Sum using postions

Algorthim sumHelper(L,p)

If L.isLast(p) then 1

return p.element; 1

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

return s+p.element(); n

O(n)

Algorthim sum(L)

If(L.isEmpty()) then return 0. 1

return sumHelper(L,L.first()); 1

This is Big O of O(n)

/\*\* sum using postion traversal Js code\*/

function sum(L){

return sumHelper(L,L.first())

}

function sumHelper(L,p){

if(L.isLast(p)) return p.element()

return p.element()+sumHelper(L,L.after(p))

}

Sum Using Rank

Algortim sumHelper(L,i). Sequnce List

If L.size()==i then return 0. 1 1

return L.elemAtRank(i)+ sumHelper(L,i=i+1) n n(n+1)/2

Algorthim sum(L).

If(L.size()===0) return 0; 1 1

Return sumHelper(L,0) 1 1

Big O of this using Sequnce is O(n)

Big O of this using List is O(n2)

/\*\* sum using rank traversal Js code\*/

function sum(L){

if(L.size()===0) return 0;

return sumHelper(L,0)

}

function sumHelper(L,i){

if(i==L.size()) return 0

return L.elemAtRank(i)+sumHelper(L,i=i+1)

}

Find maximum using postion

Algorthim findMaxHelper(L,p,max)

if L.isLast(p) then return max. 1

else if (p.element()>max). 1

max=p.element(). 1

return findMaxHelper(L,L.after(p),max). n

Algorthim findMax(L).

p:=L.first() 1

return findMaxHelper(L,L.after(p),p.element() 1

/\*\* Find maximum using postion \*/

function findMax(L){

let p=L.first()

return findMaxHelper(L,L.after(p),p.element())

}

function findMaxHelper(L,p,max){

if(L.isLast(p)) return max

else if(p.element()>max)

max=p.element();

return findMaxHelper(L,L.after(p),max)

}

Find maximum using rank

Algorthim finsMaxHelper(L,I,max). Sequnce List

if(i===L.size()) return max. 1. 1

else if max<L.elemAtRank(i) 1 1

max:=L.elmAtRAnk(i) n n(n+1)/2

return finsMaxHelper(L,i=i+1;max) n n

Algorthim findMax(L)

If L.size()===0 throw Error(“empty list”) 1 1

Return finsMaxHelper(L,1,L.elemAtRank(0)). 1 1

Big O of findMAx using sequnce O(n)

Big O of findMAx using List O(n2)

/\*\* Find maximum using Rank Js code\*/

function findMax(L){

if(L.size()===0) throw Error("empty list")

return findMaxHelper(L,1,L.elemAtRank(0))

}

function findMaxHelper(L,i,max){

if(i===L.size())

return max

else if(max<L.elemAtRank(i))

max=L.elemAtRank(i)

return findMaxHelper(L,i=i+1,max)

}

Algorthim sub subSetHelper(n,set,subset,start)

Set.push(subset) n

For(let i=start;i<=n;i++) n

subset.push(i)

subSetHelper(n,set,subset,start=start+1) n\*2^n

subset.pop() n

Algorthim subOfSubSet(L)

set:=[]; 1

subset:=[] 1

subSetHelper(n,set,subset,1) 1

return set 1

over all Big o of this is O(n\*2^n)

function setOfSubsets(n) {

let set = [];

let subSet = [];

subSetHelper(n, set, subSet, 1);

return set;

}

function subSetHelper(n, set, subSet, start) {

set.push(subSet);

for (let i = start; i <= n; i++) {

subSet.push(i);

subSetHelper(n, set, subSet, start+=1);

subSet.pop();

}

}

Text, letter

Description automatically generated