Ein verbessertes Verfahren zur Feststellung der Isomorphie endlicher Graphen

Hinteregger, Josef

1976

5. Kapitel. Programmierung des Verfahrens zur Konstruktion der disjunkten Basis von H(..,G1,...,Gk). Zeitabschätzung

urn:nbn:at:at-ubi:2-12744

5. Kapitel. Programmierung des Verfahrens zur Konstruktion der disjunkten Basis von $H(\Delta,G_1,..,G_k)$. Zeitabschätzung.

Zunächst werden einige Abänderungen des Verfahrensschrittes A (Definition 4.1) durchgeführt, die zwar das Ergebnis (die Konstruktion der disjunkten Basis) nicht beeinträchtigen, in manchen Fällen das Verfahren jedoch beschleunigen.

5.1 Definition

Verfahrensschritt A' bezeichne die Abbildungsvorschrift, die aus dem Verfahrensschritt A von 4.1 dadurch entsteht, daß dort die Abbildungen f_{st} durch die Abbildungen

$$f_{rst} \quad \left(f_{rst}(x,y) := \begin{array}{l}|\{z\in X \mid f(z,z)=r \wedge f(x,z)=s \wedge f(y,z)=t\}| \\ \text{für } r\in f(\Delta),\ s,t\in \operatorname{Im} f\end{array}\right)$$

ersetzt werden:

$A'(f) := g \circ (f,..,f_{rst},..)$, g Anordnung von $\operatorname{Im}(f..f_{rst}..)$.

5.2 Hilfssatz

Sei $f \in F$ (von 4.1), $s,t \in \operatorname{Im} f$.

Dann gilt: $$f_{st} = \sum_{r\in f(\Delta)} f_{rst} .$$

Beweis $M_{st}(x,y) := \{z\in X \mid f(x,z)=s \wedge f(y,z)=t\}$

$M_{rst}(x,y) := \{z\in X \mid f(x,z)=s \wedge f(y,z)=t \wedge f(z,z)=r\}$.

$M_{st}(x,y) = \bigcup_{r\in f(\Delta)} M_{rst}(x,y)$ für $x,y \in X$:

$r\in f(\Delta) \Rightarrow M_{rst}(x,y) \subset M_{st}(x,y)$ klar.

$z \in M_{st}(x,y)$, $r_o := f(z,z) \Rightarrow z \in M_{r_o st}(x,y)$.

Für $r_1,r_2 \in f(\Delta)$ gilt: $M_{r_1 st}(x,y) \cap M_{r_2 st}(x,y) \neq \emptyset \Rightarrow$

$\Rightarrow r_1 = f(z,z) = r_2$, also $M_{rst}(x,y)$ disjunkt für festes s,t.

$$\Rightarrow f_{st}(x,y) = |M_{st}(x,y)| = \sum_{r\in f(\Delta)} |M_{rst}(x,y)| =$$

$$= \sum_{r\in f(\Delta)} f_{rst}(x,y)$$

□

5.3 Folgerung

$\forall f \in F$ (siehe 4.1): $A'(f) \leq A(f) \leq f$.

Beweis

$A'(f)(x,y) = A'(f)(a,b) \Rightarrow$

$\forall s,t \in \mathrm{Im}\, f, \forall r \in f(\Delta)$: $f_{rst}(x,y) = f_{rst}(a,b)$, $f(x,y)=f(a,b)$

$\Rightarrow \forall s,t \in \mathrm{Im}\, f$: $\underline{f_{st}(x,y)} = \sum_{r\in f(\Delta)} f_{rst}(x,y) =$

$= \sum_{r\in f(\Delta)} f_{rst}(a,b) = \underline{f_{st}(a,b)}$,

$f(x,y) = f(a,b)$,

also $A(f)(x,y) = A(f)(a,b)$.

$A(f) \leq f$ nach 4.2.1 .

5.4 Hilfssatz

Sei $f \in F$ mit $f \leq \chi_\Delta$, d.h. $f(x,y)=f(a,b) \Rightarrow \left[x=y \Leftrightarrow a=b\right]$,

$r \in f(\Delta)$, $s,t \in \mathrm{Im}\, f$, $j \in \mathbb{N}$, $0 \leq j \leq n = |X|$.

Dann gilt: $f^{-1}_{rst}(j) = \left[f^{-1}(s)\ (1)\ f^{-1}(r)\right]\ (j)\ \left[f^{-1}(t)\right]^T$.

Beweis

$B(x,z) := \{z' \in X \mid f(x,z)=s \wedge f(z,z')=r\}$.

Es gilt: $z' \in B(x,z) \Rightarrow f(z,z') = r \Rightarrow (r\in f(\Delta), f\leq\chi_\Delta) \Rightarrow z=z'$,

also $B(x,z) = \emptyset$ oder $B(x,z) = \{z\}$.

$f_{rst}(x,y) = j \Leftrightarrow j = |\{z \in X \mid f(z,z)=r \wedge f(x,z)=s \wedge f(y,z)=t\}| =$

$(\ f(z,z)=r \wedge f(x,z)=s \Leftrightarrow B(x,z)=\{z\} \Leftrightarrow |B(x,z)|=1\ )$

$= |\{z\in X \mid |B(x,z)| = 1 \wedge f(y,z) = t\}| \Leftrightarrow$

$(\ |B(x,z)| = 1 \Leftrightarrow (x,z) \in f^{-1}(s)\ (1)\ f^{-1}(r)\ )$

$\Leftrightarrow$ $(x,y) \in$ rechte Seite.

Der Verfahrensschritt A' wird jetzt in zwei Schritte (A_1, A_2) unterteilt $(A' = A_2 \circ A_1)$, nämlich in die Berechnung von $(f,..,f_{rst},..)$ für die Elemente (x,x) der Diagonale, A_1, und für die übrigen Elemente (x,y) mit $x \neq y$, A_2:

$$h_1(x,y) := \begin{cases} (f,..,f_{rst},..)(x,v) & \text{für } x=y \\ f(x,y) & x\neq y \end{cases}$$

$A_1(f) := g_1 \circ h_1$, wobei g_1 Anordnung von Im h_1.

$$h_2(x,y) := \begin{cases} f(x,y) & \text{für } x=y \\ (f,..,f_{rst},..)(x,y) & x\neq y \end{cases}$$

$A_2(f) := g_2 \circ h_2$, g_2 Anordnung von Im h_2.

A_1 entspricht der Anwendung von "extended features" bei Unger [1o] und Levi [5] , dem "Algoritmo di affinamento" bei Sirovich [7].
Bei vielen "unregelmäßigen" Graphen genügt bereits die iterierte Anwendung von A_1 auf die durch den Grad des gegebenen Graphen erzeugte Partition der Diagonale zur Bestimmung der Automorphismenpartition der Knoten.
Für spezielle Graphen, nämlich für Arboreszenzen und zusammenhängende funktionale Graphen hat Tinhofer in [9] bewiesen, daß dadurch die Automorphismenpartition erhalten wird.
Da A_1 außerdem nur n Berechnungen von Funktionswerten $(f,..,f_{rst},..)(x,x)$ erfordert gegenüber $n.(n-1)$ bei A_2, definieren wir den folgenden Verfahrensschritt A":

5.5 Definition
"Verfahrensschritt A"" bezeichne die Abbildung: $A" := A_2 \circ A_1^n$, d.h. zuerst wird A_1 sooft durchgeführt, bis die Diagonale nicht mehr weiter verfeinert wird(nach Satz 3.11 ist dies nach spätestens n-mal der Fall), und dann A_2.

Der "Verfahrensschritt A" führt ebenfalls zur disjunkten Basis von $H(\Delta,G_1,..,G_k)$:

5.6 Satz
Voraussetzungen wie 4.4 . $f' := (A")^{n^2}(f^{(o)})$.
Dann gilt: $f' \equiv f$.

Beweis

$f' \leq f$: durch Induktion.

$A''(f^{(o)}) = A_2 o A_1 o A_1^{n-1}(f^{(o)}) = A' \circ A_1^{n-1}(f^{(o)}) \overset{(5.3)}{\leq}$

$A \circ A_1^{n-1}(f^{(o)})$. 5.3 gilt auch für A_1: $A_1(f) \leq f$:

$A \circ A_1^{n-1}(f^{(o)}) \overset{(4.2.2)}{\leq} A(f^{(o)})$.

Sei jetzt $(A'')^i(f^{(o)}) \leq A^i(f^{(o)})$.

$(A'')^{i+1}(f^{(o)}) = A' \circ A_1^{n-1} \circ (A'')^i(f^{(o)}) \overset{(5.3)}{\leq}$

$\leq A \circ A_1^{n-1} \circ (A'')^i(f^{(o)}) \leq$

$\left[A_1^{n-1} \circ (A'')^i(f^{(o)}) \overset{(5.3)}{\leq} (A'')^i(f^{(o)}) \overset{(I.A.)}{\leq} A^i(f^{(o)})\right]$

$\leq A^{i+1}(f^{(o)})$.

$f \leq f'$:

Sei $M' := \left\{M'_j\right\}_{j \in \mathrm{Im}\, f'}$, $M'_j := f'^{-1}(j)$.

Zeige: $M' \subset H(\Delta, G_1, .., G_k)$.

genau wie in Teil I des Beweises von 4.4: $M^{(o)}$ bleibt gleich;
in Hilfssatz 4.5 ist $(r, r_{m_i m_i}, .., r_{11}) := (f^{(i)}, .., f^{(i)}, .)(a,b)$
zu ersetzen durch $(m_o, .., m_{rst}, ..) := (f'^{(i)}, .., f'^{(i)}_{rst}, ..)(a,b)$,
weiters $M_s^{(i)}(r_{st}) M_t^{(i)T}$ durch:
$\left[M_s^{(i)}\ (1)\ M_r^{(i)}\right](m_{rst})\left[M_t^{(i)}\right]^T$, wobei jetzt der Durchschnitt
über die $r \in f(\Delta)$, $s,t \in \mathrm{Im}\, f'^{(i)}$ zu nehmen ist, und
$M_r^{(i)}$ durch $M_{m_o}^{(i)}$ zu ersetzen. (Siehe auch Hilfssatz 5.4).
Die Aufteilung von A' in A_1 und A_2 ändert hier nichts, die
Formel gilt dann eben nur für die $l \in f'^{(i+1)}(\Delta)$
(bzw. $f'^{(i+1)}(X \times X \setminus \Delta)$), für die übrigen l haben sich die
$M_l^{(i+1)}$ nicht geändert, d.h. es gibt ein l_1 mit $M_{l_1}^{(i)} = M_l^{(i+1)}$.
Induktionsschluß wie in Teil I) des Beweises 4.4:
$M' \subset H(\Delta, G_1, .., G_k)$.

also $M' \subset H(\Delta, G_1, .., G_k)$:

$\forall\, j \in \mathrm{Im}\, f'\ \exists\, I_j \in \mathrm{Im}\, f : \quad M'_j = \bigcup_{s \in I} M_s$.

Seien $x,y,a,b \in X$, $j := f'(x,y)$:

$f(x,y) = f(a,b) =: s \Rightarrow s \in I_j$, $\underline{(a,b) \in M_s \subset M'_j} \Rightarrow$

$\Rightarrow f'(a,b) = j = f'(x,v)$,

also $f \leq f'$ und somit $f \equiv f'$.

5.7 Hilfssatz

Hilfssatz 4.6 gilt auch für A_1 und A_2, d.h.
falls $f'^{(i+1)} = A_1(f'^{(i)})$ und falls $f'^{(i+1)} = A_2(f'^{(i)})$.

Beweis für A_2 wie bei 4.6 .

$\left[M_s^{(i)}\ (1)\ M_r^{(i)}\right]\ (m)\ \left[M_t^{(i)}\right]^T = \left[M_t^{(i)}\ (1)\ M_r^{(i)}\right]\ (m)\ \left[M_s^{(i)}\right]^T$,

leicht zu sehen mit Hilfssatz 5.4 .

für A_1 durch Induktion:

für $f^{(o)}$ wie in 4.6. Behauptung gelte für i:

$M_1^{(i+1)}$, $1 \in f^{(i+1)}(\Delta)$: $M_1^{(i+1)} \subset \Delta$, $M_1^{(i+1)T} = M_1^{(i+1)}$.

$1 \notin f^{(i+1)}(\Delta)$: $\exists\, 1_1 \in \mathrm{Im}\, f^{(i)}$: $M_{1_1}^{(i)} = M_1^{(i+1)}$; nach I.A.

oder weil $f^{(i)} = A_2(f^{(i-1)})$: $\exists\, 1'_1 \in \mathrm{Im}\, f^{(i)}$: $M_{1_1}^{(i)\,T} = M_{1'_1}^{(i)}$

$1' := g_i(1'_1) \notin f^{(i+1)}(\Delta)$: $M_{1'_1}^{(i)} = M_{1'}^{(i+1)} =$

$= M_1^{(i+1)T}$ □

Als Anordnung g_i in 4.1 und g_o in 4.4 wird folgende Vorschrift verwendet:

Zuerst wird $\mathrm{Im}(f^{(i)}, .., f_{rst}^{(i)}, ..)$ lexikographisch angeordnet. (im Programm Zeile 642-658 für A_2, 5o7-526 für A_1),
dann werden diejenigen Tupel $(f^{(i)}, .., f_{rst}^{(i)}, ..)(x,y)$, welche lexikographisch größer sind als $(f^{(i)}, .., f_{rst}^{(i)}, ..)(y,x)$, unmittelbar hinter $(f^{(i)}, .., f_{rst}^{(i)}, ..)(y,x)$ eingefügt. (Zeile 746-751). Nach Hilfssatz 5.6 ist diese zweite Vor-

schrift unabhängig von der Wahl von (x,y). Falls $M_j^{(i)T} \neq M_j^{(i)}$ ist, gilt dann aufgrund dieser Anordnung: $M_j^{(i)T} = M_{j+1}^{(i)}$. Hilfssatz 5.6(4.6) erspart dann einige Berechnungen von $(f^{(i)},..,f_{rst}^{(i)},..)(y,x)$:

Falls $M_j^{(i)}$ aufgeteilt wurde auf $M_{j_1}^{(i+1)},..,M_{j_r}^{(i+1)}$, so wird auch $M_j^{(i)}$ aufgeteilt auf $M_{j'_1}^{(i+1)},..,M_{j'_r}^{(i+1)}$;

für die $(x,y) \in M_j^{(i)T}$ braucht also $(f^{(i)},..,f_{rst}^{(i)},..)(x,y)$ nicht mehr berechnet zu werden. (Zeile 708-723, bzw. 680-692 für r=1).

Eine Einsparung von Berechnungen von $f_{rst}^{(i)}(x,y)$ bringt auch folgende Tatsache:

Falls für ein $r \in f^{(i+1)}(\Delta)$ und für $s,t \in \text{Im } f^{(i+1)}$ gilt:

$\exists\, r' \in f^{(i)}(\Delta)$, $s',t' \in \text{Im } f^{(i)}$ mit $M_r^{(i+1)}=M_{r'}^{(i)}$, $M_s^{(i+1)}=M_{s'}^{(i)}$ und $M_t^{(i+1)}=M_{t'}^{(i)}$ ($M_{r'}^{(i)}$, $M_{s'}^{(i)}$ und $M_{t'}^{(i)}$ wurden durch Schritt i+1 nicht aufgeteilt),

so ist für $x,y \in X$: $f_{rst}^{(i+1)}(x,y) = f_{r's't'}^{(i)}(x,y)$

und wegen $f^{(i+1)} \leq f_{r's't'}^{(i)}$ braucht $f_{rst}^{(i+1)}$ nicht berechnet zu werden.

Für $M_j^{(i+1)}$ wird die Eigenschaft: $\exists\, j' \in \text{Im } f^{(i)}$: $M_j^{(i+1)}=M_{j'}^{(i)}$ im Attribut "ALT" der Partitionsmenge festgehalten. (Zeile 167) ("GANZ ALT" gibt an, ob die Partitions-menge $M \subset \Delta$ seit der letzten Anwendung von A_2 einmal aufgeteilt wurde)

5.8 Abschätzung der vom Verfahren benötigten Rechenzeit

1) Zeitaufwand zur Berechnung von

$(f,..,f_{rst},..)(x,y)$ für ein $(x,y) \in X \times X$:

Für höchstens n Tripel (r_j,s_j,t_j) $(r_j \in f(\Delta), s,t \in \mathrm{Im}\, f)$

ist $m_{r_j s_j t_j} := f_{r_j s_j t_j}(x,y)$ von Null verschieden, denn

$$\sum_{r \in f(\Delta), s,t \in \mathrm{Im}\, f} f_{rst}(x,y) = |X| = n.$$

$(f,..,f_{rst},..)(x,y)$ wird daher als Liste von lexikografisch absteigend angeordneten Quadrupeln $(r_j,s_j,t_j,m_{r_j s_j t_j})$, wobei $m_{r_j s_j t_j} \neq o$ ist, abgespeichert.

Die Berechnung der $m_{r_j s_j t_j} \neq o$ geschieht nach folgender Methode:

```
integer x,y,z,r,s,t;
for z := 1 step 1 until n do
begin      r := f(z,z);    s := f(x,z);    t := f(y,z);
     [*]  if ("Zähler (r,s,t,m_rst) in der Quadrupelliste
              vorhanden") then m_rst := m_rst + 1
          else ("Füge neuen Zähler (r,s,t,1) an der
                richtigen Stelle in die Quadrupel-
                liste ein");
end;
```

Diese Berechnung der m_{rst} für ein (x,y) erfolgt innerhalb der procedure G(K), die zur class GRAPH gehört. (Zeile 22o)

Bei $z = j$ muß r,s,t mit den höchstens j r',s',t' der bereits vorhandenen Quadrupel verglichen werden:
die Berechnung von $(f,..,f_{rst},..)(x,y)$ erfordert höchstens
$\sum_{j=1}^{n} j \simeq \underline{n^2}$ Schritte.

2) Zeitaufwand zur Anordnung der m Listen $(f,..,f_{rst},..)(x,y)$
$\mathrm{Im}\, f$

Für das Einordnen der j-ten Liste sind höchstens j Vergleiche

mit bereits errechneten Listen nötig, ein solcher Vergleich erfordert den Vergleich von höchstens n Quadrupeln:

Die Anordnung von $Im(f,..,f_{rst},..)$ erfordert höchstens $\sum_{j=1}^{m} j \cdot n$ Schritte.

Daraus ergibt sich der

3) Zeitaufwand für einen Schritt A_1 (Partition von X):

n Berechnungen von $(f,..,f_{rst},..)(x,y)$ $\quad \simeq n^3$

Anordnung von höchstens n Listen: $\sum_{j=1}^{n} j \cdot n \quad \simeq n^3$

mit t_1 proportional n^3

und der

4) Zeitaufwand für einen Schritt A_2 (Partition von X×X):

$\sim n^2$ Berechnungen von $(f,..,f_{rst},..)(x,y)$: $\quad \simeq n^4$

Anordnung von höchstens n^2 Listen $\quad \sum_{j=1}^{n^2} j \cdot n \quad \simeq n^5$

mit t_2 proportional n^5.

Die Anzahl der Durchführungen von A_1 ist höchstens gleich n (3.11). Für A_2 ist dieselbe Abschätzung nach (3.11) n^2; in den praktisch berechneten Beispielen mit n zwischen 1o und 2o brauchte A_2 jedoch höchstens fünfmal durchgeführt zu werden.

Die Rechenzeit des Verfahrens ist somit theoretisch maximal der siebten Potenz von n proportional; die praktisch erprobten Werte liegen jedoch deutlich darunter.

Die am häufigsten durchlaufenen Unterprogramme wurden in Assembler-Sprache programmiert:
INCM1 (Zeile 142) führt die Anweisung [*] durch,
COMPARE (139) vergleicht zwei Listen von ganzen Zahlen,
COMPR(qliste,L) (145) speichert die Quadrupelliste qliste als Liste von ganzen Zahlen in einen externen Speicherbereich(außerhalb des SIMULA-Programms) von L.I1 bis L.I2 ab, und SETLU(I) (148) ermöglicht es, derartigen Speicherbereich wieder freizugeben.

5.9 Programm

Da ein Programm für eine spezielle Anwendung natürlich auf die dort vorkommenden Graphen ausgerichtet sein muß, wurde hier versucht, mehr Wert auf Übersichtlichkeit zu legen. Daher wurde auch die Sprache SIMULA verwendet, die durch das Klassenkonzept die Möglichkeit bietet, Objekte mit Attributen und spezifischen Anweisungen zu definieren:

z.B. class partitionsmenge; ...eine Partitionsmenge $M_j^{(i)}$ (Zeile 166).

```
begin boolean alt, ganz alt;..bereits erwähnte logische
                              Variable.
      ref(iliste) liste; .....Verweis auf die Liste
                              g_i^-1(j).
      ref(head) kliste1,kliste2;...Listenköpfe der Liste
                              der (x,y),die Elemente dieser
                              P.menge sind, bezüglich Graph1
                              bzw. Graph2.
      integer INDEX;..... INDEX = j.
      boolean procedure complete; dieses Unterprogramm prüft,
      (für eine Partitionsmenge ⊂ Δ ), ob der durch die
      Knoten z, für die (z,z) Element dieser Partitions-
      menge ist, bestimmte Untergraph"vollständig"ist,
      d.h. alle Kanten (x,y) mit x≠y aus derselben
      Partitionsmenge sind.
    die Anweisungen dieser Klasse führen eine Vorbesetzung
    der Attribute durch.
end PARTITIONSMENGE;
```

Weiters wird z.B. der Verfahrensschritt A_1 von den Anweisungen der Class KNOTENPARTITION durchgeführt (Zeile 5oo - 582), der Verfahrensschritt A_2 jedoch von den Anweisungen der Class KANTENPARTITION (Zeile 632 - 752).

Das Programm führt für zwei gegebene Graphen die parallele Berechnung von f und f' durch, wobei die Bedingungen von 4.9 bei jedem Schritt überprüft werden.
Es sind folgende Ausgänge möglich:

- bei der Überprüfung der Bedingungen von 4.9 werden Unterschiede festgestellt, die eine Isomorphie ausschließen.
- die Knoten beider Graphen werden eindeutig angeordnet: ein einziger Isomorphismus existiert.
- die Voraussetzungen von 4.11 sind für ein x_o mit $f(x_o,x_o) =: j$. Es sind dann $|M_j|$ Isomorphismen möglich.
- falls keine der obigen Bedingungen erfüllt ist, wird versucht, die Graphen parallel in unabhängige Untergraphen zu zerlegen, für die dann einzeln die Voraussetzungen von 4.11 geprüft werden.

Die beiden Graphen können auf folgende Arten gegeben sein:

- durch Einlesen der Nachfolgerliste für jeden Knoten (procedure EINLESENG, Zeile 1o7).
- durch Besetzung der Adjazenzmatrix nach einer Zufallsfunktion (procedure ZUFALLSGRAPH, Zeile 29).
- durch Erzeugung eines "starred polygon" ([14]) (procedure STARRED POLYGON, Zeile 49):
  das ist ein Graph, dessen Knoten derart angeordnet werden können($X = \{x_1,..,x_n\}$, daß gilt:
  $(x_i,x_j) \in G \iff \forall\, k=1,..,n-1: (x_{i+k},x_{j+k}) \in G.$
  (i+k,j+k modulo n gerechnet)
  Ein solcher Graph ist gegeben durch sein "Symbol"
  $S := \{s \mid (x_1,x_{1+s}) \in G\}$. (Der bei 4.15 links abgebildete Untergraph ist ein "starred polygon" und hat das Symbol {1,3,4,9,1o,12})

Der zweite Graph kann außerdem durch eine Zufallspermutation (procedureZUFALLSPERMUTATION, Zeile 81) aus dem ersten erzeugt werden (procedure PERMUTIERE , Zeile 99).

```
≠BEGIN≠  ≠REAL≠ ZEITLIMIT;

          ≠INTEGER≠ N,U;
  ≠REAL≠ X;   ≠BOOLEAN≠ SYMMETRISCH, EINLESEN, SCHLEIFEN, AUS;
          ZEITLIMIT := 1000*ININT;
IMAGE :- BLANKS(70);
ANF..
OUTIMAGE;  EJECT(LINE+3);
OUTIMAGE; OUTTEXT(≠≠-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-≠≠);O
                                                        UTIMAGE;
          N := ININT;         X := INREAL;
          SYMMETRISCH := N < 0;          EINLESEN :=  X = 0.0;
          SCHLEIFEN := X < 0;
          ≠IF≠ SYMMETRISCH ≠THEN≠ N := -N;
OUTTEXT(≠≠ N= ≠≠); OUTINT(N,2);
≠IF≠ SYMMETRISCH ≠THEN≠ OUTTEXT(≠≠  SYMMETRISCHER≠≠);
          ≠IF≠ SCHLEIFEN ≠THEN≠ X := -X;
≠IF≠ EINLESEN ≠THEN≠ OUTTEXT(≠≠ GRAPH, EINGELESEN: ≠≠)
  ≠ELSE≠ OUTTEXT(≠≠ ZUFALLSGRAPH≠≠);
≠IF≠ SCHLEIFEN ≠THEN≠ OUTTEXT(≠≠ MIT SCHLEIFEN≠≠);
                    U : =     CLOCK.SUB(7,2).GETINT*2+1;
≠IF≠ ¬ EINLESEN ≠THEN≠
≠BEGIN≠ OUTTEXT(≠≠, KANTENHAEUFIGKEIT=≠≠);
          OUTFIX(X,1,3);
≠END≠ THEN;
≠IF≠ ENDIMAGE ≠THEN≠ ≠ELSE≠ ≠INSPECT≠ SYSIN ≠DO≠
≠BEGIN≠ AUS := IMAGE.SUB(POS,3) = ≠≠AUS≠≠;
  ≠IF≠ AUS ≠THEN≠ SETPOS(POS+3);   ≠END≠;
SIMSET≠BEGIN≠

≠PROCEDURE≠       ZUFALLSGRAPH(F,N  ,KANTENHAEUFIGKEIT,SYMMETRI
                                                        SCH,
                                                SCHLEIFEN);
     ≠INTEGER≠≠ARRAY≠F;  ≠INTEGER≠N   ; ≠REAL≠KANTENHAEUFIGKEIT;
          ≠BOOLEAN≠SYMMETRISCH,SCHLEIFEN;

≠BEGIN≠ ≠INTEGER≠ I,J,K,M;
                    ≠IF≠ ¬ ENDIMAGE ≠THEN≠ U := ININT;
                    OUTTEXT(≠≠, U=≠≠);
                    OUTINT(U,14);
          M := N*(N-1)*KANTENHAEUFIGKEIT;
          ≠IF≠ SYMMETRISCH ≠THEN≠ M := M//2;
          ≠FOR≠ K := 1 ≠STEP≠ 1 ≠UNTIL≠ M ≠DO≠
        ≠BEGIN≠  I := RANDINT(1,N,U);        J := RANDINT(1,N,U);
            F[I,J] := 1;       ≠IF≠ SYMMETRISCH ≠THEN≠ F[J,I]:=1;
          ≠END≠ FOR K;
          ≠IF≠ SCHLEIFEN ≠THEN≠ ≠ELSE≠
≠IF≠ AUS ≠THEN≠ AUSDR (F ,N);
          ≠FOR≠ I := 1 ≠STEP≠ 1 ≠UNTIL≠ N ≠DO≠ F[I,I] :=0;
≠END≠ ZUFALLSGRAPH;

≠PROCEDURE≠ STARRED POLYGON(F,N,SYMMETRISCH);
≠INTEGER≠ ≠ARRAY≠ F; ≠INTEGER≠ N; ≠BOOLEAN≠ SYMMETRISCH;
≠BEGIN≠ ≠INTEGER≠ I,J      ;
          SYSIN.SETPOS(SYSIN.POS+1);
          OUTTEXT(≠≠ STARRED POLYGON, SYMBOL=(≠≠);
          ≠WHILE≠ ¬ ENDIMAGE ≠DO≠
          ≠BEGIN≠  J := ININT; OUTINT(J,2); OUTCHAR($,$);
                    J := J-1;
```

```
                    ≠FOR≠ I := 1 ≠STEP≠ 1 ≠UNTIL≠ N ≠DO≠
                      F[I,(I+J)≠MOD≠N + 1] := 1;
                    J := N - J - 2;
                    ≠IF≠ ¬ SYMMETRISCH ≠THEN≠ ≠ELSE≠
                    ≠FOR≠ I := 1 ≠STEP≠ 1 ≠UNTIL≠ N ≠DO≠
                      F[I,(I+J)≠MOD≠N + 1] := 1;
          ≠END≠ WHILE ¬ENDIMAGE;
          SETPOS(POS-1);   OUTCHAR(§)§); OUTIMAGE;
≠END≠ STARRED POLYGON;

≠PROCEDURE≠ AUSDR(F,N);
≠INTEGER≠ ≠ARRAY≠ F; ≠INTEGER≠ N;
≠BEGIN≠ ≠INTEGER≠ I,J;
          OUTIMAGE;
          ≠FOR≠ I := 1 ≠STEP≠ 1 ≠UNTIL≠ N ≠DO≠
          ≠BEGIN≠ OUTINT(I,2);  OUTCHAR($:$);
                    ≠FOR≠ J := 1 ≠STEP≠ 1 ≠UNTIL≠ N ≠DO≠
                    ≠IF≠ F[I,J]  = 1 ≠THEN≠
                      ≠BEGIN≠ OUTINT(J,2); OUTCHAR($,$); ≠END≠;
                    SETPOS(POS-1); OUTCHAR($*$);  SETPOS(POS+2);
          ≠END≠ FOR I;
≠END≠ AUSDR;


≠PROCEDURE≠       ZUFALLSPERMUTATION          (PERMUT,N);
          ≠INTEGER≠ ≠ARRAY≠ PERMUT; ≠INTEGER≠ N;

≠BEGIN≠ ≠INTEGER≠ I,J,K,M,Z;
          ≠FOR≠ K := 1 ≠STEP≠ 1 ≠UNTIL≠ N ≠DO≠ PERMUT[K] := K;
OUTIMAGE; OUTTEXT(≠≠ZUFALLSPERMUTATION, U=≠≠);
OUTINT(U,TRUNK(LN(U)/LN(10.0)) + 1 );  OUTIMAGE;
          M := RANDINT(1,N,U);
          ≠FOR≠ K := 1 ≠STEP≠ 1 ≠UNTIL≠ M ≠DO≠
        ≠BEGIN≠  I := RANDINT(1,N,U);        J := RANDINT(1,N,U);
                    Z := PERMUT[I];  PERMUT[I] := PERMUT[J];
                    PERMUT[J] := Z;
          ≠END≠ FOR K;
≠FOR≠ K := 1 ≠STEP≠ 1 ≠UNTIL≠ N ≠DO≠ OUTINT(K,3); OUTIMAGE;
≠FOR≠ K := 1 ≠STEP≠ 1 ≠UNTIL≠ N ≠DO≠ OUTINT(PERMUT[K],3); OUTI
                                                    MAGE;
≠END≠ ZUFALLSPERMUTATION;


≠PROCEDURE≠       PERMUTIERE          (F1,P,F2,N);
          ≠INTEGER≠ ≠ARRAY≠ F1,F2,P; ≠INTEGER≠ N;
≠BEGIN≠ ≠INTEGER≠ I,J;
        ≠FOR≠I:=1≠STEP≠1≠UNTIL≠N≠DO≠≠FOR≠J:=1≠STEP≠1≠UNTIL≠N≠DO≠
          F2[P[I],P[J]] := F1[I,J];
≠END≠PERMUTIERE;


≠PROCEDURE≠ EINLESENG (F,N,SYMMETRISCH);
      ≠INTEGER≠ ≠ARRAY≠ F;  ≠INTEGER≠ N; ≠BOOLEAN≠ SYMMETRISCH;

≠BEGIN≠ ≠INTEGER≠ I,J,K;
          INIMAGE;    OUTTEXT(≠≠>>>>>>>≠≠);
          ≠WHILE≠ SYSIN.IMAGE.SUB(1,4) ≠NE≠ ≠≠ENDE≠≠ ≠DO≠
          ≠BEGIN≠
OUTTEXT(SYSIN.IMAGE.SUB(1,70));OUTIMAGE; SYSIN.SETPOS(71);
SYSIN.IMAGE.SKIPBLANKS; ≠IF≠ SYSIN.POS < 81 ≠THEN≠ ≠BEGIN≠
```

```
116 OUTTEXT(SYSIN.IMAGE.SUB(7,10)); OUTIMAGE; ≠END≠; SYSIN.SETPOS
                                                    (1);
117                     ≠WHILE≠ ¬ ENDIMAGE ≠DO≠
118                     ≠BEGIN≠ I := ININT;
119                               ≠WHILE≠ INCHAR ≠NE≠ $*$ ≠DO≠
120                             ≠BEGIN≠  J := ININT;        F[I,J] :=1;
121                                 ≠IF≠ SYMMETRISCH ≠THEN≠ F[J,I]:=1;
122                               ≠END≠ WHILE NE * ;
123                     ≠END≠  WHILE < K;
124                     ININIMAGE;
125         ≠END≠ WHILE NE ENDE;        SYSIN.SETPOS(79);    OUTIMAGE;
126 ≠END≠ EINLESEN;
127
128 LINK     ≠CLASS≠  QUADRUPEL;
129 ≠BEGIN≠ ≠INTEGER≠K,S,T,M ;≠END≠;
130
131 HEAD     ≠CLASS≠  QUADRUPELLISTE;
132 ≠BEGIN≠ ≠REF≠(HEAD)RESERVOIR;
133          ≠INTEGER≠ I;
134          RESERVOIR :- ≠NEW≠ HEAD;
135          ≠FOR≠ I := 1 ≠STEP≠ 1 ≠UNTIL≠ N ≠DO≠
136          ≠NEW≠ QUADRUPEL.INTO(RESERVOIR);
137 ≠END≠    QUADRUPELLISTE;
138
139 ≠EXTERNAL≠ ≠COMPASS≠ ≠INTEGER≠ ≠PROCEDURE≠          COMPARE(L1,L
                                                    2);
140 ≠REF≠(ILISTE) L1,L2;;
141
142 ≠EXTERNAL≠ ≠COMPASS≠     ≠PROCEDURE≠          INCM1(QLISTE,PR,PS,P
                                                    T);
143 ≠INTEGER≠ PR,PS,PT;  ≠REF≠(HEAD) QLISTE;;
144
145 ≠EXTERNAL≠ ≠COMPASS≠     ≠PROCEDURE≠          COMPR(QLISTE,L);
146 ≠REF≠(HEAD) QLISTE; ≠REF≠(ILISTE) L;;
147
148 ≠EXTERNAL≠ ≠COMPASS≠     ≠PROCEDURE≠          SETLU(I);
149 ≠INTEGER≠ I;;
150
151 ≠CLASS≠ ILISTE;
152 ≠BEGIN≠ ≠INTEGER≠ I1,I2;
153          ≠INTEGER≠ ≠PROCEDURE≠ LAENGE;  LAENGE := I2 - I1 + 1;
154 ≠END≠ ILISTE;
155
156
157
158 LINK     ≠CLASS≠  KK;
159 ≠BEGIN≠ ≠REF≠(PARTITIONSMENGE)PM; ≠END≠;
160 KK       ≠CLASS≠  KNOTEN(X);          ≠INTEGER≠ X;
161          ≠BEGIN≠  ≠END≠     KNOTEN;
162 KK       ≠CLASS≠  KANTE(X,Y);         ≠INTEGER≠ X,Y;
163          ≠BEGIN≠ ≠REF≠(KANTE) T ;;  ≠END≠     ;
164
165
166 LINK     ≠CLASS≠  PARTITIONSMENGE;
167          ≠BEGIN≠  ≠BOOLEAN≠ ALT, GANZ ALT;
168                   ≠REF≠(ILISTE) LISTE;
169                   ≠REF≠(HEAD) KLISTE1,KLISTE2;
170                   ≠INTEGER≠ INDEX;
171
172 ≠BOOLEAN≠ ≠PROCEDURE≠ COMPLETE;
```

```
173 ≠IF≠ KLISTE1.FIRST.SUC = = KLISTE1.LAST ≠THEN≠ COMPLETE := ≠TR
                                                    UE≠≠ELSE≠
174 ≠BEGIN≠
175           ≠REF≠(KNOTEN)K;    ≠INTEGER≠X;
176           ≠REF≠(PARTITIONSMENGE)PV;
177           K:-KLISTE1.FIRST;       X:=K.X;
178           K:-K.SUC;       PV :- GRAPH1.KF[X,K.X].PM;
179           ≠FOR≠ K:- K.SUC    ≠WHILE≠ K =/= ≠NONE≠ ≠DO≠
180        ≠IF≠ PV =/= GRAPH1.KF[X,K.X].PM ≠THEN≠ ≠GOTO≠ E COMPLETE;
181           COMPLETE := ≠TRUE≠;
182 E COMPLETE.. ;  ≠END≠ COMPLETE ELSE;
183                      LISTE :- ≠NEW≠ ILISTE;
184                      KLISTE1 :- ≠NEW≠ HEAD;
185                      KLISTE2 :- ≠NEW≠ HEAD;
186           ≠END≠ PARTITIONSMENGE;
187
188 ≠PROCEDURE≠       AUSGABE(H);                  ≠REF≠ (HEAD) H;
189
190 ≠BEGIN≠ ≠REF≠(PARTITIONSMENGE) PM; ≠REF≠(KNOTEN)K;
191           PM :- H.FIRST;          OUTIMAGE;
192           ≠WHILE≠ PM =/= ≠NONE≠ ≠DO≠  ≠INSPECT≠ PM ≠DO≠
193           ≠BEGIN≠  ≠FOR≠ K :- KLISTE1.FIRST, KLISTE2.FIRST ≠DO≠
194                      ≠BEGIN≠  OUTCHAR($[$);
195                               ≠WHILE≠ K =/= ≠NONE≠ ≠DO≠
196                              ≠BEGIN≠  OUTINT(K.X,2); OUTCHAR($,$);
197                                       K :- K.SUC;
198                               ≠END≠ WHILE K;
199                               SETPOS(POS-1);     OUTCHAR($]$);
200                             OUTCHAR($:$);
201                      ≠END≠ FOR K:- KLISTE1 ;
202                 SETPOS(POS-1); OUTCHAR($;$);          SETPOS(POS+2);
203                      PM :- PM.SUC;
204           ≠END≠ WHILE PM;
205 ≠END≠ AUSGABE;
206
207
208
209
210
211
212
213 ≠REF≠(GRAPH)      GRAPH1, GRAPH2;
214
215 ≠CLASS≠ GRAPH (F,N);
216           ≠INTEGER≠ ≠ARRAY≠ F;        ≠INTEGER≠ N;
217
218 ≠BEGIN≠ ≠REF≠ (KK)          ≠ARRAY≠  KF [1..N,1..N] ;
219
220       ≠REF≠ (ILISTE) ≠PROCEDURE≠           G(K);    ≠REF≠(KK) K;
221   ≠BEGIN≠   ≠REF≠(PARTITIONSMENGE) PR,PS,PT;     ≠REF≠(KNOTEN)KZ;
222                      ≠INTEGER≠ X,Y,Z,R;          ≠BOOLEAN≠ ALT;
223                      ≠REF≠(HEAD)KLISTE;
224
225                      ≠IF≠ K ≠IS≠ KNOTEN ≠THEN≠ ≠GOTO≠ KN;
226                    X := K≠QUA≠KANTE.X;          Y := K≠QUA≠KANTE.Y;
227                      PR :- KNOTP.LAST;
228                      ≠WHILE≠ PR =/= ≠NONE≠ ≠DO≠
229                      ≠BEGIN≠
230                               ALT := PR.GANZ ALT;
231                  R := PR.INDEX;   KLISTE:-≠IF≠≠THIS≠GRAPH==GRAPH1
```

```
232                     ≠THEN≠ PR.KLISTE1 ≠ELSE≠ PR.KLISTE2;
233                             KZ :- KLISTE.FIRST;
234                             ≠WHILE≠ KZ =/= ≠NONE≠ ≠DO≠
235                             ≠BEGIN≠  Z := KZ.X;
236                             PS :- KF [X,Z].PM;
237                             PT :- KF [Y,Z].PM;
238                                     ≠IF≠ ALT ≠THEN≠
239                                   ≠BEGIN≠  ≠IF≠ PS.ALT ∧ PT.ALT
240                                                 ≠THEN≠ ≠GOTO≠ E1;
241                                     ≠END≠ THEN;
242
243                               INCM1(GLISTE,R,PS.INDEX,PT.INDEX);
244
245                             E1..    KZ :- KZ.SUC;
246                             ≠END≠   WHILE KZ =/= NONE;
247                             PR :- PR.PRED;
248                     ≠END≠   WHILE PR =/= NONE;
249                     ≠GOTO≠ E;
250
251
252             KN..    X := K≠QUA≠KNOTEN.X;
253                     PR :- KNOTP.LETZTE ALT;
254 B..                 ALT := PR.ALT;  ≠IF≠ ALT ∧ ¬KN1 ≠THEN≠ ≠GOTO≠
                                                        E3;
                  R := PR.INDEX;   KLISTE:-≠IF≠≠THIS≠GRAPH==GRAPH1
255                   ≠THEN≠ PR.KLISTE1 ≠ELSE≠ PR.KLISTE2;
256                 KZ :- KLISTE.FIRST;
257                 ≠WHILE≠ KZ =/= ≠NONE≠ ≠DO≠
258                 ≠BEGIN≠  Z := KZ.X;
259                             PS :- KF [X,Z].PM;
260                  ≠IF≠ (≠IF≠ALT∧KN1≠THEN≠PS.ALT≠ELSE≠ ≠FALSE≠)
261                             ≠THEN≠ ≠GOTO≠ E2;
262
263                               INCM1(GLISTE,R,PS.INDEX,PS.INDEX);
264
265                             KZ :- KZ.SUC;
266 E2..                ≠END≠   WHILE KZ =/= NONE;
267                     PR :- PR.PRED;
268 E3..                ≠IF≠ PR =/= ≠NONE≠ ≠THEN≠ ≠GOTO≠ B;
269                     ;
270 E..                COMPR(QLISTE,L);     G:- L;
271             ≠END≠ G(K);
272
273
274             ≠REF≠(HEAD) P;              ≠INTEGER≠ I,J,F1,F2,MAX;
275             ≠REF≠(PARTITIONSMENGE) ≠PROCEDURE≠          WO(I);
276                     ≠INTEGER≠ I;
277             ≠BEGIN≠ ≠ REF≠ (PARTITIONSMENGE) PM;
278                     PM :- P.FIRST;
279                     ≠WHILE≠ PM =/= ≠NONE≠ ≠DO≠
280                 ≠BEGIN≠  ≠IF≠ I < PM.INDEX ≠THEN≠ ≠GOTO≠ KLEINER;
281                             ≠IF≠ I = PM.INDEX ≠THEN≠ ≠GOTO≠ GLEICH;
282                               PM :- PM.SUC;
283                     ≠END≠ WHILE PM;
284                     PM :- ≠NEW≠ PARTITIONSMENGE;  PM.INDEX := I;
285                     PM.INTO(P);
286                     ≠GOTO≠ GLEICH;
287                     ≠NEW≠ PARTITIONSMENGE.PRECEDE(PM);
288 KLEINER..           PM :- PM.PRED;    PM.INDEX := I;
289                     WO :- PM;
290 GLEICH..
```

```
291           ≠END≠ W0;
292
293
294           ≠REF≠ (KK) K,K1;
295           ≠REF≠(ILISTE) L;              L:- ≠NEW≠ ILISTE;
296
297         ≠FOR≠I:=1≠STEP≠1≠UNTIL≠N≠DO≠≠FOR≠J:=1≠STEP≠1≠UNTIL≠N≠DO≠
298             ≠IF≠ MAX < F[I,J] ≠THEN≠ MAX := F[I,J]
299          ≠ELSE≠ ≠IF≠ F[I,J] < 0 ≠THEN≠ ABORT(≠≠LABEL NEGATIV≠≠);
300           MAX := MAX + 10;
301
302           P :- KNOTP;
303           ≠FOR≠ I := 1 ≠STEP≠ 1 ≠UNTIL≠ N ≠DO≠
304           ≠BEGIN≠  K :- ≠NEW≠ KNOTEN(I);
305                    K.PM :- W0(F[I,I]);
306                    KF[I,1] :- K;
307           ≠END≠ FOR I;
308
309           P :- KANTP;
310       ≠FOR≠I:=1≠STEP≠1≠UNTIL≠N≠DO≠≠FOR≠J:=I+1≠STEP≠1≠UNTIL≠N≠DO≠
311      ≠BEGIN≠ K:- ≠NEW≠ KANTE(I,J);          K1 :- ≠NEW≠ KANTE(J,I);
312                K≠QUA≠KANTE.1 :- KT;            KT≠QUA≠KANTE.T :- K;
313                    F1 := F[I,J];                 F2 := F[J,I];
314               K.PM :- W0(F1*MAX + F2);     K1.PM :- W0(F2*MAX+F1);
315                    KF[I,J] :- K;                 KF[J,I] :- K1;
316           ≠END≠ FOR I,J;
317
318           ≠IF≠ GRAPH1 =/= ≠NONE≠ ≠THEN≠ ≠GOTO≠ M2;
319         ≠FOR≠I:=1≠STEP≠1≠UNTIL≠N≠DO≠≠FOR≠J:=1≠STEP≠1≠UNTIL≠N≠DO≠
320             ≠INSPECT≠ KF[I,J] ≠DO≠ INTO(PM.KLISTE1);
321           ≠GOTO≠ E;
322 M2..      ≠FOR≠I:=1≠STEP≠1≠UNTIL≠N≠DO≠≠FOR≠J:=1≠STEP≠1≠UNTIL≠N≠D
                                                          O≠
323             ≠INSPECT≠ KF[I,J] ≠DO≠ INTO(PM.KLISTE2);
324 E..       ;
325 ≠END≠ GRAPH;
326
327
328 ≠REF≠(HEAD) SINGLES;
329
330
331
332 ≠REF≠(HEAD) UGL;     ≠REF≠(SUBGRAPH) SG;
333 LINK      ≠CLASS≠  SUBGRAPH;
334 ≠BEGIN≠  ≠REF≠(PARTITION) P;
335           ≠PROCEDURE≠ ISOTEST2;
336            ≠BEGIN≠
337                      ≠REF≠(PARTITIONSMENGE)PM,PN,PV;
338                     ≠REF≠(PARTITION) SAVP;      ≠INTEGER≠ N1;
339                       ≠REF≠(KNOTEN)K1,K2,K3;
340                     SAVP:-KNOTP;
341                     P.LETZTE ALT :- P.LAST;
342                     KNOTP :- P;
343           N1 := 4*P.CARD;
344           SETLU(0);
345
346 PM :- KNOTP . FIRST;
347
348 ≠WHILE≠ PM =/= ≠NONE≠ ≠DO≠
349 ≠BEGIN≠
```

```
350         ≠IF≠ GRAPH1.G(PM.KLISTE1.FIRST≠QUA≠KNOTEN).LAENGE  =  N1
351             ≠THEN≠ ≠GOTO≠ ISO2;
352           PM :- PM.SUC;
353 ≠END≠ WHILE PM;
354
355 WEISS NICHT..
356 OUTIMAGE; OUTTEXT(≠≠KEINE AUSSAGE UEBER ISOMORPHISMUS≠≠);
357 OUTIMAGE;  OUTTEXT(≠≠RESULTIERENDE KNOTENPARTITION: ≠≠);
358           AUSGABE(KNOTP);
359           ≠GOTO≠ E ISOTEST2;
360
361 ISO2..
362  OUTIMAGE; OUTINT(PM.KLISTE1.CARDINAL,2);
363 OUTTEXT(≠≠  ISOMORPHISMEN MOEGLICH; EINER DAVON:    ≠≠);
364
365 K1 :- PM.KLISTE1.FIRST; K2 :- pM.KLISTE2.FIRST;
366 X1 :- K1.X;              X2 := K2.X;
367 PN :- KNOTP.FIRST;                ≠WHILE≠ PN =/= ≠NONE≠ ≠DO≠
368 ≠BEGIN≠ K1 :- PN.KLISTE1.FIRST;    K2 :- PN.KLISTE2.FIRST;
369           ≠WHILE≠ K1 =/= ≠NONE≠ ≠DO≠
370           ≠BEGIN≠  PV :- GRAPH1.KF[X1,K1.X].PM;          K3 :- K2;
371            ≠WHILE≠ GRAPH2.KF[X2,K3.X].PM=/=PV ≠DO≠ K3 :- K3.SUC;
372                    ≠IF≠ K3 == K2 ≠THEN≠ K2 :- K2.SUC
373                    ≠ELSE≠ K3.PRECEDE(K2);
374                    K1 :- K1.SUC;
375           ≠END≠ WHILE K1;
376           PN :- PN.SUC;
377 ≠END≠ WHILE PN;
378
379 AUSGABE(KNOTP);
380 E ISOTEST2..      KNOTP :- SAVP;
381           ≠END≠ ISOTEST2;
382           P :- ≠NEW≠ PARTITION;
383 ≠END≠ SUBGRAPH;
384
385 ≠PROCEDURE≠ DECOMPOSE(KNOTP)INTO LIST OF COMPONENTS:(L);
386 ≠REF≠(PARTITION)KNOTP;     ≠REF≠(HEAD) L;
387 ≠BEGIN≠
388         ≠REF≠(KNOTEN) K;          ≠REF≠(PARTITIONSMENGE) P1,P2,P3;
389           ≠REF≠(SUBGRAPH)SG;          ≠INTEGER≠ X;
390           ≠FOR≠ P1 :- KNOTP.FIRST ≠WHILE≠ P1 =/= ≠NONE≠ ≠DO≠
391           ≠BEGIN≠
392           SG :- ≠NEW≠ SUBGRAPH; SG.INTO(L);   P1.INTO(SG.P);
393           ≠WHILE≠ P1 =/= ≠NONE≠ ≠DO≠
394           ≠BEGIN≠  K :- P1.KLISTE1.FIRST;      X:=K.X;
395                    P2 :- KNOTP.FIRST; ≠WHILE≠ P2 =/= ≠NONE≠ ≠DO≠
396                    ≠BEGIN≠  K :- P2.KLISTE1.FIRST;
397                             P3 :- GRAPH1.KF[X,K.X].PM;
398                             ≠FOR≠K:-K.SUC≠WHILE≠K=/=≠NONE≠ ≠DO≠
399                             ≠IF≠ P3 =/= GRAPH1.KF[X,K.X].PM
400                             ≠THEN≠ ≠BEGIN≠ P3 :- P2.SUC;
401                                      P2.INTO(SG.P);   P2 :- P3;
402                                    ≠GOTO≠ E1;
403                             ≠END≠THEN;
404                             P2 :- P2.SUC;
405                    E1..   ;  ≠END≠ WHILE P2;
406                    P1 :- P1.SUC;
407           ≠END≠ WHILE P1;
408           ≠END≠ FOR P1;
409 ≠END≠ DECOMPOSE;
```

```
410
411 ≠REF≠( PARTITION) KNOTP,KANTP;
412 HEAD≠CLASS≠ PARTITION;
413 ≠BEGIN≠ ≠REF≠(PARTITIONSMENGE) LETZTE ALT;
414          ≠BOOLEAN≠ NEU, NON ISOMORPH;
415          ≠INTEGER≠ ≠PROCEDURE≠ CARD;
416          ≠BEGIN≠  ≠REF≠(PARTITIONSMENGE)PM; ≠INTEGER≠ C;
417                   PM :- FIRST;
418              ≠WHILE≠PM=/=≠NONE≠ ≠DO≠ ≠BEGIN≠ C:= C + PM.KLISTE1
419                    .CARDINAL ;  PM :- PM.SUC; ≠END≠; CARD := C;
420          ≠END≠ CARDINAL;
421 ≠END≠PARTITION;
422
423 PARTITION ≠CLASS≠ KNOTENPARTITION;
424 ≠BEGIN≠ ≠REF≠(HEAD) RESERVE;
425          ≠REF≠(PARTITIONSMENGE) ≠PROCEDURE≠ NEUKNOTENPM(L);
426                   ≠REF≠ (ILISTE) L;
427
428          ≠BEGIN≠  ≠REF≠(PARTITIONSMENGE) PM;
429           PM :- ≠IF≠ RESERVE.EMPTY ≠THEN≠ ≠NEW≠ PARTITIONSMENGE
430                   ≠ELSE≠ RESERVE.FIRST;
431                   ≠INSPECT≠ PM ≠DO≠
432                   ≠BEGIN≠ ≠IF≠ L == ≠NONE≠ ≠THEN≠ ≠GOTO≠ NO;
433                   LISTE.I1 := L.I1;       LISTE.I2 := L.I2;
434        NO..      ALT := GANZ ALT := ≠FALSE≠;   ≠END≠ INSPECT PM;
435                   NEUKNOTENPM :- PM;
436          ≠END≠ NEUKNOTENPM;
437
438          ≠PROCEDURE≠ ENTFERNE(PN);
439                   ≠REF≠(PARTITIONSMENGE) PN;
440
441        ≠BEGIN≠  ≠REF≠(PARTITIONSMENGE) PV; ≠REF≠(KNOTEN) K1,K2;
442                   ≠INTEGER≠ X1,X2;
443                            ≠INSPECT≠ PN ≠DO≠
444                            ≠BEGIN≠  INTO(SINGLES);
445                                     K1 :- KLISTE1.FIRST;
446                                     K2 :- KLISTE2.FIRST;
447                                  ≠IF≠ PRED == ≠NONE≠ ≠THEN≠ INDEX := 1
448                                   ≠ELSE≠  INDEX := PRED≠QUA≠PARTITIONS
449                                              MENGE.INDEX + 1 ;
450                                   ≠END≠     INSPECT PN;
451                                 X1 := K1.X;  X2 := K2.X;  PV :- FIRST;
452 E2..                              ≠IF≠ PV == ≠NONE≠ ≠THEN≠ ≠GOTO≠ E2;
453                                   K1 :- PV.KLISTE1.FIRST;
454                                   K2 :- PV.KLISTE2.FIRST;
455                                   ≠WHILE≠ K1 =/= ≠NONE≠ ≠DO≠
456                                   ≠BEGIN≠  ≠INSPECT≠ GRAPH1 ≠DO≠
457                            ≠BEGIN≠  ≠INSPECT≠ KF [X1,K1.X]≠WHEN≠KANTE≠DO≠
458                                        ≠BEGIN≠  OUT; PM :- ≠NONE≠ ≠END≠;
459                                     ≠INSPECT≠ KF [K1.X,X1]≠WHEN≠KANTE≠DO≠
460                                        ≠BEGIN≠  OUT; PM :- ≠NONE≠ ≠END≠;
461                                                 KF [X1,K1.X] :- ≠NONE≠;
462                                                 KF [K1.X,X1] :- ≠NONE≠;
463                                   ≠END≠;
464                                   ≠INSPECT≠ GRAPH2 ≠DO≠
465                            ≠BEGIN≠  ≠INSPECT≠ KF [X2,K2.X]≠WHEN≠KANTE≠DO≠
466                                        ≠BEGIN≠  OUT; PM :- ≠NONE≠ ≠END≠;
467                                     ≠INSPECT≠ KF [K2.X,X2]≠WHEN≠KANTE≠DO≠
468                                        ≠BEGIN≠  OUT; PM :- ≠NONE≠ ≠END≠;
469                                                 KF [K2.X,X2] :- ≠NONE≠;
```

```
                                        FF [X2,K2.X] :- ≠NONE≠;
                              ≠END≠;
                                   K1 :- K1.SUC;      K2 :- K2.SUC;
                              ≠END≠     WHILE K1 =/= NONE;
                              PV :- PV.SUC;      ≠GOTO≠ B2;
E2..          ;
        ≠END≠ ENTFERNE;

        ≠REF≠(PARTITIONSMENGE) PA,PN,PV,LETZTE NEU;
        ≠REF≠ (ILISTE) L;
        ≠INTEGER≠ I,IND;              ≠REF≠ (KNOTEN) K1,K2;
        ≠REAL≠ ZEIT;

        SINGLES  :- ≠NEW≠ HEAD;
        RESERVE :- ≠NEW≠ HEAD;
        DETACH;
WEITER1..
        KNI := ≠TRUE≠;
        ZEIT := ELAPSED;
        LETZTE ALT :- LAST;
        PA :- FIRST;
        ≠WHILE≠ PA =/= ≠NONE≠ ≠DO≠  ≠INSPECT≠ PA ≠DO≠
        ≠BEGIN≠  ALT := GANZ ALT;
                 GANZ ALT := ≠TRUE≠;
                 PA :- SUC;
        ≠END≠ WHILE PA =/= NONE;

WEITER..
        OUTCHAR(5*6);
        I := KANTP.LAST≠QUA≠PARTITIONSMENGE.INDEX;
        LETZTE ALT :- LAST;
        PA :- FIRST;

B1..    K1 :- PA.KLISTE1.FIRST;
        K2 :- PA.KLISTE2.FIRST;
        LETZTE NEU :- LAST;
        SETLU(0);
                              ≠WHILE≠ K1 =/= ≠NONE≠ ≠DO≠
                              ≠BEGIN≠ L :- GRAPH1.G(K1);
                                       PN :- LETZTE NEU  ;
                                       ≠FOR≠ PN :- PN.SUC ≠WHILE≠
                                          PN =/= ≠NONE≠ ≠DO≠
                              ≠BEGIN≠   IND := COMPARE(L,PN.LISTE);

                                                ≠IF≠ IND > 0 ≠THEN≠
                                                   ≠GOTO≠ MAGGICRE;
                                                ≠IF≠ IND = 0 ≠THEN≠
                       ≠BEGIN≠ SETLU(L.I1-1); ≠GOTO≠ UGUALE; ≠END≠;
                                       ≠END≠     FOR PN :- PN.SUC;
                                       PN :- NEUKNOTENPM(L);
                                    PN.INTO(≠THIS≠KNOTENPARTITION);
                                       ≠GOTO≠ UGUALE;
                       MAGGICRE..      NEUKNOTENPM(L).PRECEDE(PN);
                                       PN :- PN.PRED;
                       UGUALE..        K1.PM :- PN;
                                       K1 :- K1.SUC;
                              ≠END≠     WHILE K1 =/= NONE;

                              ≠WHILE≠ K2 =/= ≠NONE≠ ≠DO≠
                              ≠BEGIN≠ L :- GRAPH2.G(K2);
```

```
                                        PN :- LETZTE NEU;
                                    ≠FOR≠ PN :- PN.SUC ≠WHILE≠ PN =/=
                                        ≠NONE≠ ≠DO≠
                                ≠BEGIN≠  IND := COMPARE(L,PN.LISTE);

                                              ≠IF≠ IND > 0 ≠THEN≠
                                            ≠GOTO≠ NICHT ISOMORPH;
                                              ≠IF≠ IND = 0 ≠THEN≠
                        ≠BEGIN≠ SETLU(L,I1-1); ≠GOTO≠ IGUAL ; ≠END≠;
                       ≠END≠    FOR PN:-PN.SUC; ≠GOTO≠NICHTISOMORPH;
                       IGUAL..          K2.PM :- PN;
                                        K2 :- K2.SUC;
                                ≠END≠   WHILE K2 =/= NONE;

                                ≠IF≠ LETZTE NEU.SUC == LAST ≠THEN≠
                            ≠BEGIN≠ ≠INSPECT≠ LAST≠QUA≠PARTITIONSMENGE
                                ≠DO≠ ≠BEGIN≠    ALT := ≠TRUE≠;
                                        GANZ ALT := PA.GANZ ALT;

                                ≠END≠;   ≠END≠    THEN ;


                                PN :- LETZTE NEU;
                        ≠FOR≠ PN :- PN.SUC ≠WHILE≠ PN =/= ≠NONE≠ ≠DO≠
                                ≠INSPECT≠ PN ≠DO≠
                                ≠BEGIN≠  INDEX := I := I+1;

                                ≠END≠    INSPECT PN;
    ≠IF≠PA=/=LETZTE ALT≠THEN≠ ≠BEGIN≠PA:-PA.SUC; ≠GOTO≠B1≠END≠;
B3..    PA :- FIRST;
        K1 :- PA.KLISTE1.FIRST;
        K2 :- PA.KLISTE2.FIRST;
        ≠WHILE≠ K1 =/= ≠NONE≠ ≠DO≠
        ≠BEGIN≠  ≠INSPECT≠ K1 ≠DO≠ INTO(PM.KLISTE1);
                 ≠INSPECT≠ K2 ≠DO≠ INTO(PM.KLISTE2);
                 K1 :- PA.KLISTE1.FIRST;
                 K2 :- PA.KLISTE2.FIRST;
        ≠END≠ WHILE K1 =/= NONE;
        PA.INTO(RESERVE);
        ≠IF≠ PA =/= LETZTE ALT ≠THEN≠ ≠GOTO≠ B3;

        NEU := ≠FALSE≠;
        PN :- FIRST;
        ≠WHILE≠ PN =/= ≠NONE≠ ≠DO≠
        ≠INSPECT≠ PN ≠DO≠
        ≠IF≠ KLISTE1.CARDINAL ≠NE≠ KLISTE2.CARDINAL
        ≠THEN≠ ≠GOTO≠ NICHT ISOMORPH
        ≠ELSE≠   ≠BEGIN≠  PN :- SUC;
                          ≠IF≠ ¬ALT ≠THEN≠ NEU := ≠TRUE≠
                     ≠ELSE≠ ≠IF≠ KLISTE1.FIRST== KLISTE1.LAST
                       ≠THEN≠ENTFERNE(≠THIS≠PARTITIONSMENGE);
                 ≠END≠    ELSE;

≠IF≠ ZEITLIMIT < E LAPSED ≠THEN≠ ABORT(≠≠ZEITLIMIT≠≠);
        KN1 := ≠FALSE≠;
        ≠IF≠ NEU ∧ ¬EMPTY ≠THEN≠ ≠GOTO≠ WEITER;
        ZEIT :=(ELAPSED-ZEIT)/100.0;
        OUTFRAC(ZEIT,1,≠IF≠ZEIT<10.0≠THEN≠3≠ELSE≠TRUNK(LN(ZEIT
                                                )/LN(10.0)
```

```
589             )+2);
590          LETZTE ALT :- LAST;
591          DETACH;
592
593
594          ≠GOTO≠ WEITER1;
595
596 NICHT ISOMORPH..
597          NON ISOMORPH := ≠TRUE≠;
598 ≠END≠    KNOTENPARTITION;
599
600
601 PARTITION ≠CLASS≠ KANTENPARTITION;
602 ≠BEGIN≠
603
604          ≠REF≠(HEAD)RESERVE;
605          ≠REF≠ (PARTITIONSMENGE) ≠PROCEDURE≠ NEUKANTENPM(L);
606          ≠REF≠(ILISTE) L;
607
608          ≠BEGIN≠  ≠REF≠(PARTITIONSMENGE) PM;
609           PM :- ≠IF≠ RESERVE.EMPTY ≠THEN≠ ≠NEW≠ PARTITIONSMENGE
610                    ≠ELSE≠ RESERVE.FIRST;
611                    ≠INSPECT≠ PM ≠DO≠
612                    ≠BEGIN≠ ≠IF≠ L == ≠NONE≠ ≠THEN≠ ≠GOTO≠ NO;
613                    LISTE.I1 := L.I1;  LISTE.I2 := L.I2;
614          NO..      ALT := ≠FALSE≠;      ≠IF≠ KLIST E1 == ≠NONE≠
615                           ≠THEN≠    ≠BEGIN≠ KLISTE1 :-≠NEW≠HEAD;
616                                             KLISTE2 :-≠NEW≠HEAD;
617                                               ≠END≠  THEN;
618                    ≠END≠ INSPECT PM;
619                    NEUKANTENPM :-  PM;
620          ≠END≠ NEU KANTENPM;
621          ≠BOOLEAN≠ SYM;
622          ≠REF≠(PARTITIONSMENGE) PA,PN,PV,LETZTE NEU;
623          ≠REF≠(KANTE) K1,K2;
624          ≠REF≠(ILISTE) L;            ≠INTEGER≠ I, IND;
625          ≠REAL≠ ZEIT;
626
627
628          RESERVE :- ≠NEW≠ HEAD;
629          DETACH;
630 WEITER ..
631          ZEIT := ELAPSED;
632          I := FIRST≠QUA≠PARTITIONSMENGE.INDEX - 1;
633          NEU := ≠FALSE≠;
634          LETZTE ALT :- LAST;
635          PA :- FIRST;
636
637 B1..     LETZTE NEU :- LAST;
638          SETLU(0);
639          ≠IF≠ PA.KLISTE1.EMPTY ≠THEN≠ ≠GOTO≠ E1;
640          K1 :- PA.KLISTE1.FIRST;
641          SYM :=              K1.T.PM == PA    ;
642         ≠FOR≠ K1 :- PA.KLISTE1.FIRST ≠WHILE≠ K1 =/= ≠NONE≠ ≠DO≠
643          ≠BEGIN≠  L :- GRAPH1.G(K1);
644                   PN :- LETZTE NEU;
645                   ≠FOR≠ PN :- PN.SUC ≠WHILE≠ PN =/= ≠NONE≠ ≠DO≠
646                            ≠BEGIN≠  IND := COMPARE(L,PN.LISTE);
647                                              ≠IF≠ IND > 0 ≠THEN≠
648                            ≠GOTO≠ MAGGIORE;
```

```
649                                                  ≠IF≠ IND = 0 ≠THEN≠
650                    ≠BEGIN≠ SETLU(L.I1-1); ≠GOTO≠ UGUALE; ≠END≠;
651                  ≠END≠ FOR PN .- PN.SUC;
652                  PN :- NEUKANTENPM(L);
653                  PN.INTO(≠THIS≠ KANTENPARTITION);
654                  ≠GOTO≠ UGUALE;
655 MAGGIORE..       NEUKANTENPM(L).PRECEDE(PN);
656                  PN :- PN.PRED;
657 UGUALE..         K1.INTO(PN.KLISTE1);
658 ≠END≠ FOR K1;
659
660        ≠FOR≠ K2 :- PA.KLISTE2.FIRST ≠WHILE≠ K2 =/= ≠NONE≠ ≠DO≠
661         ≠BEGIN≠  L :- GRAPH2.G(K2);
662                  PN :- LETZTE NEU;
663                  ≠FOR≠ PN :- PN.SUC ≠WHILE≠ PN =/= ≠NONE≠ ≠DO≠
664                           ≠BEGIN≠  IND := COMPARE(L,PN.LISTE);
665                                                  ≠IF≠ IND > 0 ≠THEN≠
666                           ≠GOTO≠ NICHT ISOMORPH;
667                                                  ≠IF≠ IND = 0 ≠THEN≠
668                    ≠BEGIN≠ SETLU(L.I1-1); ≠GOTO≠ IGUAL ; ≠END≠;
669                  ≠END≠    FOR PN :- PN.SUC;
670                  ≠GOTO≠ NICHT ISOMORPH;
671 IGUAL..          K2.INTO(PN.KLISTE2);
672 ≠END≠ FOR K2;
673
674         PN :- LAST;
675         ≠IF≠ PN == LETZTE NEU.SUC
676         ≠THEN≠   ≠BEGIN≠  ≠INSPECT≠ PN ≠DO≠
677                           ≠BEGIN≠  ALT := ≠TRUE≠;
678
679                           ≠END≠    INSPECT PN;
680                           ≠IF≠ SYM ≠THEN≠ ≠GOTO≠ SYM1;
681
682                           PN :- NEUKANTENPM(≠NONE≠);
683                           PN.INTO(≠THIS≠  KANTENPARTITION);
684                           PA :- PA.SUC;
685                    ≠COMMENT≠ D.H. DIE ZUGEHOERIGE,TRANSPONIERTE
686                                    PARTITIONSMENGE;
687                           ≠INSPECT≠ PN ≠DO≠
688                           ≠BEGIN≠  KLISTE1 :- PA.KLISTE1;
689                                    KLISTE2 :- PA.KLISTE2;
690
691                                    ALT := ≠TRUE≠;
692                           ≠END≠    INSPECT PN;
693                     ≠INSPECT≠ PA ≠DO≠ KLISTE1:-KLISTE2:-≠NONE≠;
694 SYM1..   ;        ≠END≠ THEN
695
696         ≠ELSE≠   ≠BEGIN≠  NEU := ≠TRUE≠;
697 ≠IF≠ ZEITLIMIT < E LAPSED ≠THEN≠ ABORT(≠≠ZEITLIMIT≠≠);
698
699
700                           PV :- LETZTE NEU;
701                  ≠FOR≠ PV :- PV.SUC ≠WHILE≠ PV =/= ≠NONE≠ ≠DO≠
702                           ≠INSPECT≠ PV ≠DO≠
703                       ≠IF≠ KLISTE1.CARDINAL = KLISTE2.CARDINAL
704                           ≠THEN≠
705                           ≠ELSE≠   ≠GOTO≠ NICHT ISOMORPH;
706                           ≠IF≠ SYM ≠THEN≠ ≠GOTO≠ SYM2;
707
708                           PA :- PA.SUC;
```

```
709
710                                 PV :- LETZTE NEU;
711                         ≠FOR≠ PV :- PV.SUC ≠WHILE≠ PV =/= ≠NONE≠ ≠DO≠
712                                 ≠INSPECT≠ NEUKANTENPM(≠NONE≠) ≠DO≠
713                                 ≠BEGIN≠
714                                           K1 :- PV.KLISTE1.FIRST;
715                                           K2 :- PV.KLISTE2.FIRST;
716                                           ≠WHILE≠ K1 =/= ≠NONE≠ ≠DO≠
717 ≠BEGIN≠ K1.T.INTO(KLISTE1);
718         K2.T.INTO(KLISTE2);
719         K1  :- K1.SUC;                    K2 :- K2.SUC;
720 ≠END≠   WHILE K1 =/= NONE;
721                                           FOLLOW(PV);
722                                         PV :- ≠THIS≠ PARTITIONSMENGE;
723                                 ≠END≠     INSPECT NEUKANTENPM;
724 SYM2..   ;          ≠END≠     ELSE;
725
726 E1..    ≠IF≠ PA =/= LETZTE ALT
727         ≠THEN≠ ≠BEGIN≠ PA :- PA.SUC; ≠GOTO≠ B1    ≠END≠;
728
729 B2..    PA :- FIRST;
730         PA.INTO(RESERVE);
731         ≠IF≠ PA =/= LETZTE ALT ≠THEN≠ ≠GOTO≠ B2;
732
733         PN :- FIRST;
734         ≠WHILE≠ PN =/= ≠NONE≠ ≠DO≠
735         ≠BEGIN≠  K1 :- PN.KLISTE1.FIRST;
736                  K2 :- PN.KLISTE2.FIRST;
737                  ≠WHILE≠ K1 =/= ≠NONE≠ ≠DO≠
738                  ≠BEGIN≠  K1.PM :- PN;    K1 :- K1.SUC;
739                           K2.PM :- PN;    K2 :- K2.SUC;
740                  ≠END≠ WHILE K1 =/= NONE;
741                  PN :- PN.SUC;
742         ≠END≠    WHILE PN =/= NONE;
743
744         PN :- FIRST;       ≠WHILE≠ PN =/= ≠NONE≠ ≠DO≠
745         ≠BEGIN≠  PN.INDEX := I := I+1;
746                  PV :- PN.KLISTE1.FIRST≠QUA≠KANTE.T.PM;
747                  ≠IF≠ PV =/= PN ≠THEN≠
748              ≠BEGIN≠  ≠IF≠ PV =/= PN.SUC ≠THEN≠ PV.FOLLOW(PN);
749                           PV.INDEX := I := I+1;
750                           PN :- PV;
751                  ≠END≠ THEN;
752                  PN :- PN.S UC;
753         ≠END≠ WHILE PN;
754         ZEIT :=(ELAPSED-ZEIT)/100.0;      OUTCHAR($/$);
755         OUTFRAC(ZEIT,1,≠IF≠ZEIT<10.0≠THEN≠3≠ELSE≠TRUNK(LN(ZEIT
                                                    )/LN(10.0)
756            )+2);
757         DETACH;
758         ≠GOTO≠ WEITER;
759
760 NICHT ISOMORPH..
761         NON ISOMORPH := ≠TRUE≠;
762 ≠END≠   KANTENPARTITION;
763
764
765
766 ≠REF≠ (PARTITIONSMENGE) PM,PN,PV;          ≠REF≠ (KNOTEN) K1,
                                                K2,K3;
```

```
767 ≠INTEGER≠ X1,X2,I,N1                                                    ;
768 ≠BOOLEAN≠ KN1;
769 ≠INTEGER≠ ≠ARRAY≠  FELD1,FELD2[1..N,1..N];
770 ≠INTEGER≠ ≠ARRAY≠ PERM[1..N];
771 ≠REF≠(HEAD) QLISTE;
772 QLISTE :- ≠NEW≠ QUADRUPELLISTE;
773
774 ≠IF≠ ¬ ENDIMAGE ≠THEN≠ ≠BEGIN≠ ≠INSPECT≠ SYSIN ≠DO≠
775 ≠IF≠  IMAGE.SUB(POS,1).GETCHAR = $*$ ≠THEN≠
776   STARRED POLYGON(FELD1,N,SYMMETRISCH)
777 ≠ELSE≠ ZUFALLSGRAPH(FELD1,N,  X,SYMMETRISCH,SCHLEIFEN); ≠END≠
                                                            ≠ELSE≠
778 ≠IF≠ EINLESEN ≠THEN≠ EINLESENG(FELD1,N,SYMMETRISCH)                    ;
779 !!!IMAGE;  ≠IF≠ SYSIN.IMAGE.SUB(1,2) = ≠≠ID≠≠
780 ≠THEN≠ ≠BEGIN≠ ZUFALLSPERMUTATION(PERM,N);    PERMUTIERE(FELD1,
                                                            PERM,
781                       FELD2,N)  ≠END≠
782 ≠ELSE≠
783 ≠IF≠ ¬ ENDIMAGE ≠THEN≠ ≠BEGIN≠ ≠INSPECT≠ SYSIN ≠DO≠
784 ≠IF≠ IMAGE.SUB(POS,1).GETCHAR = $*$ ≠THEN≠
785   STARRED POLYGON(FELD2,N,SYMMETRISCH)
786 ≠ELSE≠ ZUFALLSGRAPH(FELD2,N,  X,SYMMETRISCH,SCHLEIFEN); ≠END≠
                                                            ≠ELSE≠
787 ≠IF≠ EINLESEN ≠THEN≠ EINLESENG(FELD2,N,SYMMETRISCH)                    ;
788 SYSIN.SETPOS(80);
789
790 KNOTP :- ≠NEW≠ KNOTENPARTITION;    KANTP :- ≠NEW≠ KANTENPARTIT
                                                            ION;
791
792 GRAPH1 :- ≠NEW≠ GRAPH(FELD1,N);    GRAPH2 :- ≠NEW≠ GRAPH(FELD2
                                                            ,N);
793
794 ≠FOR≠ PM :- KNOTP.FIRST, KANTP.FIRST ≠DO≠≠WHILE≠PM=/=≠NONE≠≠DO
                                                            ≠
795  ≠IF≠ PM.KLISTE1.CARDINAL ≠NE≠ PM.KLISTE2.CARDINAL
796  ≠THEN≠ ≠GOTO≠ NICHT ISO
797  ≠ELSE≠ PM :- PM.SUC;
798
799 PM :- KANTP.FIRST;
800 ≠WHILE≠ PM =/= ≠NONE≠ ≠DO≠
801 ≠BEGIN≠ PN :- PM.KLISTE1.FIRST≠QUA≠KANTE.T.PM;
802          ≠IF≠ PN =/= PM ≠THEN≠ PN.FOLLOW(PM);
803          PM :- PN.SUC;
804 ≠END≠ WHILE PM;
805
806
807 I := 0;
808 ≠FOR≠ PM :- KANTP.FIRST, KNOTP.FIRST≠DO≠≠WHILE≠PM=/=≠NONE≠ ≠DO
                                                            ≠
809  ≠INSPECT≠ PM ≠DO≠
810 ≠BEGIN≠
811          INDEX :- I := I+1;
812          PM :- SUC;
813 ≠END≠ INSPECT PM;
814
815
816                                                         ≠COMMENT≠
817
818      *****  START KNOTENPARTITION  *****                          ;
819
```

```
820 W.. CALL(KNOTP);
821 ≠IF≠ KNOTP.NON ISOMORPH ≠THEN≠ ≠GOTO≠ NICHT ISO;
822 ≠IF≠ KNOTP.CARDINAL + SINGLES.CARDINAL = N ≠THEN≠ ≠GOTO≠ ISO1;
823
824                                                          ≠COMMENT≠
825
826     *****  START DER KANTENPARTITION  *****                        ;
827
828 CALL(KANTP);
829 ≠IF≠ KANTP.NON ISOMORPH ≠THEN≠ ≠GOTO≠ NICHT ISO;
830 ≠IF≠ KANTP .NEU ≠THEN≠ ≠GOTO≠ W;
831
832 ≠FOR≠ PM :- KNOTP.FIRST, KANTP.FIRST ≠DO≠≠WHILE≠ PM =/= ≠NONE≠
                                                          ≠DO≠
833  ≠INSPECT≠ PM ≠DO≠
834 ≠BEGIN≠ GANZ ALT := ALT := ≠FALSE≠; PM :- SUC          ≠END≠;
835
836
837 UGL :- ≠NEW≠ HEAD;
838 DECOMPOSE(KNOTP,UGL);
839 ≠IF≠ UGL.CARDINAL = 1 ≠THEN≠ ≠GOTO≠ NZERL;
840 OUTIMAGE; OUTTEXT(≠≠ZERLEGUNG IN UNABHENGIGE UNTERGRAPHEN: ≠≠)
                                                          ;
841 NZERL..
842 SG :- UGL.FIRST;    ≠WHILE≠ SG =/= ≠NONE≠ ≠DO≠
843      ≠IF≠ (≠IF≠SG.P.FIRST.SUC == ≠NONE≠ ≠THEN≠ SG.P.FIRST≠QUA≠
844                PARTITIONSMENGE.COMPLETE ≠ELSE≠ ≠FALSE≠ ) ≠THEN≠
845                   ≠BEGIN≠  OUTIMAGE; OUTIMAGE;
846                      OUTTEXT(≠≠VOLLSTAENDIG:≠≠); AUSGABE(SG.P);
847                   ≠END≠THEN    ≠ELSE≠
848 ≠BEGIN≠
849          OUTIMAGE; OUTIMAGE;
850          SG.ISOTEST2;          SG :- SG.SUC;
851 ≠END≠ WHILE SG;
852 ≠IF≠ SINGLES.EMPTY ≠THEN≠ ≠GOTO≠ ENDE;
853 OUTIMAGE; OUTTEXT(≠≠FIXPUNKTE: ≠≠);          AUSGABE(SINGLES);
854 ≠GOTO≠ ENDE;
855
856 ISO1..
857 OUTIMAGE; OUTTEXT(≠≠EINZIGER ISOMORPHISMUS: ≠≠);
858 AUSGABE(SINGLES);  AUSGABE(KNOTP);
859 ≠GOTO≠ ENDE;
860 NICHT ISO..OUTIMAGE;OUTTEXT(≠≠DIE BEIDEN GRAPHEN SIND NICHT IS
                                                          OMORPH≠≠);
861 OUTIMAGE;
862 ENDE..    ;
863 ≠END≠;
864 ≠IF≠ ¬ LASTITEM ≠THEN≠ ≠GOTO≠ ANF;
865 ≠END≠;
866           FINIS
```

Beispiele, die mit diesem Programm gerechnet wurden.

In der jeweils mittleren Zeile steht ein Stern für eine Durchführung von A_1, ein Schrägstrich für eine Durchführung von A_2. Dazwischen stehen die benötigten Rechenzeiten in Sekunden.
Man sieht, daß bei "unregelmäßigen" Graphen gewöhnlich die Anwendung von A_1 ausreicht.

Die Knotenpartition, die durch f erzeugt wird, ist in der Form $M_j:M'_j$; ausgedruckt.

Die Schreibweise [11,12,15,16] : [12,18,16,15] für einen Isomorphismus p ist folgendermaßen zu lesen:
p(11)=12, p(12)=18 und so weiter der Reihe nach.

Graph von Beispiel 4.12:

```
 N= 18  SYMMETRISCHER GRAPH, EINGELESEN: >>>>>>
1: 2,3,4,5,6,7,8,9*    2: 3,8,9,10,11,12,17*    3:  4,7,10,11,12,13*
4:  5,6,11,12,13,14*    5:  6,9,12,13,14,15*    6:  7,13,14,15,16*
7: 8,14,15,16,17*       8:  9,10,15,16,17*      9:  10,11,16,17*
10:  11,14,17,18*       11:   12,15,18*   12:  13,16,18*   13:  14,17,
18*
14:  15,18*   15:  16,18*   16:  17,18*      17:  18*


ZUFALLSPERMUTATION, U=105
  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
 17  9 14  4 11  6  3 13  2  5 18 12  8  1 15 16 10  7
*0.4/8.5***1.5/8.1*0.5/5.5


 4  ISOMORPHISMEN MOEGLICH; EINER DAVON:
[11,12,15,16]:[12,18,16,15]; [ 3, 7]:[14, 3]; [ 5, 9]:[ 2,11]; [10,
13,14,17]:[ 8, 5,10, 1]; [ 2, 4, 6, 8]:[ 4, 9,13, 6];
FIXPUNKTE:
[ 1]:[17]; [18]:[ 7];
```

Graph von Beispiel 4.13:

Hier ist ein Graph $K \in H(\Delta, G_1, .., G_k)$, dessen Grad die Knoten 12 und 13 auszeichnen soll, nicht leicht zu finden. Auch die Knoten-Quotientengraphen von $[1]_K[2]$ können die Knoten 12, 13 nicht von den Knoten 9,1o,11,14,15,16 unterscheiden.

```
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
 N= 16  SYMMETRISCHER GRAPH, EINGELESEN: >>>>>>
1:9,10,11,12*2:9,10,11,13*3:9,12,14,15*4:10,12,14,16*5:11,12,15,16*
6:9,13,14,15*7:10,13,14,16*8:11,13,15,16*

ZUFALLSPERMUTATION, U=83
  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
  5  2  3  1 14  6  7  8 16  4 11 10 12 13 15  9
*0,3/6,6**0,6/6,2**0,6/4,6*0,3/3,1

KEINE AUSSAGE UEBER ISOMORPHISMUS
RESULTIERENDE KNOTENPARTITION:
[12,13]:[10,12]:  [ 1, 2, 3, 4, 5, 6, 7, 8]:[ 1, 2, 3, 5, 6, 7, 8,14]
[ 9,10,11,14,15,16]:[ 4, 9,11,13,15,16]:
```

Der folgende Graph besteht aus den beiden getrennten Komponenten 1 bis 8 und 9 bis 16.

```
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
 N= 16  SYMMETRISCHER GRAPH, EINGELESEN: >>>>>>
1:3,4,6,7*2:4,5,6,7*3:5,6,8*4:7,8*5:7,8*6:8*9:11,12,14,15*10:12,13,15,
16*
11:13,14,16*12:14,15,15*13:15,16*14:16*

ZUFALLSPERMUTATION, U=71
  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
  5 10  7  6  1 11  3  8  9  2 14 12 13  4 15 16
*0,3/6,3***1,1/7,0*0,4/4,6
ZERLEGUNG IN UNABHENGIGE UNTERGRAPHEN:

 4  ISOMORPHISMEN MOEGLICH; EINER DAVON:
[ 2, 4, 6, 8]:[ 6,10, 8,11]:  [ 1, 5]:[ 1, 5]:  [ 3, 7]:[ 7, 3]:

KEINE AUSSAGE UEBER ISOMORPHISMUS
RESULTIERENDE KNOTENPARTITION:
[ 9,10,11,12,13,14,15,16]:[ 2, 4, 9,12,13,14,15,16]:
```

```
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
 N=  7  SYMMETRISCHER GRAPH, EINGELFSEN: >>>>>>
1:2,6,7*2:1,6,7*3:4,5,7*4:3,5,7*5:3,4,7*6:1,2,7*7:1,2,3,4,5,6*


ZUFALLSPERMUTATION, U=1
  1   2   3   4   5   6   7
  7   2   3   1   5   4   6
**0.3/0.5


KEINE AUSSAGE UEBER ISOMORPHISMUS
RESULTIERENDE KNOTENPARTITION:
[ 1, 2, 3, 4, 5, 6]:[ 1, 2, 3, 4, 5, 7];
FIXPUNKTE:
[ 7]:[ 6];
```

```
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
 N=  7  SYMMETRISCHER GRAPH, EINGELFSEN: >>>>>>
1:2,6,7*2:1,6,7*3:4,5,7*4:3,5,7*5:3,4,7*6:1,2,7*7:1,2,3,4,5,6*

>>>>>>
1:2,6,7*2:1,3,7*3:2,4,7*4:3,5,7*5:4,6,7*6:1,5,7*7:1,2,3,4,5,6*

**0.2
DIE BEIDEN GRAPHEN SIND NICHT ISOMORPH
```

```
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
 N=  9 GRAPH, EINGELESEN: >>>>>>
1:3,6,9,7*2:1,3,4,7*3:4,5,8,9*4:2,3,5,7*5:1,3,6,8*6:1,2,4,8*7:2,6,8,9*
8:4,5,6,9*9:1,2,5,7*

>>>>>>
1:2,5,6,9*2:1,6,7,9*3:2,4,7,8*4:1,2,3,5*5:4,6,8,9*6:2,3,7,8*7:3,4,5,6*
8:1,5,7,9*9:1,3,4,8*

*0.2/1.6**0.6
EINZIGER ISOMORPHISMUS:
[ 6]:[ 4];  [ 1]:[ 5];  [ 7]:[ 9];  [ 2]:[ 1];  [ 8]:[ 3];  [ 9]:[ 8];
[ 4]:[ 2];  [ 3]:[ 6];  [ 5]:[ 7];
```

```
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
 N= 13 GRAPH, EINGELESEN:  STARRED POLYGON, SYMBOL=( 2, 5, 6)
 STARRED POLYGON, SYMBOL=( 4,10,12)
*0.2/2.8*0.2/1.7


KEINE AUSSAGE UEBER ISOMORPHISMUS
RESULTIERENDE KNOTENPARTITION:
[ 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13]:[ 1, 2, 3, 4, 5, 6, 7, 8, 9,
10,11,12,13];
```

```
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
 N= 17 GRAPH, EINGELESEN:  STARRED POLYGON, SYMBOL=( 1, 3)
 STARRED POLYGON, SYMBOL=( 2, 6)
*0.3/6.8*0.3/6.1*0.4/4.2


17  ISOMORPHISMEN MOEGLICH; EINER DAVON:
[ 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,16,17]:[ 1, 3, 5, 7, 9,
11,13,15,17, 2, 4, 6, 8,10,12,14,16];
```

```
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
 N= 13 ZUFALLSGRAPH, KANTENHAEUFIGKEIT=0.5, U=          1111
ZUFALLSPERMUTATION, U=4605595
  1  2  3  4  5  6  7  8  9 10 11 12 13
 12  2  9  1  5  6 11  7 10  3 13  4  8
***).6
EINZIGER ISOMORPHISMUS:
[ 6]:[ 6];  [ 8]:[ 7];  [10]:[ 3];  [13]:[ 8];  [ 9]:[10];  [ 3]:[ 9];
[ 7]:[11];  [ 5]:[ 5];  [ 4]:[ 11];  [ 1]:[12];  [11]:[13];  [12]:[ 4];
[ 2]:[ 2];
```

```
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
 N= 13 ZUFALLSGRAPH, KANTENHAEUFIGKEIT=0.5, U=          3333
ZUFALLSPERMUTATION, U=1778361
  1  2  3  4  5  6  7  8  9 10 11 12 13
  1  2  3  4  5  6 10  8  9 12 11  7 13
**1.1
EINZIGER ISOMORPHISMUS:
[ 3]:[ 3];  [ 5]:[ 5];  [ 9]:[ 9];  [11]:[11];  [12]:[ 7];  [ 2]:[ 2];
[ 4]:[ 4];  [ 6]:[ 6];  [ 7]:[10];  [13]:[13];  [10]:[12];  [ 8]:[ 8];
[ 1]:[ 1];
```

```
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
 N= 13 ZUFALLSGRAPH, KANTENHAEUFIGKEIT=0.5, U=          5555
ZUFALLSPERMUTATION, U=7339335
  1  2  3  4  5  6  7  8  9 10 11 12 13
  1  2  8 11  5  6 13  3  9 10  4 12  7
***1.1
EINZIGER ISOMORPHISMUS:
[11]:[ 4];  [ 5]:[ 5];  [ 7]:[13];  [ 1]:[ 1];  [10]:[10];  [ 6]:[ 6];
[ 3]:[ 8];  [ 9]:[ 9];  [12]:[12];  [ 4]:[11];  [13]:[ 7];  [ 2]:[ 2];
[ 8]:[ 3];
```

```
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
 N= 18 ZUFALLSGRAPH, KANTENHAEUFIGKEIT=0.3, U=          4565
 1: 2, 5, 6,10,18*   2: 2, 3, 5, 9,13*   3: 2, 5, 8,14,16*   4:15,16,
18*   5: 1, 5, 6*   6: 5, 7, 8, 9,10,11,14,18*   7:10,11,15*   8: 2, 6
, 8,16*   9: 7, 9,12,15*  10:13*  11: 2, 3,10,11,12,13,17,18*  12: 2,
 3, 7, 8,14*  13: 5, 6,11,12,13,15,17*  14: 5,16*  15: 1, 2, 6*  16: 4
, 8,17*  17: 2,10,13,17,18*  18: 2, 6, 7, 8,17*
ZUFALLSPERMUTATION, U=1123069
  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
  5  2  3  4  1  6 18  8  9 10 11 12 13 14 15 16 17  7
***2.4
EINZIGER ISOMORPHISMUS:
[ 6]:[ 6]:  [13]:[13]:  [18]:[ 7]:  [17]:[17]:  [ 8]:[ 8]:  [16]:[16]:
[ 5]:[ 1]:  [11]:[11]:  [ 3]:[ 3]:  [ 1]:[ 5]:  [ 2]:[ 2]:  [ 4]:[ 4]:
[12]:[12]:  [ 9]:[ 9]:  [14]:[14]:  [10]:[10]:  [ 7]:[18]:  [15]:[15]:
```

```
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
 N= 18  SYMMETRISCHER ZUFALLSGRAPH, KANTENHAEUFIGKEIT=0.5, U=
          4565
 1: 2, 5,10,15,18*   2: 1, 3, 5, 8, 9,11,12,15,17*   3: 2, 5, 8,11,14,
16*   4:15,16,18*   5: 1, 2, 3, 5, 6,13,14*   6: 5, 7, 8, 9,10,11,13,
15,18*   7: 6,10,12,15,18*   8: 2, 3, 6, 8,16,18*   9: 2, 6, 9,12,15*
10: 1, 6, 7,11,13,17*  11: 2, 3, 6,10,11,12,13,17,18*  12: 2, 7, 9,11,
13,14*  13: 5, 6,10,11,12,13,15,17*  14: 3, 5,12*  15: 1, 2, 4, 6, 7,
 9,13*  16: 3, 4, 8,17*  17: 2,10,11,13,16,17,18*  18: 1, 4, 6, 7, 8,
11,17*
ZUFALLSPERMUTATION, U=7463069
  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
 11  6  3  4 12  7  2  8 16 10  1 18 13 14 15  9 17  5
****2.1
EINZIGER ISOMORPHISMUS:
[11]:[ 1]:  [ 2]:[ 6]:  [ 6]:[ 7]:  [15]:[15]:  [18]:[ 5]:  [13]:[13]:
[ 3]:[ 3]:  [12]:[18]:  [ 5]:[12]:  [17]:[17]:  [10]:[10]:  [ 8]:[ 8]:
[16]:[ 9]:  [ 9]:[16]:  [ 4]:[ 4]:  [14]:[14]:  [ 1]:[11]:  [ 7]:[ 2]:
```

```
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
 N= 18  SYMMETRISCHER ZUFALLSGRAPH, KANTENHAEUFIGKEIT=0.3, U=
          4565
 1: 2, 5,10,15*   2: 1, 3, 5, 8, 9,15,17*   3: 2, 8,11,16*   4:16,18*
 5: 1, 2, 5, 6,13*   6: 5, 8,10,11,18*   7:10,15,18*   8: 2, 3, 6, 8,
16,18*   9: 2,15*  10: 1, 6, 7,11,13,17*  11: 3, 6,10,11,12,13*  12:11
,13*  13: 5,10,11,12*  14*  15: 1, 2, 7, 9*  16: 3, 4, 8,17*  17: 2,10
,16,18*  18: 4, 6, 7, 8,17*
ZUFALLSPERMUTATION, U=3511241
  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
  1 15  3  6 12  9 18  8  2 13  5  7 10  4 17 16 14 11
****1.9
EINZIGER ISOMORPHISMUS:
[ 2]:[15]:  [10]:[13]:  [ 7]:[18]:  [14]:[ 4]:  [18]:[11]:  [11]:[ 5]:
[ 8]:[ 8]:  [ 6]:[ 9]:  [15]:[17]:  [16]:[16]:  [13]:[10]:  [ 5]:[12]:
[ 1]:[ 1]:  [ 3]:[ 3]:  [17]:[14]:  [ 9]:[ 2]:  [12]:[ 7]:  [ 4]:[ 6]:
```