ma the ma tisch | cen trum

REKENAFDELING

NR 16/71 MEI

R.P. VAN DE RIET, G. NOGAREDE
A NOTE ON AUTOMATIC STORAGE OF
ARBITRARY TREES IN ALGOL 60

amsterdam 1971

stichting
mathematisch
centrum

MC

REKENAFDELING NR 16/71 MEI

R.P. VAN DE RIET, G. NOGAREDE
A NOTE ON AUTOMATIC STORAGE OF
ARBITRARY TREES IN ALGOL 60

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

0. Summary

In this note a method is demonstrated for making an elegant ALGOL 60 program which stores a tree in a one-dimensional array.

1. The problem

Consider a set s of trees t; each tree being either an atom (representable by an integer) or the root of an arbitrary number n(n > 0) of other trees t_i, i = 1,...,n, $t_i \in s$.
The trees are assumed to be pure in the sense that no two trees are subtrees of each other.

It is supposed that the trees have to be stored once, have to be inspected often, but will not be changed or deleted.
In that case the following storage organization may be chosen.
Declare the integer array *INF*[1: *bound*].
If t is an atom, then store the integer representing this atom in *INF*[*t*], where *t* is some appropriate value, called the location of t; otherwise, if t is the root of n trees t_i, with locations t_i, i = 1,...,n, fill n+1 consecutive array elements *INF*[*t*],...,*INF*[*t*+n] as follows:

INF[*t*] := *t*+n+1 and
for i = 1,...,n: if t_i is a root then *INF*[*t*+i] := t_i, otherwise
t+i = t_i and *INF*[*t*+i] contains the integer representing the atom t_i,

where *t* is again some appropriate value, called the location of t.

A tree consisting of N atoms and M roots will occupy N+2M places in *INF*. For the storage of this tree a program has to be written which performs the N+2M necessary assignments. For any given tree it is not too difficult to write down these assignment statements; the order of which has to be chosen judiciously however.
The problem is to construct auxiliary procedures, which perform the assignment statements, by means of which for every arbitrary tree a program can be made with a structure reflecting directly the structure of this tree.

In section 3 the solution the solution of the problem will be given in the form of the procedures *STORE TREE* and *L*, which store an arbitrary tree consisting of at least one root.
Before doing this it is necessary to introduce, in the next section, means for treating lists with a variable number of elements as actual parameters in procedure statements.

2. Lists of variable length as parameter

Consider a set of lists, each list consisting of a variable number n of elements e_i, i = 1,...,n; these elements have to be treated by a standard process P in the ordering $e_1,\ldots,e_n$. Before treating the elements, an initialization process P_0 has to be executed which uses the number n.

A way of programming this in ALGOL 60, where it is not allowed to have a variable number of actual parameters in a procedure statement, is the following:

Declare the procedure *L*:

```
Boolean procedure L(first,ei); Boolean first; integer ei;
begin comment count the number of elements; cnt := cnt+1;
      if first then P0;
      P; L := false
end L
```

The list $(e_1,e_2,\ldots,e_n)$ is then treated by the statement:

```
L(...L(L(true,
         e1),
      e2),
   ...
   en).
```

3. Storage of trees

A procedure *STORE TREE* and a procedure *L* are given now, by means of which the tree of the following example is stored in the array *INF*. Note that *STORE TREE* stores trees consisting of at least one root.

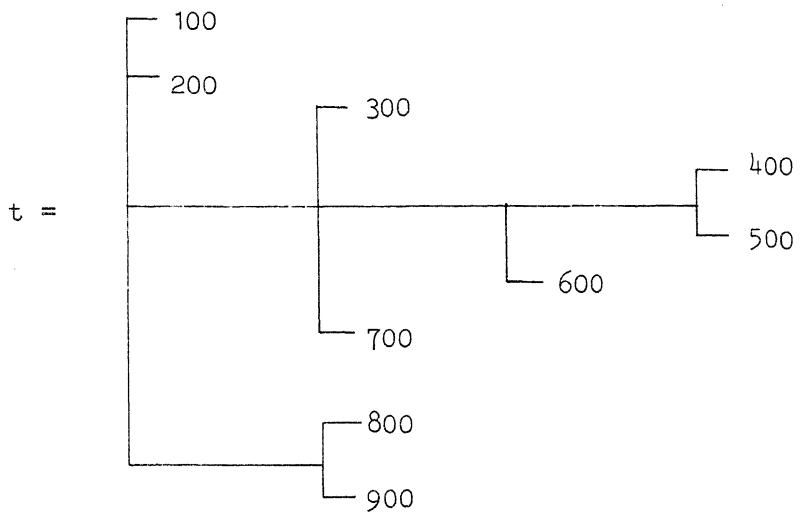
3.1. Example

In the ALGOL 60 program of section 3.2. the following tree t is stored.

In bracketed form:

t = (100,200,(300,((400,500),600),700),(800,900)).

Two-dimensionally t looks like:



3.2. The ALGOL 60 program

```
begin integer ptr,fill ptr,t;
  integer array INF[1:1000];
  integer procedure STORE TREE(list); Boolean list;
  begin STORE TREE:= fill ptr:= ptr:= ptr + 1;
    if list then;
  end STORE TREE;

  Boolean procedure L(first,inf); Boolean first; integer inf;
  begin integer loc fill ptr;
    ptr:= ptr + 1;
    if first then
    begin INF[fill ptr]:= ptr + 1;
      fill ptr:= fill ptr + 1;
      if ptr > 1000 then EXIT
    end;
    loc fill ptr:= fill ptr;
    INF[loc fill ptr]:= inf;
    fill ptr:= loc fill ptr + 1;
    L:= false
  end L;
```

```
    ptr:= 0;
    t:= STORE TREE(L(L(L(L(true,
                100),
                200),
                STORE TREE(L(L(L(true,
                      300),
                      STORE TREE(L(L(true,
                            STORE TREE(L(L(true,
                                  400),
                                  500))),
                            600))),
                      700))),
                STORE TREE(L(L(true,
                      800),
                      900))));
    NLCR; PRINTTEXT(‹the contents of INF is displayed.
    Each line contains the information of one root›);
    t:= 0; fill ptr:= 1; for t:= t + 1 while t ≤ ptr do
    begin if t = fill ptr then
      begin NLCR; fill ptr:= INF[fill ptr] end;
      ABSFIXT(2,0,t); FIXT(3,0,INF[t]); SPACE(5)
  end end
```

```
the contents of INF is displayed.
Each line contains the information of one root
  1   +6      2 +100      3 +200      4   +6      5  +16
  6  +10      7 +300      8  +10      9 +700
 10  +13     11  +13     12 +600
 13  +16     14 +400     15 +500
 16  +19     17 +800     18 +900
```