AN APPLICATION OF BINARY TREES

Vasile LUPŞE, Ovidiu COSMA

Abstract This article presents an application of binary trees in the merge of two or more lots, without requiring a previous sorting of the data.

Keywords: binary trees, search trees, graphs

Introduction

A tree is a connected graph without cycles. A tree has the following properties:

There is a node in which no arch enters, and it is named root. Excepting the root, every other node has the property that in it enters a single arch. This connects the node with another node named predecessor or parent. One or mode arches can come out of a node. Every arch connects the node to another one named successor or sun.

The nodes ore organized in several levels, the first level being occupied by the root. The nodes in the last level are named terminal nodes or leaves. No arch comes out of the leaves. The nodes can contain any type of information. This information is named the keys of the tree.

A binary tree has the property that every node has at most two successors. They are named left and right successors. The memory representation of binary trees can be made by static or dynamic allocation.

In this paper the dynamic allocation will be used. In every node both the useful information and the connections to the successors will be placed. The following data type is defined for this purpose:

```
type pnod=^nod;
nod=record
        info:integer;
        left,right:pnod;
end;
```

The useful information can be of any type, not necessary integer, as in the above example. The *left* and *right* identifiers are pointers to the left respectively right successor.

A binary tree example is presented in figure 1.

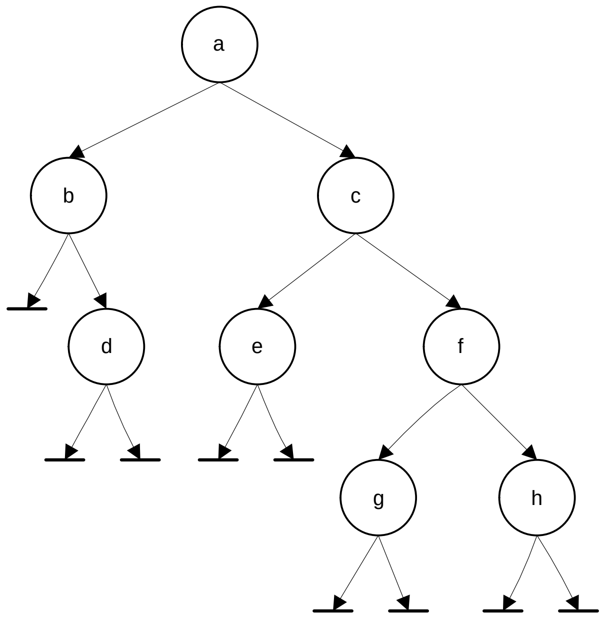

Figure1: Binary tree example.

1. The crossing of binary trees

The crossing of a binary tree, implies visiting of every node, in a certain order. A reference to the root of the tree is necessary for performing this operation. The following crossing algorithms are known:

A.) *Root – Left subtree – Right subtree*, also named pre-order crossing.
The following procedure can be used to perform this operation.

```
Procedure pre_order(p:pnod);
  begin
        if p<>nil then
                begin
                        write(p^.info);
                        pre_order(p^.left);
                        pre_order(p^.right);
                end;
  end;
```

B.) *Left subtree – Root – Right subtree*, also named in-order crossing.
The following procedure can be used to perform this operation.

```
Procedure in_order(p:pnod);
  begin
        if p<>nil then
                begin

                        in_order(p^.left);
                        write(p^.info);
                        in_order(p^.right);
                end;
  end;
```

C.) *Left subtree – Right subtree – Root*, also named post-order crossing.
The following procedure can be used to perform this operation.

```
Procedure post_oreder(p:pnod);
  begin
        if p<>nil then
                begin

                        post_order(p^.left);
                        post_order(p^.right);
                        write(p^.info);

                end;
  end;
```

2. Binary search trees

A binary search tree is a binary tree that has the property that for each node the key in the left successor is *smaller* than the key in the node and the right successor key is *greater* than the node key.

We consider that the theoretical concept is extremely simple, but the applications in the information retrieval are spectacular.

The creation of a binary search tree is performed by the next application.

```
program search_tree;
  type pnod=^nod;
  node=record
        info:integer;
        left,right:pnode;
  end;
  var rad:pnode,x:integer;
```

```
procedure create_node(var q:pnode,x:integer);
  begin
        if q<>nil then
              begin
                    if x<q^.info then
                          create_node(q^.left,x)
                    else if x>q^.info then
                          create_node(q^.right,x)
                    else
                          writeln('element exists');
              end
        else
              begin
                    new(q);
                    q^.info:=x;
                    q^.left :=nil ;
                    q^.right:=nil;
              end
  end ;

procedure pre_order(p :pnode) ;
  begin
        if p<>nil then
              begin
                    write(p^.info);
                    pre_order(p^.left);
                    pre_order(p^.right);
              end;
  end;

begin
        rad:=nil;
        write('x='); readln(x);
        while x<>0 do
              begin
                    create_node(rad,x);
                    write('x='); readln(x);
              end;
        pre_order(rad);
end.
```

Observations

a.) The keys of a binary search tree are distinct, because the presented algorithm does not allow the duplication of the same key.
b.) The smallest and the greatest of the keys can be easily retrieved because the smallest is placed in the left-most node (the left-most leave predecessor) and the greatest key is placed in the right-most node.
c.) The left-most and right-most leaves can be easily accessed starting from the root, and following only left-successor respectively right-successor connections.
d.) The information retrieval applications are based on the observation that in real situations, the data used for building the tree is usually supplied in a random order. This aspect assures that the constructed tree will not be a degenerated tree (list).

4. Possible application of search trees

The following applications of search trees are possible:

- Arrays sorting;
- Merge of two or more arrays, that do not need to be previously sorted;
- The determination of the maximum and minimum value of an array.

REFERENCES

1.Knuth D.E.,"Tratat de programarea calculatorului.Algoritmi fundamentali",Editura Tehnica,1974.
2.Knuth D.E.,"Tratat de programarea calculatoarelor"(vol 3)(Sortare si cautare),Editura Tehnica,Bucuresti,1976.
3.Livovski L.,Georgescu H.,"Sinteza si analiza algoritmilor",Bucuresti,1986
4.Manber U.,"Introduction to Algorithms ,A Creative Aproach",1989.
5.Mateescu E,Maxim I.,"Arbori",Editura Tara Fagilor,Suceava,1997.

Received: 18.05.2004

Department of Mathematics and Computer Science
Faculty of Sciences, North University of Baia Mare
E-mails: vasilelupse@yahoo.co.uk
cosma@alphanet.ro