UIT Report Vol. 12, No. 2, pp. 225~238, 1981 연구논문집 제12권 제 2 호 pp. 225~238, 1981

The Design of a Special Language for the Description of Hangul Patterns*

Jae Jin Koh
Dept, of Computer Science
(Received April 10, 1981)

〈Abstract〉

General concept of syntactic patten recognition system is introduced in the first chapter. Then, The structural characteristics of hangul characters are discussed. The left-right scanning method is presented. The sequences of X-Y coordinates are obtained from the scanning. They are the digitized images of hangul patterns. Basic characters are extracted by using the algorithm that collects the connected points from the sequences. From the basic character images the tree is constructed. Then, the tree is tested whether it is adaptable to the tree grammar for the hangul patterns. The tree grammar for the hangul patterns is presented. The correct tree is recognized by using the tree automaton for the hangul patterns. The tree automaton for the hangul patterns is presented. The program for basic character extraction algorithm is presented.

한글 모형의 기술을 위한 특수 언어의 설계

고 제 전
전 자 계 산 학 과
(1881, 4. 10 접수)

〈요 약〉

문법적 모형인식 시스템의 일반적 개념이 첫장에 소개된다. 그 다음에 한글 문자의 구조적 특성이 논의된다. 좌에서 우로 스캐닝하는 방법이 제시되고, 그 스캐닝으로 부터 X-Y 좌표들의 수열이 구해진다. 그 수열은 한글모형의 숫자화된 이미지가 된다.

한글의 기본문자는 그 수열로 부터 연결된 점들을 모으는 알고리즘을 사용함으로써 추출된다. 그리고 그 기본문자 이미지로 부터 木構造가 구성된다. 그 木構造는 한글 모형을 기술하는 木構造문법에 적합한지 안한지 시험을 받게 된다. 정확한 木構造는 한글모형을 인식하는 木構造자동인식기에 의해서 인식이 된다. 한글을 위한 木構造 문법과 木構造 자동인식기가 제시 되었다.

I. Introduction

For the design of a special language for the description of hangul patterns, The overall explanation of syntactic pattern recognition system is required.

In the syntactic approach to pattern recognition, patterns are specified as building up out of subpatterns in various ways of composition just as phrases and sentences are built up by concatenating characters. The language which provides the structural description of patterns in terms of a set of pattern primitives and

*이 논문은 1980년도 울산공대 실험연구비에 의하여 이루어 졌음.

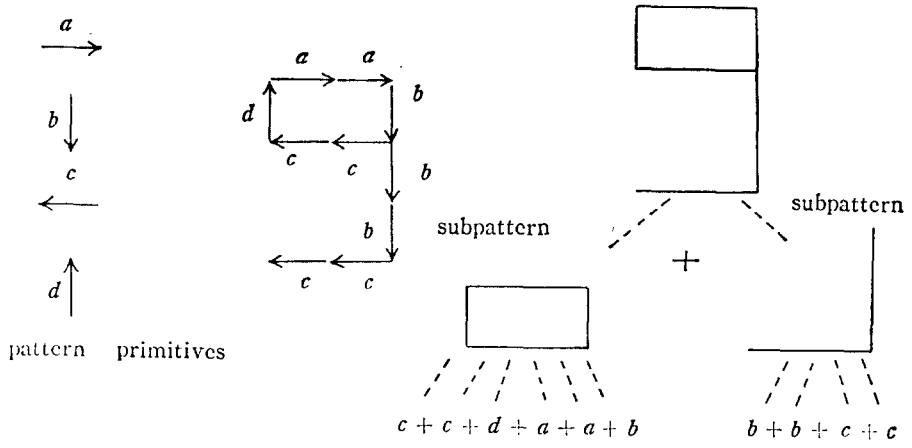


Fig. 1. pattern and its tructural description.

their composition operations, is sometimes called "pattern description language". The rules governing the composition of primitives into patterns are usually specified by the so-called "grammar" of the pattern description language. After each primitives within the pattern is identified, The recognition process is accomplished by performing a syntax analysis or parsing of the "sentence" describing the given pattern to determine whether or not it is syntactically correct with respect to the grammar. In the meantime, The syntax analysis also produces a structural description of the sentence representing the given pattern (usually in the form of a tree structure).

A syntactic pattern recognition system can be considered as consisting of three major parts; namely, preprocessing, pattern description, and syntax analysis. A simple block diagram of the system is shown in Fig. 2.

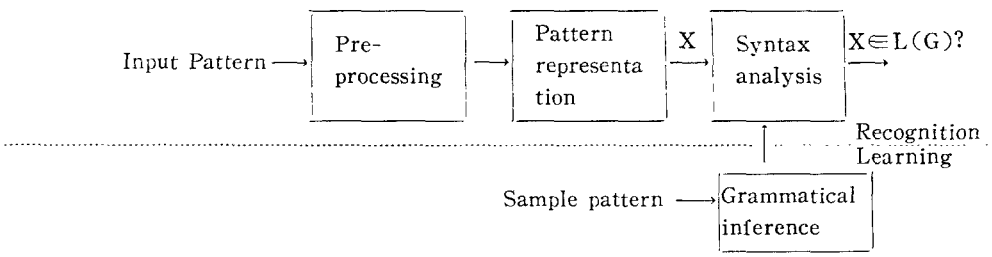


Fig. 2. Block diagram of a syntactic pattern recognition system.

The function of preprocessing include[1] pattern encoding and approximation, and[2] filtering, restoration, and enhancement. At the output of the preprocessor, presumably, we have patterns with reasonably "good quality". Each preprocessed pattern is then represented by a language-like structure (For example, a string or graph). The operation of this pattern-representation process consists of [1] pattern segmentation, and[2] primitive extraction. The decision on whether or not the representation (pattern) is syntactically correct will be performed by the "syntax analyzer". When performing the syntax analysis, the analyzer can usually produce a complete syntactic description, in terms of a parse-tree, of the pattern provided it is syntactically correct. Otherwise, the pattern is either rejected or analyzed on

the basis of other given grammars, which presumably describe other possible classes of patterns under consideration.

A set of primitives commonly used to describe boundaries or skeletons is the chain code given by Freeman. Under this scheme, a rectangular grid is overlaid on the two dimensional pattern, and straight line segments are used to connect the grid points falling closest to the pattern. Each line segment is assigned an octal digit according to its slope. The pattern is thus represented by a chain or chains of octal digits. Fig. 3. illustrates the primitives and the coded string describing a curve.

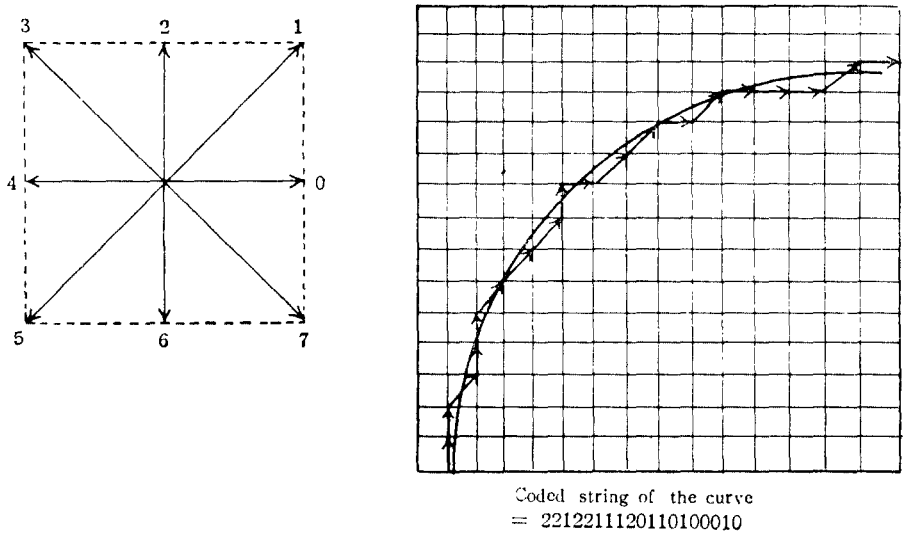


Fig. 3. Freeman's chain code

This method can be used for describing arbitrary two dimensional figures composed of straight or curved lines and line segments.

Ⅱ. The Structure of Hangul Characters

Hangul (Korean characters) consists of consonants and vowels. There are 14 consonants and 10 vowels. There are also 5 combined consonants and 4 combined vowels.

Basic consonants

ㄱㄴㄷㄹㅁㅂㅅㅇㅈㅊㅋㅌㅍㅎ

Combined consonants

ㄲㄸㅃㅆㅉ

Basic vowels

ㅏㅑㅓㅕㅗㅛㅜㅠㅡㅣ

Combined vowels

ㅐㅒㅔㅖ

Hangul phonemes consists of the first consonants phonemes and the second vowel phonemes. Optionally the third consonant phonemes can be attached to the above hangul phonemes.

Fig. 4. illustrates the phonemes structure of Hangul.

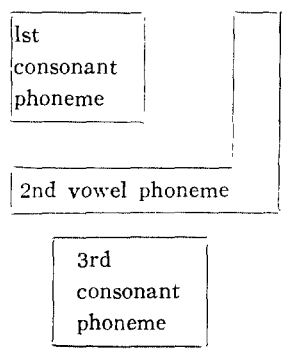


Fig. 4. The phoneme structure of Hangul

The 1st consonant phonemes of Hangul
ㄱㄴㄷㄹㅁㅂㅅㅇㅈㅊㅋㅌㅍㅎ
ㄲㄸㅃㅆㅉ
The 2nd vowel phonemes of Hangu
ㅏㅑㅓㅕㅗㅛㅜㅠㅡㅣ
ㅐㅒㅔㅖ
ㅘㅙㅚㅝㅞㅟㅢ
The 3rd consonant phonemes of Hangul
ㄱㄴㄷㄹㅁㅂㅅㅇㅈㅊㅋㅌㅍㅎ
ㄲㅆ
ㄳㄵㄶㄺㄻㄼㄽㄾㄿㅀㅄ

Ⅲ. The Scanning Method of Hangul Patterns

We scan the hangul pattern from left-top to right-bottom. This method translates hangul pattern into the (x, y) sequences of the pattern points. Fig.5. illustrates the scanning method.

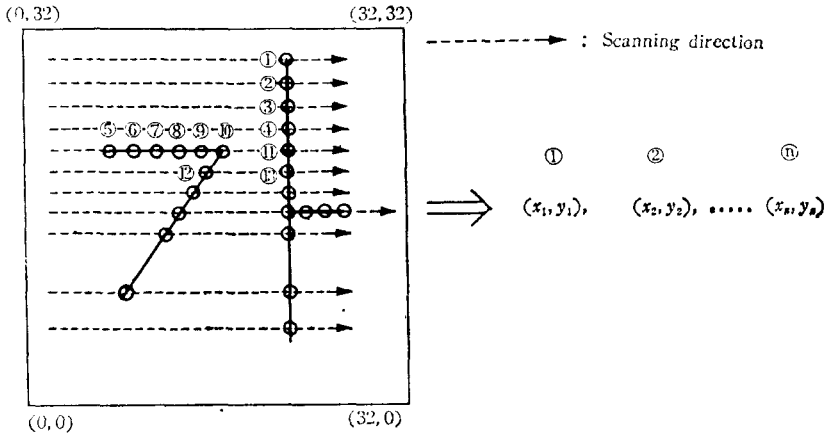


Fig. 5. The scanning method of hangul pattern

Ⅳ. The Internal Represention of Hangul Patterns

From the (x, y) coordinates sequences of the pattern points, we can extract the basic characters by comparing the points; We try to find the connected lines. By the connectedness of the basic character, we can classify the pattern into the subpatterns.

EXAMPLE.

빛$\Longrightarrow$, {ㅂ, ㅊ, ㄹ}

$G=\{(x_i, y_i) \mid i=1, 2, \cdots, n\}$

$\Longrightarrow G=G_1 \cup G_2 \cup \cdots \cup G_l$

Where G_i is a connected subpattern.

The program for the basic character extraction algorithm is attached to the appendix[1].

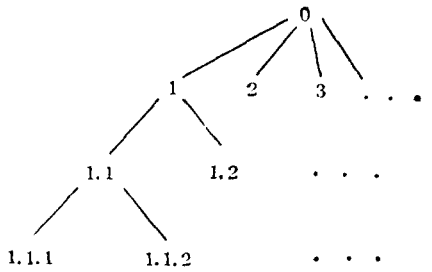
Ⅴ. The Tree Grammar for the Construction of Hangul Patterns

In the beginning we give some basic definitions on trees.

Definition 1: Let N^+ be the set of positive integers. Let U be the free monoid generated by N^+. Let. be the operation and O the identity of U. The depth of $a \in U$ is denoted by $d(a)$ and defined as follows: $d(O)=0$, $d(a.i)=d(a)+1$, $i \in N^+$. $a \leq b$ iff there exists $x \in U$ such that $a.x=b$. a and b are incomparable iff $a \not\leq b$ and $b \not\leq a$.

Example.

The partial ordering on U, called the unive rsal tree domain.



Definition 2: D is a tree domain iff D is a finite subset of U satisfying 1) $b\in D$ and $a<b$ implies $a\in D$, and 2) $a.j\in D$ and $i<j$ in N^+ implies $a.i\in D$.

Definition 3: A stratified alphabet is a pair $\langle \Sigma, r\rangle$ where Σ is a finite set of symbols and $r:\Sigma\longrightarrow N=N^+\cup\{0\}$, For $x\in\Sigma$, $r(x)$ is called stratification of x. Let $\Sigma_n=r^{-1}(n)$. A stratified alphabet is also called a ranked alphabet. A tree will be defined as a mapping from a tree domain into some set of symbols.

Definition 4: A tree over Σ (*i.e.*, over $\langle \Sigma, r\rangle$) is a function α: $D\rightarrow\Sigma$ such that D is a tree domain and $r[\alpha(a)]=\max\{i\mid a\cdot i\in D\}$. The domain of a tree will be denoted by $D(\alpha)$. Let T_Σ be the be the set of all trees over Σ.

Definition 5: A regular tree grammar over $\langle V_T, r\rangle$ is a system

$G_t=(V, r', p, S)$

satisfying the following conditions.

1) $\langle V, r'\rangle$ is a finite-ranked alphabet with $V_T\subset V$ and $r'|V_T=r$. The elements of $V-V_T$ are called nonterminal symbols.

2) p is a finite set of productions of the form $\phi\rightarrow\phi$, where ϕ and ϕ are trees over $\langle V, r'\rangle$.

3) S is a finite subset of T_V whose elements are called the axioms of G_t, where T_V is the set of trees over alphabet V.

Definition 6: $L(G_t)=\{\alpha\in T_{V_T}\mid$there exists $Y\in S$ such that $Y\Rightarrow\alpha$ in $G_t\}$ is called the language generated by G_t.

Definition 7: A tree grammar $G_t=(V, r', p, S)$ over $\langle V_T, r\rangle$ is expansive iff each production in p is of the form

$X_0\longrightarrow x$ or $X_0\longrightarrow x$ where $x\in V_T$ and

$X_1\ldots X_{r(x)}$

$X_0, X_1, \cdots, X_{r(x)}$ are nonterminal symbols.

Theorem 1: For each regular tree grammar $G_t=(V, r, p, S)$ over V_T one can effectively construct an equivalent expansive grammar.

Example: Tree representations of Hangul basic characters. The primitives are Freeman's '8 octal digits chain code and 3 pseudoconcatenation operators. Fig. 6. illustrates the 3 pseudoconcatenation operators.

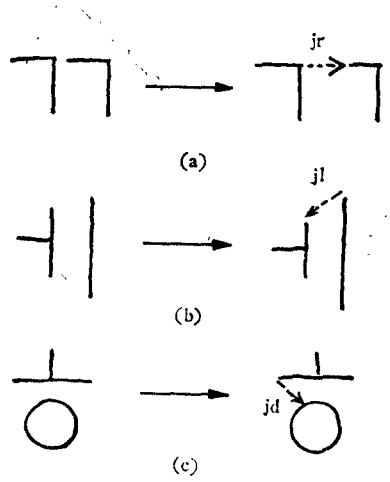
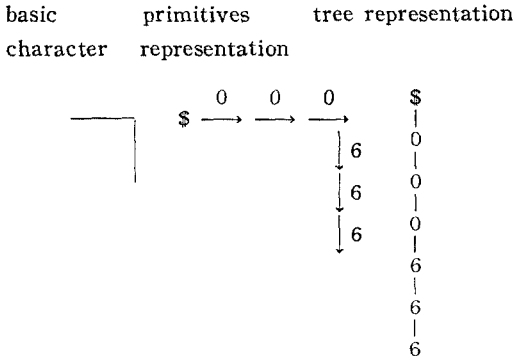
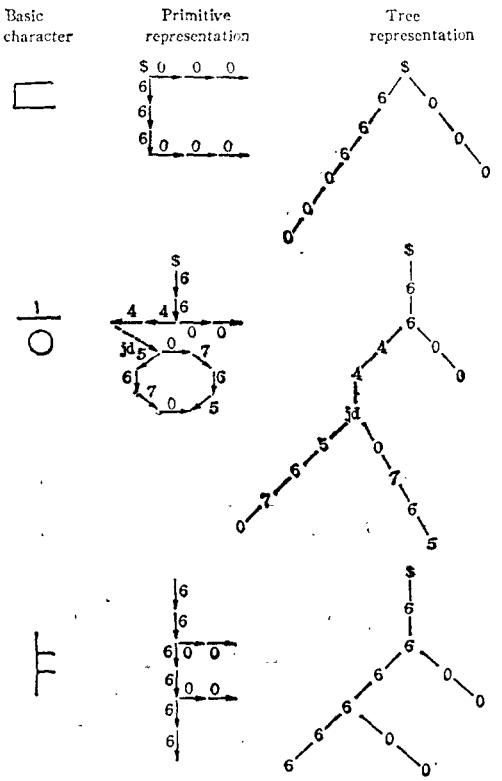


Fig. 6. Terminal symbols of pseudo-operators.
(a) jr(jump right) (b) jl(jump left)
(c) jd(jump down)



Where $ stands for start primitive.



Example: Tree grammar for the construction of Hangul basic characters(ㄱㄴㄷㄹㅁㅂㅅ).

$G_t=(V, r, p, S)$

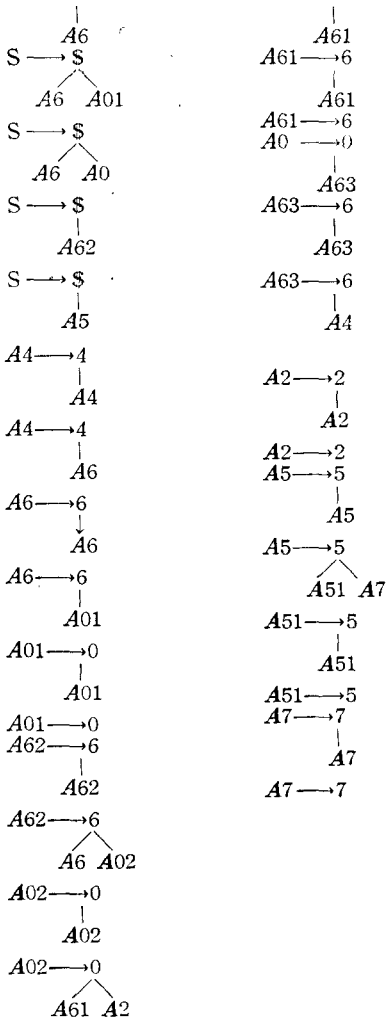
Where $V=\{S, A0, A01, A02, A2, A4, A5, A51, A6, A61, A62, A63, A7, 0, 2, 4, 5, 6, 7, \$\}$

and $V_T=$ { S : (starting point); 0 : ⟶; 2 : ↑; 4 : ⟵; 5 : ↙; 6 : ↓; 7 : ↘ }

$r(0)=\{0,1,2\}$, $r(2)=\{0,1\}$, $r(4)=\{1\}$, $r(5)=\{0,1,2\}$, $r(6)=\{0,1,2\}$, $r(7)=\{0,1\}$

$|0|=|2|=|4|=|5|=|6|=|7|$

P : S ⟶ $ (A0)
S ⟶ $
A0 ⟶ 0 (A0)
A0 ⟶ 0



Ⅲ. The Tree Automaton for the Recognition of Hangul Pattens

We introduce some basic definitions on tree automaton.

Definition 8: Let (Σ, r) be a stratified alphabet and $\Sigma=\{\sigma_1, \sigma_2, \cdots, \sigma_k\}$. A finite Σ-automaton or tree automaton over Σ is a system $M_t=(Q, f_1, \cdots, f_K, F)$ Where 1) Q is a finite set of state.

2) For each i, $1\leq i\leq k$, f_i is a relation on $Q^{r(\sigma i)}$ X Q ; and

3) $F\subset Q$ is a set of final states.

If each f_i, $1 \leq i \leq k$, is a function, $f_i : Q^{r(\sigma i)} \longrightarrow Q$, then M_t is deterministic; Otherwise M_t is nondeterministic.

Definition 9 : $T(M_t) = \{\alpha \in T_\Sigma \mid$ there exists $X \in F$ such that $\rho(\alpha) \sim X\}$ is called the language accepted by M_t. A tree grammar G_t is equivalent to a tree automaton M_t if $L(G_t) = T(M_t)$.

Theorem 2 : For every regular tree grammar G_t, one can effectively construct a deterministic tree automaton M_t such that $T(M_t) = L(G_t)$

Example: The tree automaton corresponding to the preceding tree grammar accepting the hangul basic character (ㄱ, ㄴ, ㄷ, ㄹ, ㅁ, ㅂ, ㅅ) is as follows.

$M_t = \{Q, f0, f2, f4, f5, f6, f7, f\$, F\}$

Where $Q = \{q0, q2, q5, q6, q7, q06, q60, q064, q0646, q06460, q602, q\alpha, q\$, , q$ㅂ q_β qㄱ, qㄴ, qㄷ, qㄹ, qㅁ, qㅂ, qㅅ$\}$

f : $f0 = q0$ $f2 = q2$ $f5 = q5$ $f6 = q6$ $f7 = q7$ $f0(q0) = q0$ $f6(q0) = q06$ $f6(q06) = q06$ $f\$(q06) = q$ㄴ $f6(q6) = q6$ $f0(q6) = q60$ $f0(q60) = q60$ $f\$(q60) = q$ㄱ $f\$(q06, q0) = q$ㄷ $f\$(q06, q60) = q$ㅁ $f4(q06) = q064$ $f4(q064) = q064$ $f6(q064) = q0646$ $f6(q0646) = q0646$ $f0(q0646) = q06460$ $f0(q06460) = q06460$ $f\$(q06460) = q$ㄹ $f5(q5) = q5$ $f7(q7) = q7$ $f5(q5, q7) = q_\beta$ $f5(q_\beta) = q_\beta$ $f2(q2) = q2$ $f0(q6, q2) = q602$ $f0(q602) = q602$ $f6(q06, q602) = q_\alpha$ $f6(q_\alpha) = q_\alpha$ $f\$(q_\alpha) = q$ㅂ $f\$(q_\beta) = q$ㅅ $Q = \{q$ㄱ, qㄴ, qㄷ, qㄹ, qㅁ, qㅂ, qㅅ$\}$

Ⅶ. Conclusion

For the design of a special language with respect to Hangul patterns, we take the approach of syntactic pattern recognition to it. Basically Hangul can be represented by the graph structure. therefore tree system as a special case of graph system was adopted as an analyzing and a recognition system. Actually Hangul can be represented correctly by trees. The tree grammar for the construction of Hangul basic characters was developed. And then, the tree automaton for the recognition of Hangul basic characters was developed. The left-top to right-bottom scanning method was developed for the scanning of Hangul patterns. This method contributes to the automatic Hangul pattern recognition system. Finally we thanks to the sponsor (college project manager) and the laborotory which provides the testing instruments to us.

References

1. P.W. Becker, An Introduction to The Design of Pattern Recognition Device, New York: Springer, 1971.
2. K.S. Fu and B.K. Bhargava, "Tree Systems for Syntactic Pattern Recognition", IEEE Trans. on Comp., Vol. C—22, Dec., 1973.
3. T. Agui, M. Nakajima, T.K. Kim, and E.T. Takahashi, "A Method of Recognition and Representation of Korean Characters by Tree Grammars", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-1, NO. 3, July 1979.
4. K.S. Fu, Digital Pattern Recognition, New York: Springer, 1976.
5. B. Moyyer and K.S. Fu, "A Tree System Approach for Fingerprint Pattern Recognition", IEEE Trans. on Comp., Vol. C—25, NO.3, March 1976.

Appendix (1)

The program for the basic character extraction algorithm

INPUT SOURCE DATA

12	17
5	16
8	16
12	16
5	15
8	15
10	15
11	15
12	15
5	14
6	14
7	14
8	14
12	14
5	13
8	13
10	13
11	13
12	13
5	12
6	12
7	12
8	12
12	12
12	11
6	9
7	9
8	9
9	9
10	9
11	9
12	9
12	8
12	7
7	6
8	6
9	6
10	6

```
        11      6
        12      6
         7      5
         7      4
         7      3
         7      2
         8      2
         9      2
        10      2
        11      2
        12      2

DOS FORTRAN IV 360N-FO-479 3-8  FIQ  DATE 11/04/80 TIME 13.27.39 PAGE 0001
0001                SUBROUTINE FIQ (IB, IP1)
0002                DIMENSION IB(2,100), IP1(20,20)
0003                DATA MARK1, MARK 2/1H*, 1H /
0004                COMMON IP(20,20)
0005                IF (IB(1,1).EQ.0) GO TO 99
0006                DO 6 I=1,20
0007                DO 6 J=1,20
0008              6 IP1(I,J)=MARK2
0009                DO 1 I=1,100
0010                IF (IB(1,I).EQ.0) GO TO 2
0011                NK=IB(1,I)
0012                MK=IB(2,I)
0013                MK=20-MK
0014              1 IP1(MK, NK)=MARK1
0015              2 DO 3 I=1,20
0015
0016                IP1 (1,I)=MARK1
0017                IP1(20,I)=MARK1
0018                IP1(I,1)=MARK1
0019              3 IP1(I,20)=MARK1
0020                DO 20 I=1,20
0021                DO 20 J=1,20
0022             20 IP (I,J)=IP1(I,J)
0023                WRITE (3,4) ((IP1(I,J),J=1,20), I=1,20)
0024              4 FORMAT (20(1X,A1))
0025             99 RETURN
0026                END

DOS FORTRAN IV 360N-FO-479 3-8 SORTQ DATE 11/04/80 TIME13.27.06 PAGE 0001
0001                SUBROUTINE SORTQ (IA)
0002                DIMENSION IA(2,100)
```

```
0003          COMMON KM (2,400), N
0004          IP=0
0005          DO 1 I=1,2
0006          DO 1 J=1,100
0007        1 IA (I,J)=0
0008          K=1
0009          DO 2 I=1,N
0010          IF (KM(1,I).EQ.99) GO TO 2
0011          IA(1,K)=KM(1,I)
0012          IA(2,K)=KM(2,I)
0013          KM(1,I)=99
0014          KM(2,I)=99
0015          M=I+1
0016          GO TO 3
0017        2 CONTINUE
0018          GO TO 999
0019        3 CONTINUE
0020          DO 4 I=M,N
0021          IF (KM(1,I).EQ.99) GO TO 4
0022          DO 5 J=1,K
0023          IF (IABS(KM(1,I)-IA(1,J)).GT.1.OR.IABS (KM(2,1)-IA(2,J))
              +GT.1) GO TO 5
0024          GO TO 6
0025        5 CONTINUE
0026          GO TO 4
0027        6 K=K+1
0028          IA (1,K)=KM(1,I)
0029          IA (2,K)=KM(2,I)
0030          KM (1,I)=99
0031          KM(2,I)=99
0032        4 CONTINUE
0033          DO 7 J=M,N
0034          I=N-J+M
0035          IF(KM(1,I).EQ.99) GO TO 7
0036          DO 8 L=1,K
0037          IF (IABS(KM(1,I)-IA(1,L)).GI.1.OR.IABS(KM(2,I)-IA(2,L))
              +GT.1) GO TO 8
0038          GO TO 9
0039        8 CONTINUE
0040          GO TO 7
0041        9 K=K+1
0042          IA(1,K)=KM(1,I)
0043          IA(2,K)=KM(2,I)
```

```
0044                KM(1,I)=99
0045                KM(2,I)=99
0046              7 CONTINUE
0047                IP=IP+1
0048                IF(IP.EQ.1) GO TO 3
0049            999 RETURN
0050                END
```

DOS FORTRAN IV, 360N-FO-479 3—8 MAINPGM DATE 11/04/80 TIME 13.25.21 PAGE 0001

```
0001                DIMENSION NA(2,100), NB(2,100), NO(2,100),ND(2,100),NE(2,100),
                    1NF(2,100), IQ(6)
0002                DIMENSION FA(20,20), FB(20,20),FC(20,20), FD(20,20), FE(20,20),
                    +FF(20,20)
0003                COMMON KM(2,400),N,IP(20,20)
0004             99 READ(1,2)N
0005                IF(N.EQ.99999) GO TO 999
0006                DO 1 I=1,2
0007                DO 1 J=1,N
0008              1 KM(I,J)=99
0009              2 FORMAT (I5)
0010                WRITE(3,21)
0011             21 FORMAT (1H1,10X, '*** INPUT SOURCE DATA***')
0012                DO 3 I=1,N
0013                READ(1,4) (KM(J,I),J=1,2)
0014              3 WRITE(3,22) (KM(J,I),J=1,2)
0015             22 FORMAT (13×,2I7)
0016              4 FORMAT (2I5)
0017                CALL SORTQ (NA)
0018                CALL SORTQ (NB)
0019                CALL SORTQ (NC)
0020                CALL SORTQ (ND)
0021                CALL SORTQ (NE)
0022                CALL SORTQ (NF)
0023                WRITE (3,5)N
0024              5 FORMAT (1H1,10×, '1.*DATA COUNTER **',I5,///)
0025                DO 10 I=1,6
0026             10 IQ (I)=I
0027                WRITF(3,8)(IQ(I),I=1,6)
0028              8 FORMAT (5×,6('D-SORT',I1,6×))
0029                DO 6 J=1,N
0030              6 WRITE(3,7) (NA(I,J),I=1,2),(NB(I,J),I=1,2),(NC(I,J),I=1,2),
                    +(ND(I,J),I=1,2),(NE(I,J),I=1,2),(NF(I,J),I=1,2)
0031              7 FORMAT (5X,6(I3,2X,I3,5X))
0032                WRITE(3,20)
```

```
   20 FORMAT (1H1,4X,'GRAPHING SORTED DATA')
      CALL FIQ(NA,FA)
      CALL FIQ(NB,FB)
      CALL FIQ(NC,FC)
      CALL FIQ(ND,FD)
      CALL FIQ(NE,FE)
      CALL FIQ(NF,FF)
      GO TO 99
  989 STOP
      END
```

1, *DATA COUNTER** 49

D−SORT1		D−SORT2		D−SORT		D−SORT4		D−SORT5		D−SORT6	
12	17	5	16	6	9	0	0	0	0	0	0
12	16	5	15	7	9	0	0	0	0	0	0
11	15	5	14	8	9	0	0	0	0	0	0
12	15	6	14	9	9	0	0	0	0	0	0
12	14	7	14	10	9	0	0	0	0	0	0
11	13	8	14	11	9	0	0	0	0	0	0
12	13	5	13	12	9	0	0	0	0	0	0
12	12	8	13	12	8	0	0	0	0	0	0
12	11	5	12	12	7	0	0	0	0	0	0
10	13	6	12	11	6	0	0	0	0	0	0
10	15	7	12	12	6	0	0	0	0	0	0
0	0	8	12	10	6	0	0	0	0	0	0
0	0	8	15	9	6	0	0	0	0	0	0
0	0	8	16	8	6	0	0	0	0	0	0
0	0	0	0	7	6	0	0	0	0	0	0
0	0	0	0	7	5	0	0	0	0	0	0
0	0	0	0	7	4	0	0	0	0	0	0
0	0	0	0	7	3	0	0	0	0	0	0
0	0	0	0	7	2	0	0	0	0	0	0
0	0	0	0	8	2	0	0	0	0	0	0
0	0	0	0	9	2	0	0	0	0	0	0
0	0	0	0	10	2	0	0	0	0	0	0
0	0	0	0	11	2	0	0	0	0	0	0
0	0	0	0	12	2	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0

GRAPHING SORTED DATA