Complejidad Temporal

LinkedList:
Método add:

	Complejidad:

`public void setDato(T dato) {`
`  this.dato = dato;`
`}`
	1
	
	BigO: O(1)

`public void setAnt(NodoDE<T> ant) {`
`  this.ant = ant;`
`}`
	1
	
	BigO: O(1)

`public NodoDE<T> getSig() {`
`  return sig;`
`}`
	1
	
	BigO: O(1)

`public void setSig(NodoDE<T> sig) {`
`  this.sig = sig;`
`}`
	1
	
	BigO: O(1)

`public void add(T valor) {`
`  if(size==0) {`
`  ini.setDato(valor);`
`  }`
`  NodoDE<T> aux=ini;`
`  while(aux.getSig() != ini) {`
`  aux=aux.getSig();`
`  }`
`  NodoDE<T> nuevo= new NodoDE<T>(valor);`
`  aux.setSig(nuevo);`
`  nuevo.setAnt(aux);`
`  nuevo.setSig(ini);`
`  ini.setAnt(nuevo);`
`  size++;`
`}`
	T()=1+1+1+n+n-1+1+1+1+1+1+1=n+9=T(n)
	1
	1
	
	1
	n
	n-1
	
	1
	1
	1
	1
	1
	1
	BigO: O(n)

LinkedList:
Método remove:

`public T remove(int pos) {`	1+1+1+1+n+1+n+n+n+1+1+1+1+1+1+1+1+1=14+2n
`T res=null;`	1
`if(pos>=0 && pos<size) {`	1
`NodoDE<T> act = ini;`	1
`boolean salir=false;`	1
`while(pos>0) {`	n+1
`act=act.getSig();`	n
`pos--;`	n
`salir=true;`	n
`}`	
`res=act.getDato();`	1
`NodoDE<T> ant=act.getAnt();`	1
`NodoDE<T> sig=act.getSig();`	1
`ant.setSig(sig);`	1
`sig.setAnt(ant);`	1
`if(!salir) {`	1
`ini=sig;`	1
`}`	
`size--;`	1
`}`	
`return res;`	1
`}`	BigO: O(n)
`public boolean remove(T dato) {`	1+O(n)+1+O(n)+1+1=4+(14+2n)+(3n+8)=26+5n
`boolean res=false;`	1
`int index=indexOf(dato);`	O(n)
`if(index>=0) {`	1
`remove(index);`	O(n)
`res=true;`	1
`}`	
`return res;`	1
`}`	BigO:O(n)
`public int indexOf(T dato) {`	1+1+1+1+n+1+n+n+1+1+1=3n+8=O(n)
`int pos=-1;`	1
`if (ini.isEmpty()) {`	1
`NodoDE<T> act=ini;`	1
`pos=0;`	1
`while(pos<size && !act.getDato().equals(dato))`	n+1
`{`	n
`act=act.getSig();`	n
`pos++;`	
`}`	1
`if(pos==size) {`	1
`pos=-1;`	
`}`	
`}`	1
`return pos;`	BigO:O(n)
`}`	

Complejidad Espacial

LinkedList:
Método add:

	Complejidad:

`public void setDato(T dato) {`
`  this.dato = dato;`
`}`
	1
	1
	
	BigO: O(1)

`public void setAnt(NodoDE<T> ant) {`
`  this.ant = ant;`
`}`
	1
	1
	
	BigO: O(1)

`public NodoDE<T> getSig() {`
`  return sig;`
`}`
	1
	1
	
	BigO: O(1)

`public void setSig(NodoDE<T> sig) {`
`  this.sig = sig;`
`}`
	1
	1
	
	BigO: O(1)

`public void add(T valor) {`
`  if(size==0) {`
`  ini.setDato(valor);`
`  }`
`  NodoDE<T> aux=ini;`
`  while(aux.getSig() != ini) {`
`  aux=aux.getSig();`
`  }`
`  NodoDE<T> nuevo= new NodoDE<T>(valor);`
`  aux.setSig(nuevo);`
`  nuevo.setAnt(aux);`
`  nuevo.setSig(ini);`
`  ini.setAnt(nuevo);`
`  size++;`
`}`
	1
	
	1
	
	1
	
	1
	
	1
	1
	1
	1
	1
	1
	10=BigO:O(1)

`public T remove(int pos) {`
`  T res=null;`
`  if(pos>=0 && pos<size) {`
`  NodoDE<T> act = ini;`
`  boolean salir=false;`
`  while(pos>0) {`
`  act=act.getSig();`
`  pos--;`
`  salir=true;`
`  }`
`  res=act.getDato();`
`  NodoDE<T> ant=act.getAnt();`
`  NodoDE<T> sig=act.getSig();`
`  ant.setSig(sig);`
`  sig.setAnt(ant);`
`  if(!salir) {`
`  ini=sig;`
`  }`
`  size--;`
`  }`
`  return res;`
`}`
	1
	0
	
	1
	1
	
	1
	1
	1
	
	1
	1
	1
	1
	1
	
	1
	
	1
	
	13=BigO:O(1)