIP,X(5),A,X(4),F(7))

382 1 1 IF SUBSTR(ALPHA ORDER,1,1) = 'Y' THEN SIGNAL ENDPAGE(SEQOUT)

/* THE FOLLOWING CODE TRAVERSERS THE BINARY TREE,
   PRINTING EACH NODE IN INORDER SEQUENCE. THE ALGORITHM
   WAS ADAPTED FROM THE ART OF COMPUTER PROGRAMMING,
   VOLUME 1, FUNDAMENTAL ALGORITHMS BY DONALD E. KNUTH
   ADDISON-WESLEY, 1975 PAGES 317 - 318 */
384 1 T1 /* INITIALIZE */
PHTIGHT = 0
P = HEADPT - ROJPT
385 1
386 1 T2 /* IS P = NULL */
387 1 IF P = NULL THEN GO TO T4
388 1 T3 /* PUT P ON STACK */
PHTIGHT = PHTIGHT + 1
PTSTACK(PHTIGHT) = P
389 1 P = P - LLINK
390 1 GO TO T2
391 1
392 1 T4 /* POP STACK */
393 1 IF PHTIGHT - 1 = 0 THEN GO TO T6
394 1 P = PTSTACK(PHTIGHT)
395 1 PHTIGHT = PHTIGHT - 1
396 1 T5 /* VISIT NODE P */
397 1 IF ORDCNT = 0 THEN DO
398 1 1 ORDCNT = 1
399 1 1 ORDLST(1) = P
400 1 1 END
401 1 ELSE DO
402 1 1 TMPORD1 = ORDLST(ORDCNT)
403 1 1 IF P - COUNT = TMPORD1 - COUNT THEN DO
405 1 2 IF ORDCNT = MOST FREQUENT THEN ORDCNT = ORDCNT + 1
407 1 2 DJ I = ORDCNT TO 2 BY -1
PRNT PROC OPTIONS(MAIN)

STMT LEVEL NEST

408 1 3 TMPORD1 = ORDLST(I - 1)
409 1 3 IF TMPORD1 = COUNT = P - COUNT THEN DO
410 1 4 ORDLST(I) = P
411 1 4 GO TO T5A
412 1 4 END
413 1 4 END
414 1 3 ORDLST(I) = ORDLST(I - 1)
415 1 3 END
416 1 2 ORDLST(I) = P
417 1 2 END
418 1 1 ELSE DO
419 1 2 IF ORDCNT ) MOST FREQUENT THEN DO
420 1 3 ORDCNT = ORDCNT + 1
421 1 3 ORDLST(ORDCNT) = P
422 1 3 END
423 1 3 END
424 1 2 END
425 1 1 END
426 1 T5A
427 1 IF SUBSTR(ALPHA ORDER,1,1) = 'Y' THEN DO
428 1 1 TOT = 0
429 1 1 DO K = 1 TO PATLEN
430 1 2 PERCNT1(K) = P - LENGT(K) / P - COUNT
431 1 2 TOT = TOT + P - LENGT(K)
432 1 2 END
433 1 1 TOTPER = TOT / P - COUNT
434 1 1 PERCNT = COUNT / PATFND
435 1 1 IF COLLAP = O THEN PUT FILE(SEQQJT) EDIT(SUBSTR(P - KEY,1,PATLEN),
436 1 1 P - COUNT,PERCNT)
437 1 1 (SKIP,A(PATLEN),X(5-PATLEN),F(7),X(3),F(9,7))
ELSE PUT FILE(SEQOUT) EDIT(SUBSTR(P - KEY, 1, PATLEN),
  P COUNT, PERCNT, (PERCNT1(I) DO I = 1 TO PATLEN),
  TOTPER)
  (SKIP, A(PATLEN), X(5 - PATLEN), F(7), X(3), F(9, 7), X(5),
  (PATLEN)(X(2), F(6, 1)), X(4), F(7, 1))

END
P = P - RLINK
GO TO T2

T6 /* FINISH PRINTING */
SIGNAL ENDPAGE(ORDOUT)
DO I = 1 TO ORDCNT
  TMPORD1 = JRDLST(I)
  TJT = 0
  DO K = 1 TO PATLEN
    PERCNT1(K) = TMPORD1 - COUNT
    PERCNT1(K) = TMPORD1 - LENGT(K) / TMPORD1 - COUNT
    TOT = TOT + TMPORD1 - LENGT(K)
  END
  TOTPER = TOT / TMPORD1 - COUNT
  PERCNT = TMPORD1 - COUNT / PATFND
  IF COLLAP = 0 THEN PUT FILE(ORDOUT) EDIT(SUBSTR(TMPORD1 - KEY, 1, PATLEN),
    TMPORD1 - COUNT, PERCNT, (PERCNT1(J) DO J = 1 TO PATLEN),
    TOTPER)
FRNT PROC OPTIONS (MAIN)

STMT LEVEL NEST

(SKIP, A(PATLEN), X(5-PATLEN), F(7), X(3), F(9,7), X(5).

(PATLEN)(X(2), F(6,1)), X(4), F(7,1))

455 1 1 END
PRNT PROC OPTIONS(MAIN)

STMT LEVEL NEST

456   1   GETCHAR PROC
457   2   LASTCHAR = NCHAR
458   2   INVLD
459   2   IF SLSH = NPOSIT - FIELD SPAN & SLSH THEN DO
460   2 1   FIRST(BREAKBEGIN) = FIRST(BREAKBEGIN) + 1
461   2 1   NPOSIT = 81
462   2 1   NCHAR = ""
463   2 1   SLSH = 161
464   2 1   FIRST TIME = 1
465   2 1   GO TO BRK
466   2 1   END
467   2   IF NPOSIT THEN DO
469   2 1   GET FILE(CARD) EDIT (INPT) (COL(1),A(60))
470   2 1   CARDS = CARDS + 1
471   2 1   NPOSIT = BEGIN FIELD
472   2 1   IF FIELD SPAN THEN DO
474   2 2   SLSH = INDEX(SUBSTR(INPT,BEGIN FIELD,
475   2 2                  BEGIN LENGTH),BREAK CHAR)
476   2 2   IF SLSH THEN SLSH = BEGIN FIELD + SLSH - 1
477   2 2   ELSE SLSH = 161
478   2 2   END
479   2 1   END
480   2   NCHAR = SUBSTR(INPT,NPOSIT,1)
481   2   NPOSIT = NPOSIT + FIELD SPAN
482   2   VALID = INDEX(ALL CHAR,S,NCHAR)
483   2   FIRST(VA) = FIRST(VA) + 1
484   2   IF VALID THEN DO
486 2 1 PUT EDIT('ERROR - ON CARD',CARDS,' AND IN COLUMN ',NPOSIT-FILE SPAN,
'THERE IS AN INVALID CHARACTER -- ',NCHAR,
')-- CHARACTER IS SKIPPED')
(SKIP,A,F(7),A,F(7),A,A,A)

487 2 1 INVALIDCNT = INVALIDCNT + 1
488 2 1 IF INVALIDCNT = INVALID THEN DO
490 2 2 PUT EDIT('NUMBER OF INVALID CHARACTERS EXCEEDS MAXIMUM SPECIFIED',
INVALID,' PROGRAM TERMINATION')(SKIP,A,F(5),A)
491 2 2 GO TO PRNT
492 2 2 END
493 2 1 GO TO INVLD
494 2 1 END
495 2 ELSE DO
496 2 1 IF VALID )= VALIDEND THEN DO
498 2 2 TOTCHAR = TOTCHAR + 1
499 2 2 END
500 2 1 ELSE DO
501 2 2 IF VALID = INVALIDBEGIN THEN GO TO INVLD
503 2 2 ELSE DO
504 2 3 NPOSIT = 81
505 2 3 NCHAR = ' '
506 2 3 FIRST TIME = 1
507 2 3 GO TO BRK
508 2 3 END
509 2 2 END
510 2 1 END
511 2 END GETCHAR
512 1 FINI END PRNT
PRNT PROC OPTIONS(MAIN)

STORAGE REQUIREMENTS.
--------------------------

THE STORAGE AREA FOR THE PROCEDURE LABELLED PRNT IS 2020 BYTES LONG.
THE STORAGE AREA FOR THE ON UNIT AT STATEMENT NO. 72 IS 224 BYTES LONG.
THE STORAGE AREA FOR THE ON UNIT AT STATEMENT NO. 75 IS 224 BYTES LONG.
THE STORAGE AREA FOR THE ON UNIT AT STATEMENT NO. 80 IS 272 BYTES LONG.
THE STORAGE AREA FOR THE ON UNIT AT STATEMENT NO. 86 IS 272 BYTES LONG.
THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED GETCHAR IS 352 BYTES LONG.
THE PROGRAM CSECT IS NAMED PRNT AND IS 14145 BYTES LONG.
THE STATIC CSECT IS NAMED ***PRNTA AND IS 4480 BYTES LONG.

*STATISTICS* SOURCE RECORDS = 542, PROG TEXT STMNTS = 512, OBJECT BYTES = 14146
PRNT PROC OPTIONS(MAIN)

COMPILER DIAGNOSTICS.

WARNINGS

IEM0227I NO FILE/STRING OPTION SPECIFIED IN ONE OR MORE GET/PUT STATEMENTS.
SYSSIN/SYSPRINT HAS BEEN ASSUMED IN EACH CASE.

IEM0764I ONE OR MORE FIXED BINARY ITEMS OF PRECISION 15 OR LESS HAVE BEEN GIVEN
HALFWORD STORAGE. THEY ARE FLAGGED '*********' IN THE XREF/ATR LIST.

IEM3896I COMPILER CORE REQUIREMENT EXCEEDED SIZE GIVEN. AUXILIARY STORAGE USED.

END OF DIAGNOSTICS.

AUXILIARY STORAGE WILL NOT BE USED FOR DICTIONARY WHEN SIZE = 69K

COMPILE TIME .28 MINS
ELAPSED TIME .64 MINS
THE FOLLOWING ARE THE EXECUTION PARAMETERS

VALID CHAR='ELMOQRSTWZ' INVALID CHAR=' ' BREAK CHAR='/' COLLAPSE='YES'

INVALID= 50 PATLen= 5 BEGIN FIELD= 21

BEGIN LENGTH= 60 FIELD SPAN= 3 MOST FREQUENT= 50

ALPHA ORDER='NO'
The total number of patterns found is 1994
The total number of different patterns found is 1243

<table>
<thead>
<tr>
<th>Character</th>
<th>Count</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>1796</td>
<td>0.327977</td>
</tr>
<tr>
<td>L</td>
<td>885</td>
<td>0.161614</td>
</tr>
<tr>
<td>M</td>
<td>221</td>
<td>0.040358</td>
</tr>
<tr>
<td>O</td>
<td>295</td>
<td>0.053871</td>
</tr>
<tr>
<td>Q</td>
<td>75</td>
<td>0.013696</td>
</tr>
<tr>
<td>R</td>
<td>1051</td>
<td>0.191928</td>
</tr>
<tr>
<td>S</td>
<td>0</td>
<td>0.000000</td>
</tr>
<tr>
<td>T</td>
<td>304</td>
<td>0.055515</td>
</tr>
<tr>
<td>W</td>
<td>726</td>
<td>0.132579</td>
</tr>
<tr>
<td>Z</td>
<td>123</td>
<td>0.022462</td>
</tr>
<tr>
<td>/</td>
<td>29</td>
<td>0.004167</td>
</tr>
<tr>
<td>invalid</td>
<td>0</td>
<td>0.000000</td>
</tr>
<tr>
<td>NAME</td>
<td>COUNT</td>
<td>FREQUENCY</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td>-----------</td>
</tr>
<tr>
<td>ELELE</td>
<td>30</td>
<td>0.0150299</td>
</tr>
<tr>
<td>LELEL</td>
<td>23</td>
<td>0.0115204</td>
</tr>
<tr>
<td>NEWEW</td>
<td>21</td>
<td>0.0105286</td>
</tr>
<tr>
<td>EMEME</td>
<td>20</td>
<td>0.0100250</td>
</tr>
<tr>
<td>EWEWE</td>
<td>20</td>
<td>0.0100250</td>
</tr>
<tr>
<td>_RLRL</td>
<td>19</td>
<td>0.0095215</td>
</tr>
<tr>
<td>_OLOL</td>
<td>15</td>
<td>0.0075073</td>
</tr>
<tr>
<td>=MEMEM</td>
<td>15</td>
<td>0.0075073</td>
</tr>
<tr>
<td>_RLRLR</td>
<td>15</td>
<td>0.0075073</td>
</tr>
<tr>
<td>DOLG</td>
<td>14</td>
<td>0.0070190</td>
</tr>
<tr>
<td>TELEL</td>
<td>11</td>
<td>0.0055084</td>
</tr>
<tr>
<td>ELTEL</td>
<td>8</td>
<td>0.0039978</td>
</tr>
<tr>
<td>_TELE</td>
<td>8</td>
<td>0.0039978</td>
</tr>
<tr>
<td>_RTEL</td>
<td>8</td>
<td>0.0039978</td>
</tr>
<tr>
<td>_TELE</td>
<td>8</td>
<td>0.0039978</td>
</tr>
<tr>
<td>_TLTL</td>
<td>8</td>
<td>0.0039978</td>
</tr>
<tr>
<td>_WLWL</td>
<td>8</td>
<td>0.0039978</td>
</tr>
<tr>
<td>_LETLE</td>
<td>7</td>
<td>0.0035095</td>
</tr>
<tr>
<td>_LTELE</td>
<td>7</td>
<td>0.0035095</td>
</tr>
<tr>
<td>_ETLLE</td>
<td>7</td>
<td>0.0035095</td>
</tr>
<tr>
<td>_LELEO</td>
<td>7</td>
<td>0.0035095</td>
</tr>
<tr>
<td>_ETLE</td>
<td>7</td>
<td>0.0035095</td>
</tr>
<tr>
<td>DOLLR</td>
<td>7</td>
<td>0.0035095</td>
</tr>
<tr>
<td>RERER</td>
<td>7</td>
<td>0.0035095</td>
</tr>
<tr>
<td>ELTLE</td>
<td>6</td>
<td>0.0030060</td>
</tr>
<tr>
<td>LELET</td>
<td>6</td>
<td>0.0030060</td>
</tr>
<tr>
<td>-------</td>
<td>----</td>
<td>-----------</td>
</tr>
<tr>
<td>LRLDL</td>
<td>6</td>
<td>0.0030060</td>
</tr>
<tr>
<td>RODOL</td>
<td>6</td>
<td>0.0030060</td>
</tr>
<tr>
<td>RWLRL</td>
<td>6</td>
<td>0.0030060</td>
</tr>
<tr>
<td>TLTLT</td>
<td>6</td>
<td>0.0030060</td>
</tr>
<tr>
<td>WLWLR</td>
<td>6</td>
<td>0.0030060</td>
</tr>
<tr>
<td>ELELO</td>
<td>5</td>
<td>0.0025024</td>
</tr>
<tr>
<td>ELELT</td>
<td>5</td>
<td>0.0025024</td>
</tr>
<tr>
<td>ELEOE</td>
<td>5</td>
<td>0.0025024</td>
</tr>
<tr>
<td>ELEOL</td>
<td>5</td>
<td>0.0025024</td>
</tr>
<tr>
<td>ELETL</td>
<td>5</td>
<td>0.0025024</td>
</tr>
<tr>
<td>EOLOL</td>
<td>5</td>
<td>0.0025024</td>
</tr>
<tr>
<td>OLELT</td>
<td>5</td>
<td>0.0025024</td>
</tr>
<tr>
<td>LOROR</td>
<td>5</td>
<td>0.0025024</td>
</tr>
<tr>
<td>DLRLR</td>
<td>5</td>
<td>0.0025024</td>
</tr>
<tr>
<td>RLRLW</td>
<td>5</td>
<td>0.0025024</td>
</tr>
<tr>
<td>RLROL</td>
<td>5</td>
<td>0.0025024</td>
</tr>
<tr>
<td>ROROL</td>
<td>5</td>
<td>0.0025024</td>
</tr>
<tr>
<td>RWRWR</td>
<td>5</td>
<td>0.0025024</td>
</tr>
<tr>
<td>TELTE</td>
<td>5</td>
<td>0.0025024</td>
</tr>
<tr>
<td>TLETET</td>
<td>5</td>
<td>0.0025024</td>
</tr>
<tr>
<td>TLTLE</td>
<td>5</td>
<td>0.0025024</td>
</tr>
<tr>
<td>WLRRL</td>
<td>5</td>
<td>0.0025024</td>
</tr>
<tr>
<td>WLWLR</td>
<td>5</td>
<td>0.0025024</td>
</tr>
</tbody>
</table>
## Job Accounting Summary

<table>
<thead>
<tr>
<th>ITEM</th>
<th>QUANTITY, THIS RUN</th>
<th>QUANTITY, ACCUMULATED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute Time</td>
<td>25.45 sec.</td>
<td>4 min. 47.44 sec.</td>
</tr>
<tr>
<td>Wait Time</td>
<td>54.82 sec.</td>
<td>12 min. 58.86 sec.</td>
</tr>
<tr>
<td>Core Storage</td>
<td>9.3 megabyte-seconds</td>
<td>118.8 megabyte-seconds</td>
</tr>
<tr>
<td>Direct Access Usage</td>
<td>553 I/O accesses</td>
<td>6,806 I/O accesses</td>
</tr>
<tr>
<td>Cards In</td>
<td>835 cards</td>
<td>7,746 cards</td>
</tr>
<tr>
<td>Pages Out</td>
<td>22 pages</td>
<td>354 pages</td>
</tr>
<tr>
<td>Lines Out</td>
<td>750 lines</td>
<td>11,562 lines</td>
</tr>
<tr>
<td>Print Costs, This Listing</td>
<td>$0.29</td>
<td></td>
</tr>
<tr>
<td>Total Run Costs</td>
<td>$3.74</td>
<td></td>
</tr>
</tbody>
</table>