**Using Rank operation. Sequnce List**

**Algortim removeDuplication(L)**

**If l.isEmpty() throw Error(“empty list”)**

**For i:=0 to L.size()-1 do n n**

**cur=L.elementAtRAnk(i) n. n\*(n+1)/2**

**for j:=i+1 to L.size()-1 do n2. n2(n\*(n+1)/2)**

**if cur==L.elementAtRAnk(j) then n2. n2(n\*(n+1)/2)**

**removeAtRank(j) n2. n2(n\*(n+1)/2)**

**j:=j-1. n2. n2(n\*(n+1)/2)**

**return L;**

**Using sequnce Big O is O(n2)**

**Using list Big O is O(n4)**

**Sequnce is more faster than List to remove duplications**

function removeDuplication(L) {

for (let i = 0; i < L.size(); i++) {

let cur = L.elemAtRank(i);

for (let j = i + 1; j < L.size(); j++) {

if (cur === L.elemAtRank(j)) {

L.removeAtRank(j);

j--;

}

}

}

return L;

}

**Algorthim isPermutation(A,B) Sequnce List**

**If A.size()!=B.size() 1 1**

**return false; 1 1**

**For i:=0 to A.size()-1 do n n**

**cur=L.elementAtRAnk(i) n n\*(n+1)/2**

**if !B.contain(cur) then. n n**

**return false; 1 1**

**return true; 1 1**

**Using Sequnce Big O is O(n)**

**Using List Big O is O(n2)**

**Sequnce is more faster than List to check permutations or not**

**N.B: I added my own new contain(elem) method inside the Sequnce class and List class**

function isPermutation(A, B) {

if (A.size() !== B.size()) return false;

for (let i = 0; i < A.size(); i++) {

curA = A.elemAtRank(i);

if (!B.contain(curA)) return false;

}

return true;

}

**using position operation**

**Algortim sortRB(L)**

**if L.isEmpty() then throw Error(“empty”) O(1)**

**p:=L.first(); 1**

**elem:=p.element(). 1**

**whiel !L.isLast(p) && L.after(p)do. n**

**If elem===Blue then n**

**L.remove(p) n**

**L.insertLast(elem) 1**

**p: =L.after(p) 1**

**elem:=p.element() 1**

**return L; 1**

**Big O of this is O(n)**

function sortRB(L){

let p=L.first()

let elem=p.element();

while(!L.isLast(p) && L.after(p)){

if(elem==="Blue"){

L.remove(p)

L.insertLast(elem)

console.log(1)

}

p=L.after(p)

elem=p.element();

}

return L;

**using Rank operation**

**Algortim sortRB(L) Sequnce List**

**For i:=0 i<L.size() do. n n**

**Cur:=L.elemAtRank(i). n n(n+1)/2**

**If cur===”Blue” then n. n**

**L.removeAtRank(i). n n(n+1)/2**

**L.insertAtRank(L.size(),cur) n n(n+1)/2**

**return L 1 1**

**Using Sequnce Bog O is O(n)**

**Using List Bog O is O(n2)**

function sortRB(L){

for(let i=0;i<L.size();i++){

let cur=L.elemAtRank(i)

if(cur==="Blue"){

L.removeAtRank(i)

L.insertAtRank(L.size(),cur)

console.log(1)

}

}

}