Algorthim sumHelper(L,p)

If L.isLast(p) then

return p.element;

s:=sumHelper(L,L.after(p))

return s+p.element();

Algorthim sum(L)

If(L.isEmpty()) then return 0;

return sumHelper(L,L.first());

Algorthim BinarySearchHelper(s,k,low,high)

If(low===high)

If(s[low]===k

Return s[low]

else

Nosunch key

else

Let mid=low+high/2

If(s[mid]===k) return s[mid]

If(s[mid]<k). return BinarySearchHelper(s,k,low,mid-1)

else BinarySearchHelper(s,k,mid+1,high)

Algorthim

Algortim. BinarySearchHelper1(s,k,low,high)

If low>high

return no such elemnets

mid :=floor(low+high)/2

if S.size()>0 && k=s[mid] then

return S[mid]

if. k<s[mid] then

high=mid-1

return BinarySearchHElper1(s,k,low,high)

else

low=mid

return BinarySearchHElper1(s,k,low,high)

Algorthim removeDuplication(L) Using positions

P:=L.first(). 1

While L.last!=p do n

removeHelper(L,p). n2

if !L.isLast(p). n

P:=L.after(p). n

Algorthim removeHelper(L,p)

q =L.after(p). 1

while !L.isLat(q) do. n

if q.element() ==p.elemnte() then n

q:=L.before(q). n

L. remove(L.after(q)). n

If !L.isLast(q) then n

q.=L.after(q). n

Algorthim sum(T)

return sumHelper(T,T.root)

PostOrder

Algorthim sumHelper(T,V )

If(T.external(V))

return 0

Else

sumL:=sumHelper(T,T.leftChiled());

sumR:=sumHelper(T,T.rightChiled());

return sumR+sumL+V.element;

PreOrder

Algorthim sumHelper(T,V )

If(T.external(V))

return 0

sum:=V.element()

sum:= sum+sumHelper(T,T.leftChiled());

sum:=sum+sumHelper(T,T.rightChiled());

return sum

Inorder

Algorthim sumHelper(T,V )

If(T.external(V))

return 0

sum:= sum+sumHelper(T,T.leftChiled());

sum:=V.element()

sum:=sum+sumHelper(T,T.rightChiled());

return sum

using internal base case

Algorthim sumHelper(T,V )

If(T.internal(V)) then

Sum:=V.element()

sum:= sum+sumHelper(T,T.leftChiled())

sum:=sum+sumHelper(T,T.rightChiled());

return sum

else return 0

using internal base case

Algorthim sumHelper(T,V )

sum:=0;

If(T.internal(V)) then

sum:=V.element()

sum:= sum+sumHelper(T,T.leftChiled())

sum:=sum+sumHelper(T,T.rightChiled());

return sum

using internal base case

Algorthim sumHelper (T,V )

sum: =0;

If(T. internal(V)) then

Sum:=V.element() +sumHelper(T,T.leftChiled())

+sumHelper(T,T.rightChiled());

return sum

Using EulerTour

Algorthim vistPostOrder(T,v,result)

result[1]=result[0]+result[2]+V.element()

Algorthim sum(T)

Return eulerTour(T,T.root)

Algorthim partioning(S)

S1:=new List(). 1

S2:=new List(). 1

half:= floor(S.size()/2). 1

While half<S.size() do. n/2

e:=S.remove(S.first). n

S1.insertLast(e); n

While !S.isEmpty() do. n

S2.inseratLast(S.remove(S.first()). n

Return (S1,S2). 1