**ExtraExDS\_4Trees – Exercise 3**

**Specification**

spec binary\_tree

genres b\_tree, node, label

operations

parent: node b\_tree -> node

left\_child: node b\_tree -> node

right\_child: node b\_tree -> node

label: node b\_tree -> label

create: b\_tree b\_tree -> tree

root: b\_tree -> node

makenull: b\_tree -> b\_tree endspec

min: b\_tree -> int

sum\_path: int btree -> bool

checksum: btree int node int-> bool

endspec

**Implementation**

node = record

element : label

leftchild: ^node

rightchild: ^node

parent: ^node

endrecord

label: elementtype

class binary\_tree:

private root: ^node

public ^node parent(n: node)

public ^node left\_child(n: node)

public ^node right\_child(n: node)

public label label(n: node)

public void create(leftTree, rightTree: ^node, l: label)

public ^node root()

public void makenull()

public int min(n: ^node)

public bool sum\_path(res: int, n: ^node)

private int checksum(res: int, n:^node, suma: int)

endclass

public int binary\_tree::min(n: ^node)

minnumber:int

aux:int

if(n==null) return -1

else

minnumber:=n^.element

if (n^.leftchild==null and n^.rightchild==null)

return minnumber

endif

if (n^.rightchild==null)

aux=min(n^.leftchild)

if(aux<minnumber) minnumber=aux

endif

else if(n^.leftchild==null)

aux=min(n^.rigthchild)

if(aux<minnumber) minnumber=aux

endif

else

aux= min(n^.rigthchild)

if(aux<minnumber) minnumber=aux

aux= min(n^.leftchild)

if(aux<minnumber) minnumber=aux

endif

return minnumber

Running Time: O(n) 🡪 We need to go through all the elements from the tree to see which is the smallest element from the nodes in the tree.

endmethod

public bool binary\_tree::sum\_path(res: int, n: ^node)

return checksum(res, n, 0)

endmethod

private int binary\_tree::checksum(res: int, n:^node, suma: int)

aux: bool

if(n==null) return -1

else

if( n^.leftchild==null and n^.rightchild==null)

if(suma==res) return true

else return false

else if(n^.rightchild==null)

suma+=n^.element

aux=sum(res, n^.leftchild, suma)

else if(n^.leftchild==null)

suma+=n^.element

aux=sum(res,n^.rigthchild, suma)

else

suma+=n^.element

aux=sum(res, n^.rigthchild, suma))

if(aux)return aux

else

aux= sum(res, n^.leftchild, suma))

return aux

endif

endif

return aux

endmethod

Running Time: O(n) 🡪 We need to go through all the elements from the tree because in the worst case each parent would have only a child except the last one and we would have to go through all the tree.