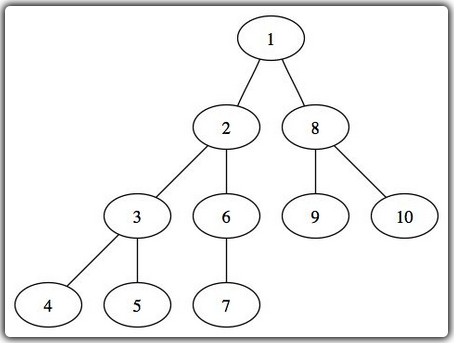
**DEPTH FIRST TREE TRAVERSAL**

Depth-first search (DFS) is an [algorithm](https://en.wikipedia.org/wiki/Algorithm) for traversing or searching [tree](https://en.wikipedia.org/wiki/Tree_data_structure) or [graph](https://en.wikipedia.org/wiki/Graph_(data_structure)) data structures. One starts at the [root](https://en.wikipedia.org/wiki/Tree_(data_structure)#Terminology) (selecting some arbitrary node as the root in the case of a graph) and explores as far as possible along each branch before [backtracking](https://en.wikipedia.org/wiki/Backtracking).

**Example of a tree**:



function DFS(Tree T ) : void

If ( not isEmpty(T) then

Manage\_root(T);

DFS(sonLeft(T));

DFS(sonRight(T));

Enf if

End function

Note: *Manage\_root() is a function that you define yourself, it is such a display function of the element that is the root*

The depth search of the previous tree gives: 1-2-3-4-5-6-7-8-9-10

**CHARACTERS IN STRINGS**

|  |  |
| --- | --- |
| version of N \* N order | version of N order |
| Function find\_chars (str1: String, str2 : String) : String  **Begin**    Variable :  len1 : integer = 0  len 2 : integer = 0  result : String = null  i ,J : integer = 0    *//the length of the two strings*  len1 = length (str1)  len2 = length (str2)  *//made the comparison character by character*  While (i < len1) do  For (j from 0 to len2 j++) then  *//if there is a match, it is added to result*  If (str1[i]==str2[j]) then  result .= str2[j]  end if  end for  end while    *//return the final result*  return result ;    end function | Function find\_chars (str1: String, str2 : String) : String  **Begin**    Variable :  len1 : integer = 0  len 2 : integer = 0  result : String = null  i ,J,k : integer = 0  already\_added : boolean    *//the length of the two strings*  len1 = length (str1)  len2 = length (str2)  *//made the comparison character by character*  While (i < len1) do  For (j from 0 to len2 j++) then  *//if there is a match, it is added to result*  If (str1[i]==str2[j]) then  already\_added = false;  for (k from 0 TO lenght($result) ; $k++) THEN    If ($str2[j] == result[k] and j != i) then  already\_added=true;  break;  end if  end for  //if the variable already\_added is false add the caractere  if($already\_added==false) then  result.= $str2[$j];  end if  end if  end if  end for  end while    *//return the final result*  return result ;    end function |

**Test:**

1. the version N\*N order of ***find\_chars*** (“*processing”,”* *processor”) will return “*prroocessss*”*
2. the version N order of ***find\_chars*** (“*processing”,”* *processor”) will return “*process*”*

**ARRAY COMPACTION**

function compact\_array(tab1: array) : array

begin

Variable :

len1 : integer = 0

tab2 : Array = null // the new array for result

i ,j : integer = 0

current : integer //tamporally variable used to keep the current value on while……do boucle

already\_added : Boolean = false

if(is\_array(tab1)) then

//the length of the array

len1 = count(tab1);

while (I < len1) do

current = tab1[i];

already\_added = false;

for(j from 0 to count(tab2) j++) then

if(tab2[j]==current) then

already\_added=true;

Enf if

End for

if(already\_added == false) then

$tab2[]= current;

End if

$i++;

End while

return tab2;

else

display ("you may give an array as parameter )";

end if

end function

**ARRAY ROTATING**

function rotate\_array(tab, n)

begin

// declaring the used variables

Variable :

Len : integer = 0

I,j : integer = 0

new\_tab : array

len= length(tab); // the length of the array to rotate

if(n > len) n = len;

new\_tab=null;

// rotating now

while(i < n) do

new\_tab [] = tab[len - (i + 1)];

i++

end while

while (j < (len - n)) do

new\_tab [] = tab [j];

j++

end while

// return the result

return new\_tab;

end function

**LEAST COMMON MULTIPLE**

function lcm(n1,n2)

begin

variable // the used variables

result : integer

I : integer = 0

Rest : integer = 0

// calculating now

result = n1 \* n2;

while(n1 > 1) do

rest = n1 MOD n2;

if(rest == 0 ) then

rest = result / n2;

break; // get out when the result found

end if

n1 = n2;

n2 = rest;

End while

return rest; // return the result

end function