プログラムのページ

担当　鈴　木　誠　道

7308　グラフのすべての木を求めるプログラム

菱沼千明（慶応義塾大学大学院工学研究科）

グラフの木は位相幾何学的な概念であるが，代数的に表わすと取扱いやすい．すなわち，一つの木をその枝の積で表わし，すべての独立な木の集合はその積の和（木積和）で表わしておく．こうすると，任意のグラフ**G**のすべての木を表わす木積和は，4つの場合に分かれ，つぎのように表わされる[1]．

$$T=\sum_{b_k\in \mathbf{B}_k} b_k$$
$$=T_0'+\left(\sum_{b_k\in \mathbf{B}_k} b_k\right)\cdot T_s'$$
$$=\left(\sum_{b_k\in \mathbf{B}_k} b\right)\cdot T_0'$$
$$=0 \qquad (1)$$

ここで，$\mathbf{B}_k=\{b_i, b_j, \cdots, b_s\}$ は**G**の中で互に両端が一致した枝（並列枝）b_k からなる集合（1本の枝のみの場合もある）を表わし，Σはこれらの要素の総和を意味する．また，T_0' は $\mathbf{B}_k$ の枝をすべて取り除いたグラフ $\mathbf{G}_0'$ の木積和であり，T_s' はこれらの枝の両端を一致させたグラフ $\mathbf{G}_s'$ の木積和である．

文献 1) に示されたすべての木を見出すアルゴリズムは式(1)に基づいて再帰的にTを求めるものである．**G**の木は多項式Tの積の項に表われてくるが，式(1)の掛け算をし，展開した形よりすべての木を求めると，グラフが大きい場合には記憶領域を莫大に必要とし，その数は指数関数的以上の割合で増す．

ここに紹介するプログラムは，基本的には文献 1) に示されたアルゴリズムと LISP 関数の定義に従うが，計算の過程で項の展開をせずにすべての木を求めるものである．つまり，いったんこの展開しない式に対応した二進木リストを作り，次にこれを走査してすべての木を一つずつ印刷する．この結果，必要とする記憶領域も文献 1) に示されたアルゴリズムの手間と同じオーダーにおさえられている．

データの表現形式

グラフを LISP がS式として直接扱えるように完全かっこ式で表現する．その形式はつぎのように定義される．

<graph>::=(<g-list>),
<g-list>::=<p-branch>|
　　<p-branch><g-list>,
<p-branch>::=(<node><node><b-list>),
<b-list>::=<branch>|<branch><b-list>,
<node>::=節点の名前を示す原始記号,
<branch>::=枝の名前を示す原始記号.

つまり，1本の枝あるいは並列な枝とその両端の節点を示す <p-branch> をサブリストとするリストで全体の枝集合，すなわちグラフが表現される．

これに対し，計算の途中で作り出される二進木リストはS式で表わすと，つぎの形式をしている．

<tree-list>::=(<t-list>),
<t-list>::=<empty>|<t-list>
　　((<b-list>)<t-list>).

プログラムと実行結果

プログラムの作成には，TOSBAC 3400/30 の KLISP システム[2]を使用し，GLISP 言語で書いてある．図1に全プログラムと評価式を示す．実行時のスタックアップ，および余分なアソシエーションリストの増加をおさえるために，ほとんどの関数はプログラム機能で定義している．以下の説明では，リストxが前述のグラフの表現形式に基づいているときグラフxと呼び，他の場合も同様である．

trees [x]：グラフxのすべての木を印刷する主関数．

treelist [x]：式(1)に従ってグラフxのすべての木を表わす二進木リストを作る．ここで，xの最初の項目 (car [x]) は任意に選ばれた並列枝（式(1)の $\mathbf{B}_k$）とその両端の節点を表わす．

gs [　]：グラフ**G**を示すxが外部変数で与えられ，これに対し $\mathbf{G}_s'$ を示すリストを値とする．

paratest [x; y]：節点xがグラフyに含まれるならば真，そうでなければ偽となる関数．

out [x; z]：<tree-list> を表わすリストxを走査して，木の枝名を印刷する．zは以前に捜し出された木の枝のリストである．

```
1    TREES[ X ]
2      = OUT[TREELIST[X];NIL] ;
3    *
4    OUT[ X ; Z ]
5      = PROG[ [Y] ;
6            [ NULL[X] -> RETURN[PRINT[Z]] ] ;
7        X;  Y := CAAR[X] ;
8        Y;  OUT[CDAR[X];CONS[CAR[Y];Z]] ;
9            [ Y:=CDR[Y] -> GO [Y] ;
10             X:=CDR[X] -> GO [X] ]
11           ]  ;
12   *
13   TREELIST[ X ]
14     = [ NULL[CDR[X]] -> LIST[LIST[CDDR[CAR[X]]]] ;
15         PARATEST[CAAR[X];CDR[X]]
16           -> [ PARATEST[CADAR[X];CDR[X]]
17                  -> NCONC[TREELIST[CDR[X]];
18                                LIST[CONS[CDDR[CAR[X]];TREELIST[GS[]]]]] ;
19                T -> LIST[CONS[CDDR[CAR[X]];TREELIST[CDR[X]]]] ] ;
20         PARATEST[CADAR[X];CDR[X]]
21              -> LIST[CONS[CDDR[CAR[X]];TREELIST[CDR[X]]]] ;
22         T -> NIL ] ;
23   *
24   GS[ ]
25     = PROG[ [Y;Z] ;
26           Z := Y := SUBST[CAAR[X];CADAR[X];CDR[X]] ;
27       X;  X := Y ;
28       Y;  [ NULL[CDR[X]] -> [ Y:=CDR[Y] -> GO [X] ;
29                               T -> RETURN[Z] ] ;
30             AND[EQ[CAAR[Y];CAADR[X]];EQ[CADAR[Y];CADR[CADR[X]]]] -> GO [Z] ;
31             AND[EQ[CAAR[Y];CADR[CADR[X]]];EQ[CADAR[Y];CAADR[X]]] -> GO [Z] ] ;
32           X := CDR[X] ;
33           GO [Y] ;
34       Z;  NCONC[CAR[Y];CDDR[CADR[X]]] ;
35           RPLACD[X;CDDR[X]] ;
36           GO [Y] ]  ;
37   *
38   PARATEST[ X ; Y ]
39     = PROG[ [] ;
40       X;  [ EQ[X;CAAR[Y]] -> RETURN[T] ;
41             EQ[X;CADAR[Y]] -> RETURN[T] ;
42             Y:=CDR[Y] -> GO [X] ] ]   .
43   *
44   TREES[((A B B1)(A C B2)(A D B3)(B C B4)(B D B5)(C D B6)) ] ;
45   *
46   TREES[((N0 N1 B1 B7)(N1 N2 B2)(N1 N3 B3)(N0 N3 B4)(N2 N3 B5 B6))] .
```

図 1 プログラムと評価式

例として，図2に2つのグラフのすべての木を求めた結果を示す．この実行に要した正味の CPU 時間は3.08秒であった．また，プログラムおよび評価式を主記憶装置に格納するのに要した LISP のセルの数は435個であった．(図2は次ページ)

おわりに，このプログラムの作成にあたり，多くの助言を賜わった慶応義塾大学中西正和氏に深謝いたします．

参考文献

1) 菱沼千明：記号回路解析に適したすべての木の生成方法，情報処理，Vol. 14, No. 12, pp. 935—941, Dec. (1973).
2) 中西正和：KLISP 説明書改訂版，慶応義塾大学情報科学研究所，昭和45年.

(昭和48年10月3日受付)

```
 FUNCTION EVALQUOTE HAS BEEN ENTERED, ARGUMENTS..
TREES
(((A B B1) (A C B2) (A D B3) (B C B4) (B D B5) (C D B6)))

(B6 B5 B3)
(B5 B4 B3)
(B6 B4 B3)
(B5 B4 B2)
(B4 B3 B2)
(B5 B3 B2)
(B4 B6 B2)
(B5 B6 B2)
(B6 B3 B1)
(B6 B5 B1)
(B3 B2 B1)
(B5 B2 B1)
(B6 B2 B1)
(B3 B4 B1)
(B5 B4 B1)
(B6 B4 B1)

 END OF EVALQUOTE, VALUE IS..
NIL


 FUNCTION EVALQUOTE HAS BEEN ENTERED, ARGUMENTS..
TREES
(((N0 N1 B1 B7) (N1 N2 B2) (N1 N3 B3) (N0 N3 B4) (N2 N3 B5 B6)))

(B5 B4 B3)
(B6 B4 B3)
(B4 B3 B2)
(B4 B5 B2)
(B4 B6 B2)
(B5 B3 B1)
(B6 B3 B1)
(B5 B4 B1)
(B6 B4 B1)
(B3 B2 B1)
(B4 B2 B1)
(B5 B2 B1)
(B6 B2 B1)
(B5 B3 B7)
(B6 B3 B7)
(B5 B4 B7)
(B6 B4 B7)
(B3 B2 B7)
(B4 B2 B7)
(B5 B2 B7)
(B6 B2 B7)

 END OF EVALQUOTE, VALUE IS..
NIL


RUN TIME   0 MIN    3080 MSEC

      **** END OF KLISP JOB ****


JOB TIME   0 MIN   10220 MSEC

   0 GARBAGE COLLECTION(S)
  66 DEPTH OF STACK USED
```

図2 実 行 結 果