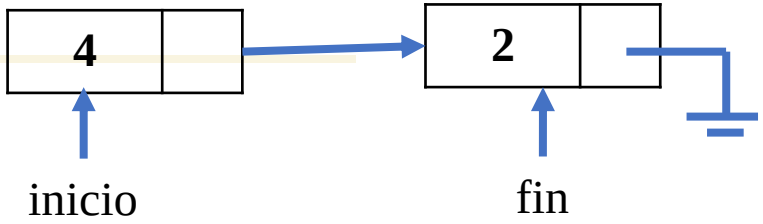
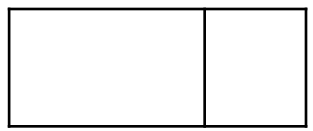
```java
public class Nodo {
    String dato;
    Nodo ap_siguiente;

    //Para agregar un nodo al final de la lista
    public Nodo(String d){
        this.dato = d;
        this.ap_siguiente = null;
    }

    //Para agregar un nodo al inicio de la lista
    public Nodo(String d, Nodo s){
        this.dato = d;
        this.ap_siguiente = s;
    }
}
```





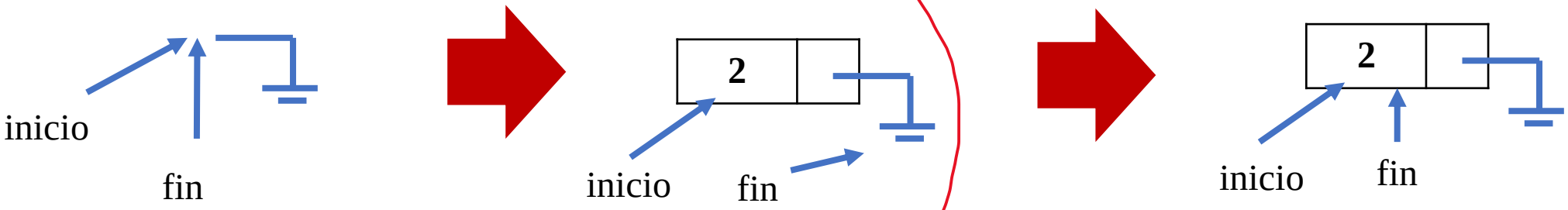
Elemento

```java
@Override
public void agregarAlInicio(String elemento) {
    //inicio apunta a null
    inicio = new Nodo( d: elemento,  s: inicio); //Ahora inicio apunta al elemento.
    if (fin == null){
        fin = inicio; //Ahora fin apunta a inicio
    }
}
```

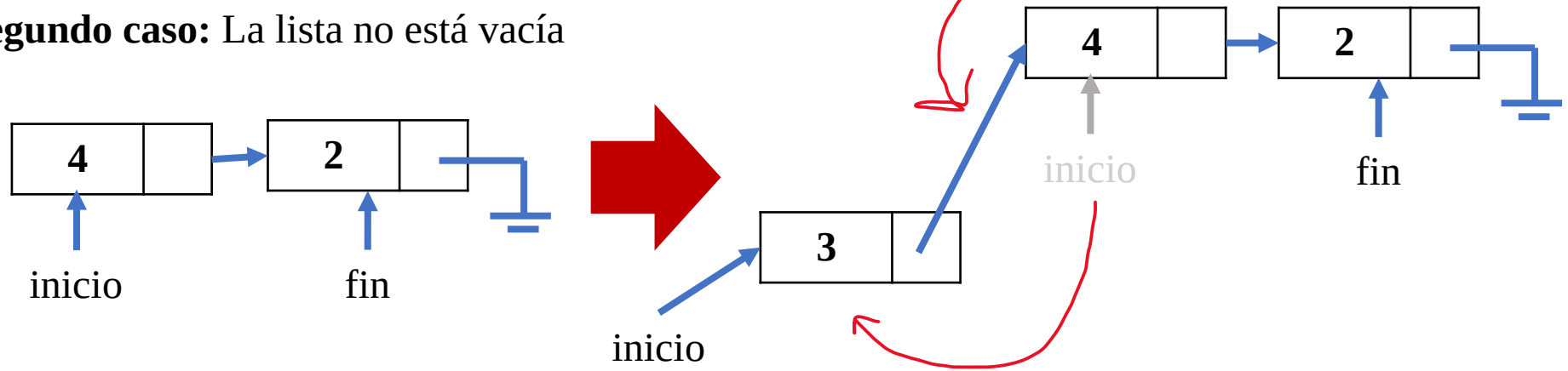
```java
0 references
public void AddFirst(string data)
{
    Node newNode = new Node(data);
    newNode.setNext(head);
    head = newNode;
}

1 reference
```

Primer caso: Lista vacía

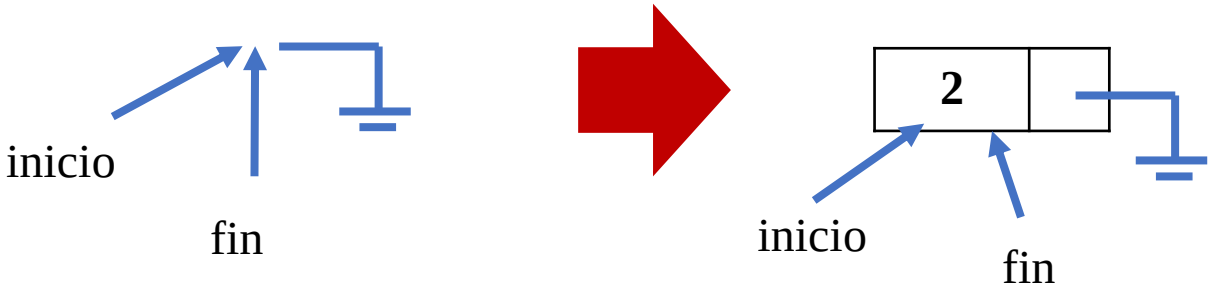


Segundo caso: La lista no está vacía

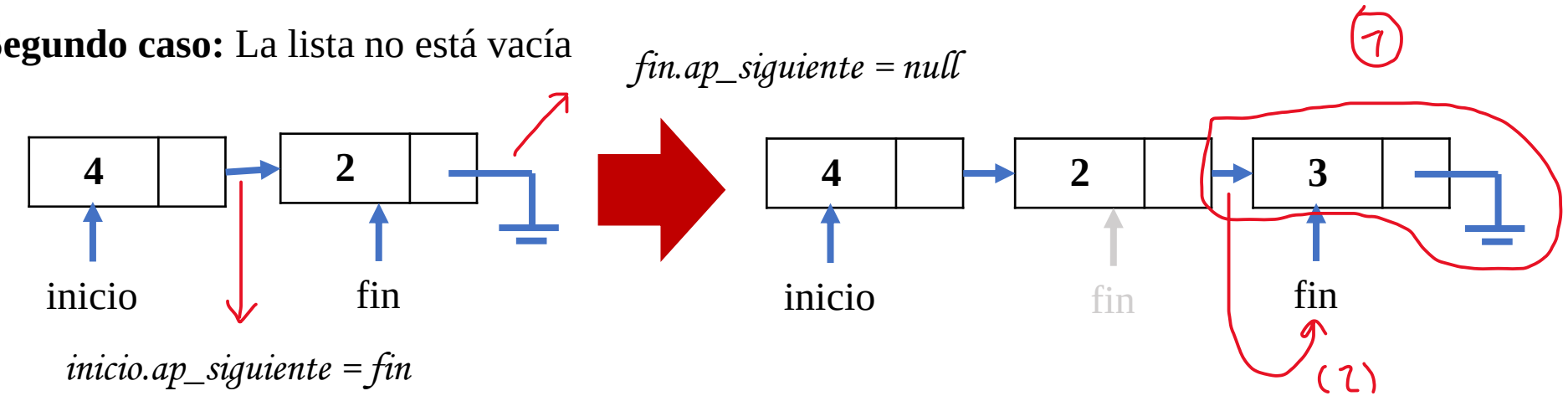


```java
@Override
public void agregarAlFinal(String elemento) {
    if(!estaVacia()){ //estaVacia: True → !estaVacia: False (No esta vacia)
        fin.ap_siguiente = new Nodo( d: elemento); //   (1)
        fin = fin.ap_siguiente;  (2)
    }else{
        inicio = fin = new Nodo( d: elemento); //inicio y fin apuntan al mismo elemento, dado que la lista esta vacia.
    }
}
```

Primer caso: Lista vacía



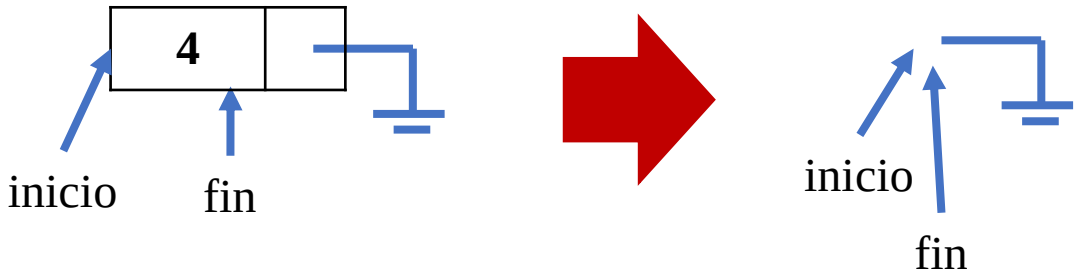
Segundo caso: La lista no está vacía



```java
@Override
public void eliminarElemento(String elemento) {
    if(!estaVacia()){
        if (inicio == fin && elemento.equals( anObject: inicio.dato)){
            inicio = fin = null;
        } else if(elemento.equals( anObject: inicio.dato)){
            inicio = inicio.ap_siguiente;
        } else{
            Nodo anterior, temporal;
            anterior = inicio;
            temporal = inicio.ap_siguiente;
            while (temporal != null && temporal.dato.equals( anObject: elemento)){
                anterior = anterior.ap_siguiente;
                temporal = temporal.ap_siguiente;
            }
            if (temporal != null){
                anterior.ap_siguiente = temporal.ap_siguiente;
                if(temporal == fin){
                    fin = anterior;
                }
            }
        }
    }
}
```

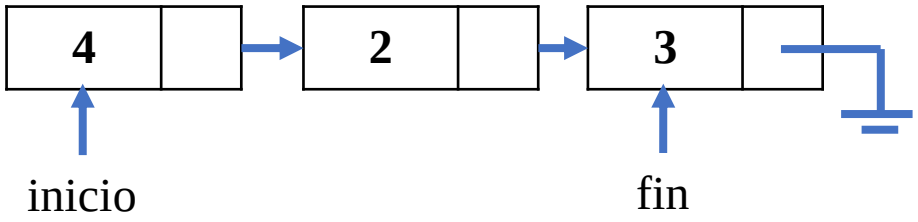
1. Si la Lista No esta vacia entonces
    1. Si inicio es igual a fin Y elemento igual a inicio de dato Entonces
        1. Apuntar inicio y fin a nulo
    2. Si No Si elemento igual a inicio de dato Entonces
        1. Apuntar inicio a inicio de siguiente
    3. Si No
        1. Crear dos Nodos, anterior y temporal
        2. Apuntar anterior a inicio
        3. Apuntar temporal a inicio de siguiente
        4. Mientras temporal sea diferente de nulo y temporal de dato diferente de elemento Hacer
            1. Apuntar anterior a anterior de siguiente
            2. Apuntar temporal a temporal de siguiente
        5. Si temporal es diferente de nulo Entonces
            1. Apuntar anterior de siguiente a temporal de siguiente
            2. Si temporal es igual a fin Entonces
                1. Apuntar fin a anterior

Primer caso

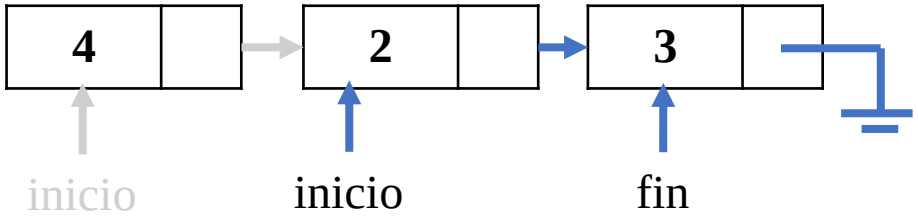


Cuando inicio y fin están en el mismo nodo.

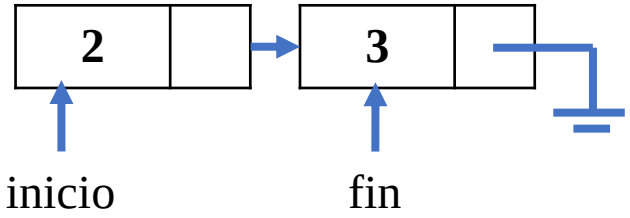
Segundo caso: elemento = 4



inicio = inicio.ap_siguiente



El nodo es eliminado por el sistema.

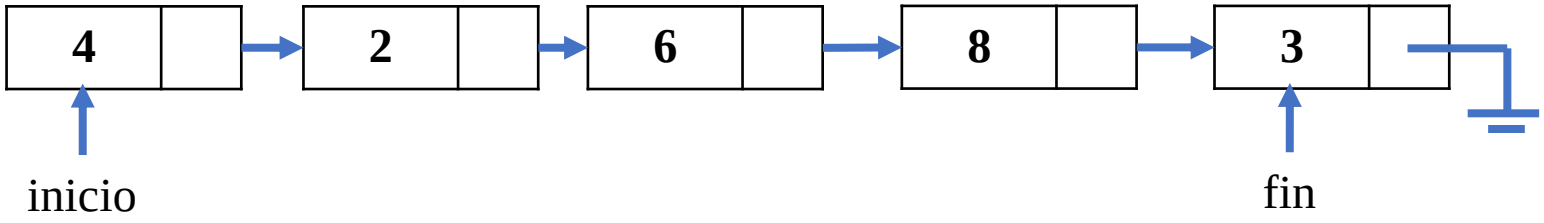


Cuando el elemento a eliminar se encuentran en el primer nodo.

1. Si la Lista No esta vacía entonces
    1. Si inicio es igual a fin Y elemento igual a inicio de dato Entonces
        1. Apuntar inicio y fin a nulo
    2. Si No Si elemento igual a inicio de dato Entonces
        1. Apuntar inicio a inicio de siguiente
    3. Si No
        1. Crear dos Nodos, anterior y temporal
        2. Apuntar anterior a inicio
        3. Apuntar temporal a inicio de siguiente
        4. Mientras temporal sea diferente de nulo y temporal de dato diferente de elemento Hacer
            1. Apuntar anterior a anterior de siguiente
            2. Apuntar temporal a temporal de siguiente
        5. Si temporal es diferente de nulo Entonces
            1. Apuntar anterior de siguiente a temporal de siguiente
            2. Si temporal es igual a fin Entonces
                1. Apuntar fin a anterior

```
108     @Override
        public void eliminarElemento(String elemento) {
110         if(!estaVacia()){
111             if (inicio == fin && elemento.equals( anObject: inicio.dato)){
112                 inicio = fin = null;
113             } else if(elemento.equals( anObject: inicio.dato)){
114                 inicio = inicio.ap_siguiente;
115             } else{
116                 Nodo anterior, temporal;
117                 anterior = inicio;
118                 temporal = inicio.ap_siguiente;
119                 while (temporal != null && temporal.dato.equals( anObject: elemento)){
120                     anterior = anterior.ap_siguiente;
121                     temporal = temporal.ap_siguiente;
122                 }
123                 if (temporal != null){
124                     anterior.ap_siguiente = temporal.ap_siguiente;
125                     if(temporal == fin){
126                         fin = anterior;
127                     }
128                 }
129             }
130         }
131     }
```

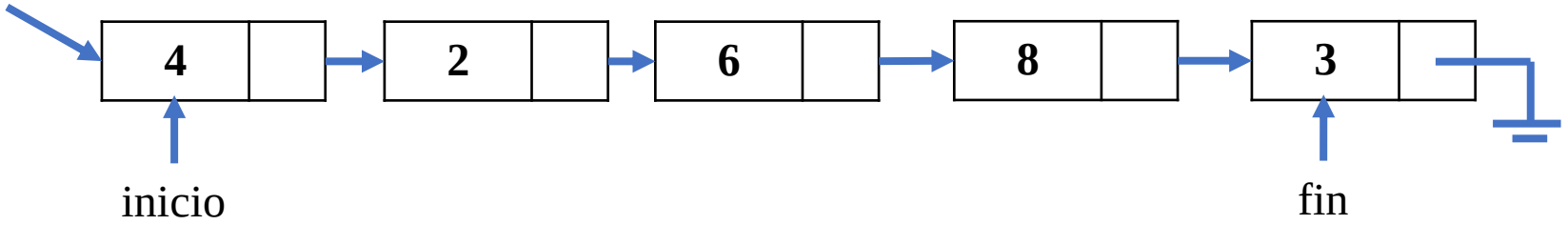
Tercer caso: elemento = 8



1. Si la Lista No esta vacía entonces
   1. Si inicio es igual a fin Y elemento igual a inicio de dato Entonces
      1. Apuntar inicio y fin a nulo
   2. Si No Si elemento igual a inicio de dato Entonces
      1. Apuntar inicio a inicio de siguiente
   3. Si No
      1. Crear dos Nodos, anterior y temporal
      2. Apuntar anterior a inicio
      3. Apuntar temporal a inicio de siguiente
      4. Mientras temporal sea diferente de nulo y temporal de dato diferente de elemento Hacer
         1. Apuntar anterior a anterior de siguiente
         2. Apuntar temporal a temporal de siguiente
      5. Si temporal es diferente de nulo Entonces
         1. Apuntar anterior de siguiente a temporal de siguiente
         2. Si temporal es igual a fin Entonces
            1. Apuntar fin a anterior

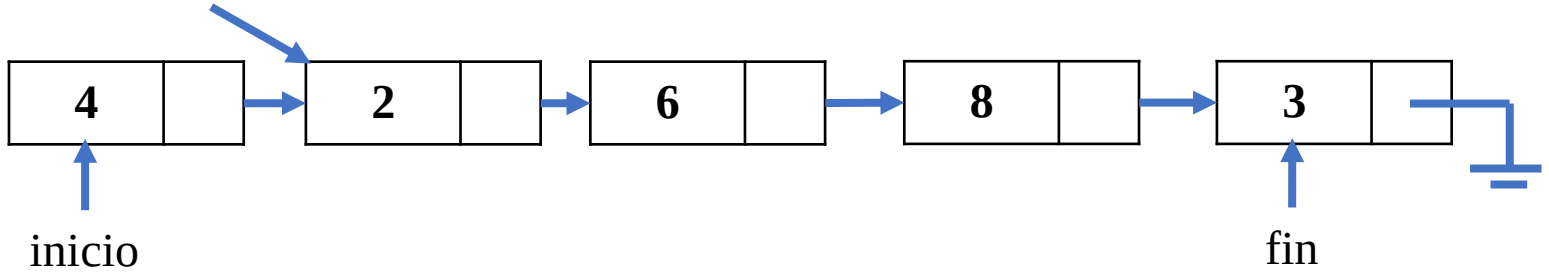
anterior = inicio

anterior

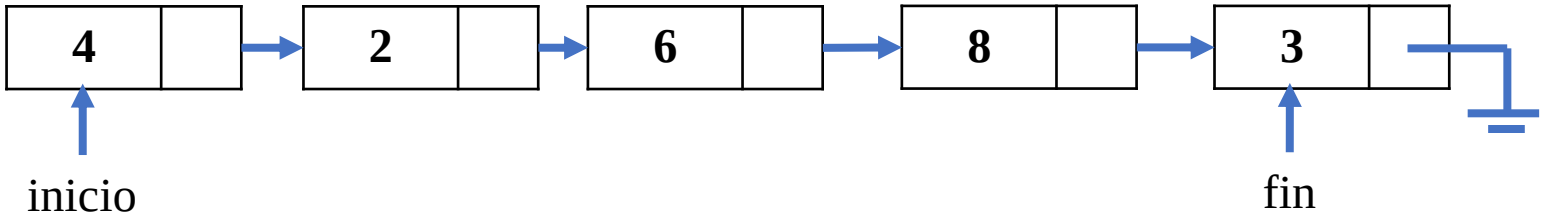


Cuando el elemento a eliminar no se encuentra en el primer nodo.

temporal

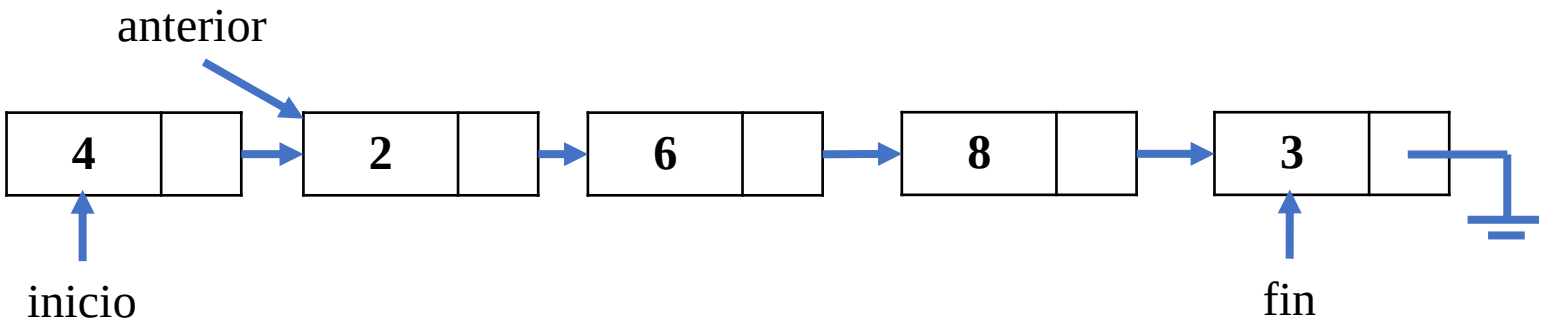


Tercer caso: elemento = 8

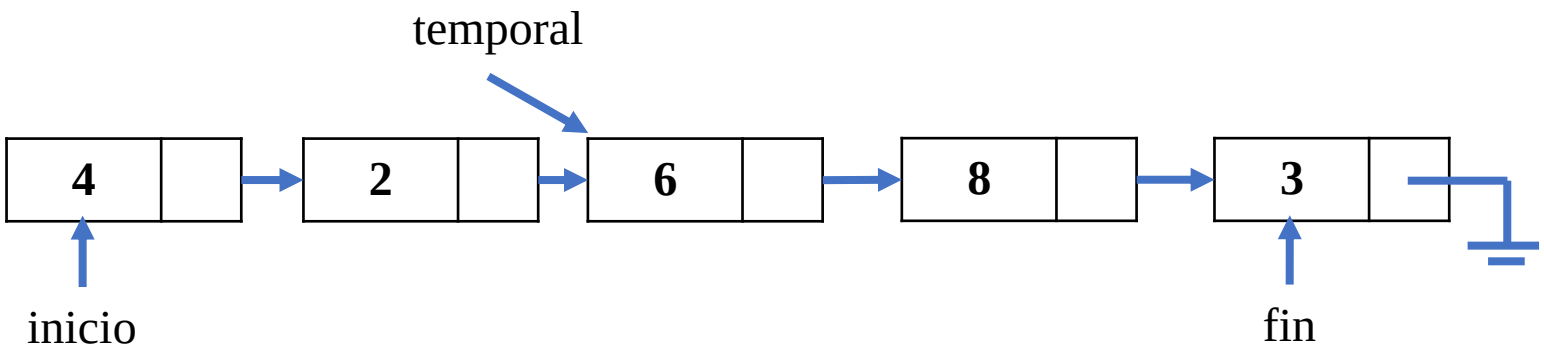


Cuando el elemento a eliminar no se encuentra en ni en el primer ni en el ultimo nodo.

Bucle: *temporal != null y temporal.dato != elemento*

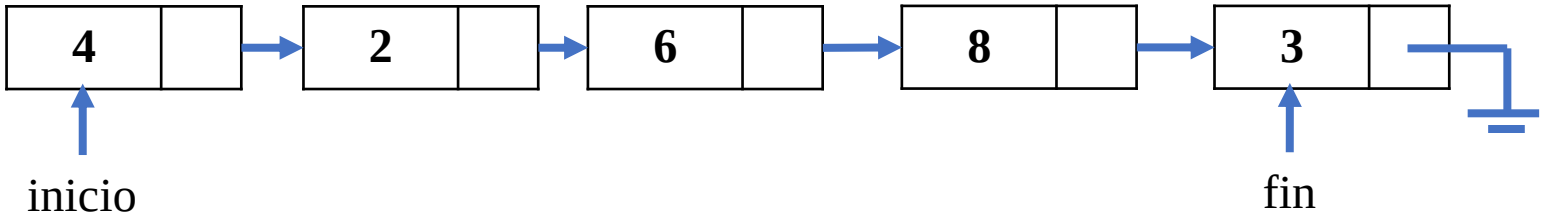


1°



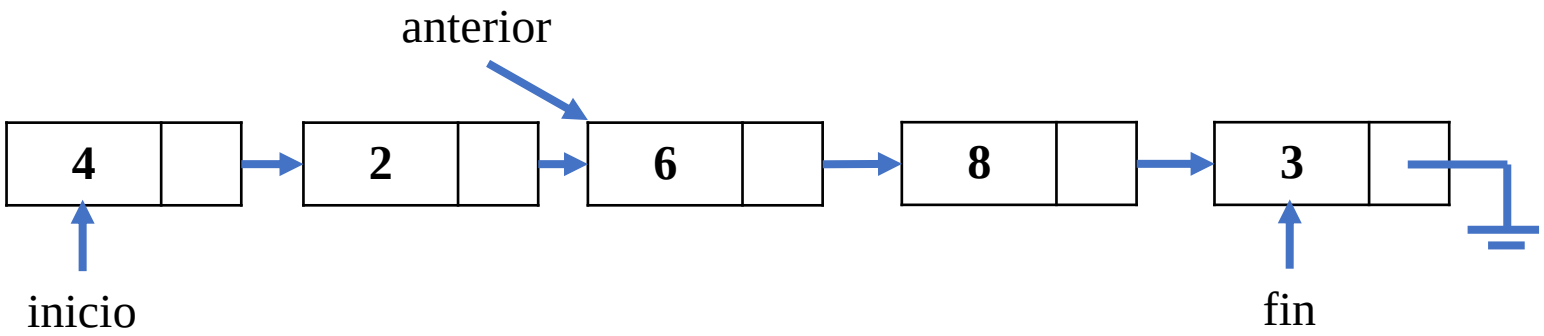
1. Si la Lista No esta vacia entonces
   1. Si inicio es igual a fin Y elemento igual a inicio de dato Entonces
      1. Apuntar inicio y fin a nulo
   2. Si No Si elemento igual a inicio de dato Entonces
      1. Apuntar inicio a inicio de siguiente
   3. Si No
      1. Crear dos Nodos, anterior y temporal
      2. Apuntar anterior a inicio
      3. Apuntar temporal a inicio de siguiente
      4. Mientras temporal sea diferente de nulo y temporal de dato diferente de elemento Hacer
         1. Apuntar anterior a anterior de siguiente
         2. Apuntar temporal a temporal de siguiente
      5. Si temporal es diferente de nulo Entonces
         1. Apuntar anterior de siguiente a temporal de siguiente
         2. Si temporal es igual a fin Entonces
            1. Apuntar fin a anterior

Tercer caso: elemento = 8

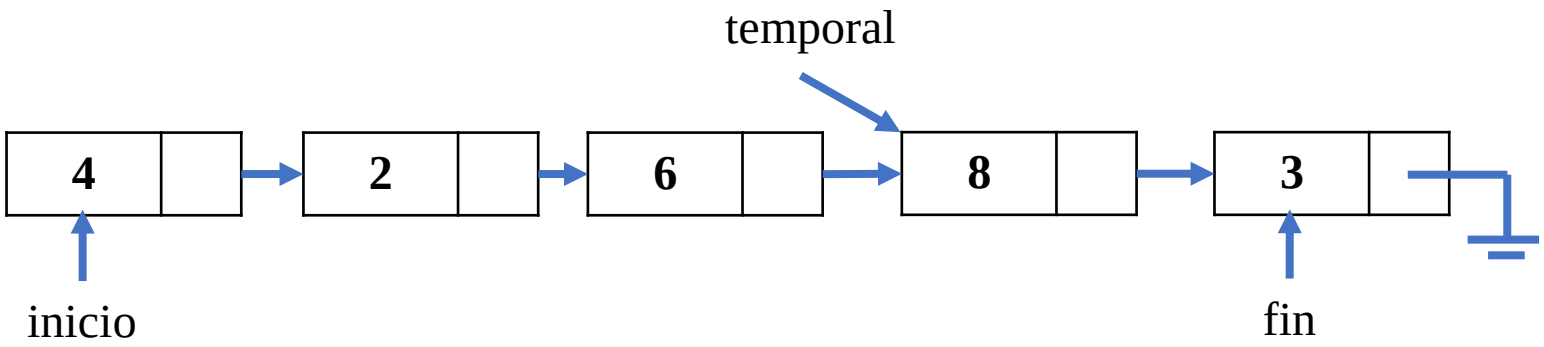


Bucle: *temporal != null y temporal.dato != elemento*

Cuando el elemento a eliminar no se encuentra en ni en el primer ni en el ultimo nodo.

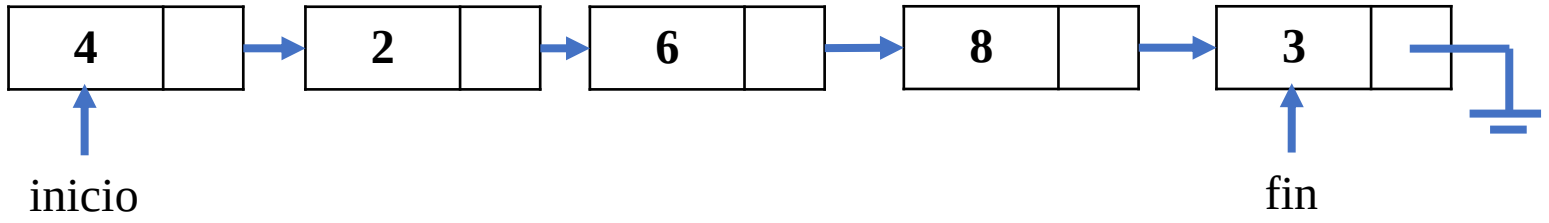


2°



1. Si la Lista No esta vacía entonces
   1. Si inicio es igual a fin Y elemento igual a inicio de dato Entonces
      1. Apuntar inicio y fin a nulo
   2. Si No Si elemento igual a inicio de dato Entonces
      1. Apuntar inicio a inicio de siguiente
   3. Si No
      1. Crear dos Nodos, anterior y temporal
      2. Apuntar anterior a inicio
      3. Apuntar temporal a inicio de siguiente
      4. Mientras temporal sea diferente de nulo y temporal de dato diferente de elemento Hacer
         1. Apuntar anterior a anterior de siguiente
         2. Apuntar temporal a temporal de siguiente
      5. Si temporal es diferente de nulo Entonces
         1. Apuntar anterior de siguiente a temporal de siguiente
         2. Si temporal es igual a fin Entonces
            1. Apuntar fin a anterior

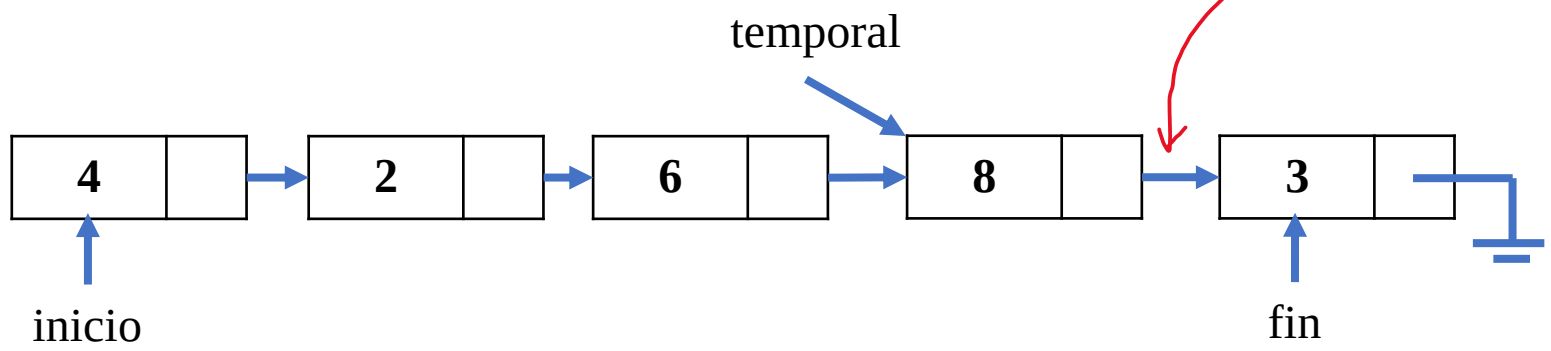
Tercer caso: elemento = 8



Cuando el elemento a eliminar no se encuentra en ni en el primer ni en el ultimo nodo.

Bucle: *temporal != null y temporal.dato != elemento*

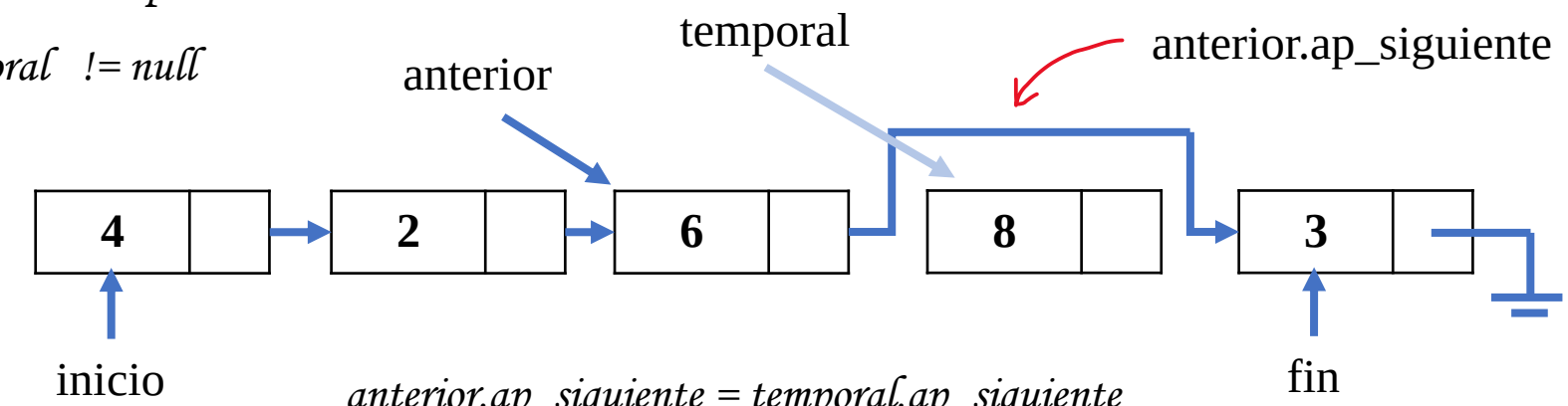
2°



1. Si la Lista No esta vacia entonces
   1. Si inicio es igual a fin Y elemento igual a inicio de dato Entonces
      1. Apuntar inicio y fin a nulo
   2. Si No Si elemento igual a inicio de dato Entonces
      1. Apuntar inicio a inicio de siguiente
   3. Si No
      1. Crear dos Nodos, anterior y temporal
      2. Apuntar anterior a inicio
      3. Apuntar temporal a inicio de siguiente
      4. Mientras temporal sea diferente de nulo y temporal de dato diferente de elemento Hacer
         1. Apuntar anterior a anterior de siguiente
         2. Apuntar temporal a temporal de siguiente
      5. Si temporal es diferente de nulo Entonces
         1. Apuntar anterior de siguiente a temporal de siguiente
         2. Si temporal es igual a fin Entonces
            1. Apuntar fin a anterior

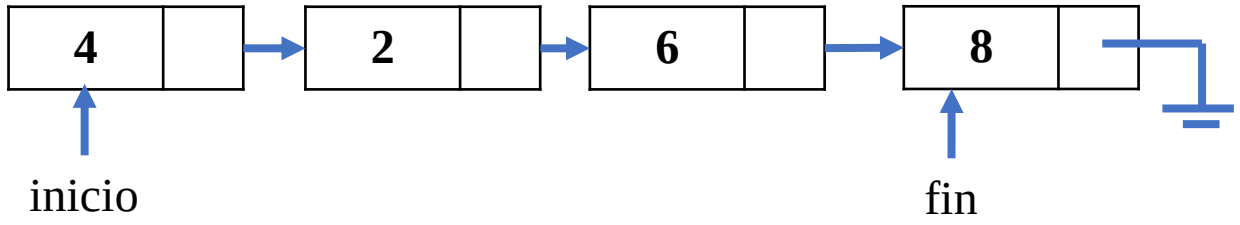
Fin del bucle: *temporal.dato == elemento*

Si: *temporal != null*



anterior.ap_siguiente = temporal.ap_siguiente

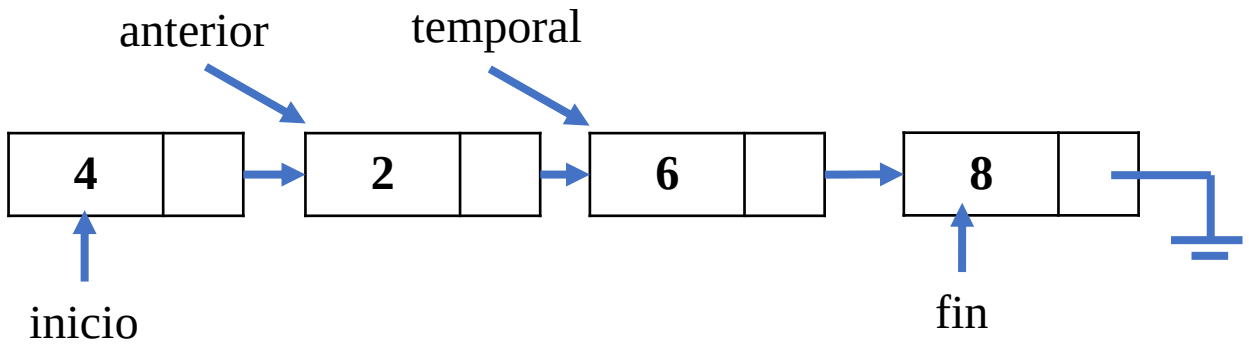
Cuarto caso: elemento = 8



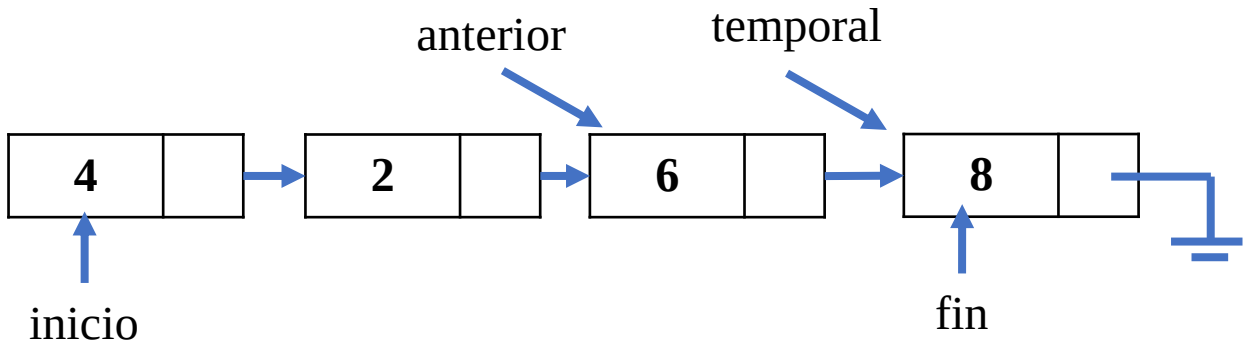
Cuando el elemento a eliminar se encuentra en el ultimo nodo.

Bucle: *temporal != null y temporal.dato != elemento*

1°



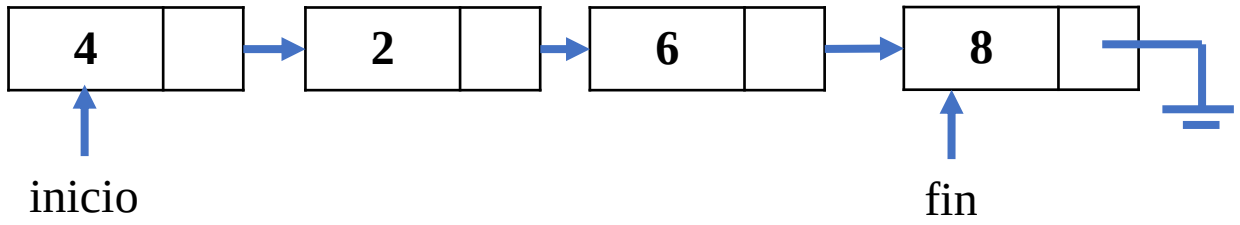
2°



1. Si la Lista No esta vacia entonces
    1. Si inicio es igual a fin Y elemento igual a inicio de dato Entonces
        1. Apuntar inicio y fin a nulo
    2. Si No Si elemento igual a inicio de dato Entonces
        1. Apuntar inicio a inicio de siguiente
    3. Si No
        1. Crear dos Nodos, anterior y temporal
        2. Apuntar anterior a inicio
        3. Apuntar temporal a inicio de siguiente
        4. Mientras temporal sea diferente de nulo y temporal de dato diferente de elemento Hacer
            1. Apuntar anterior a anterior de siguiente
            2. Apuntar temporal a temporal de siguiente
        5. Si temporal es diferente de nulo Entonces
            1. Apuntar anterior de siguiente a temporal de siguiente
            2. Si temporal es igual a fin Entonces
                1. Apuntar fin a anterior

Fin del bucle: *temporal.dato == elemento*

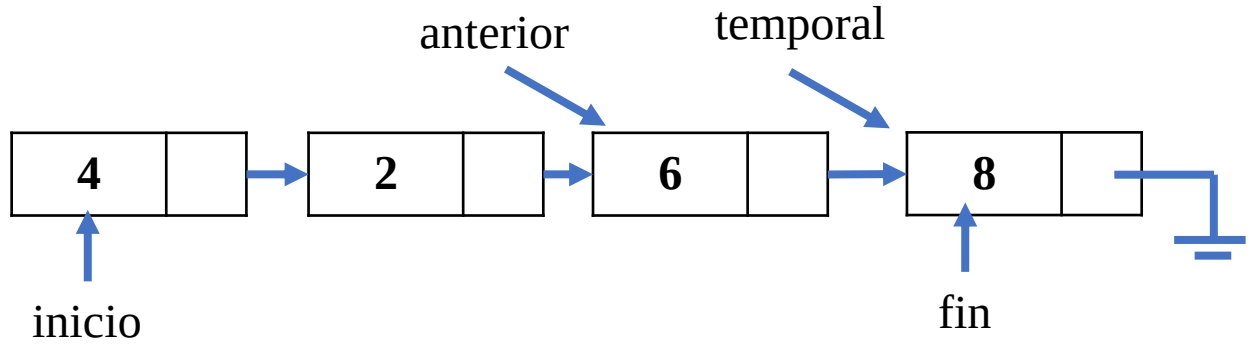
Cuarto caso: elemento = 8



Cuando el elemento a eliminar se encuentra en el ultimo nodo.

Bucle: *temporal != null y temporal.dato != elemento*

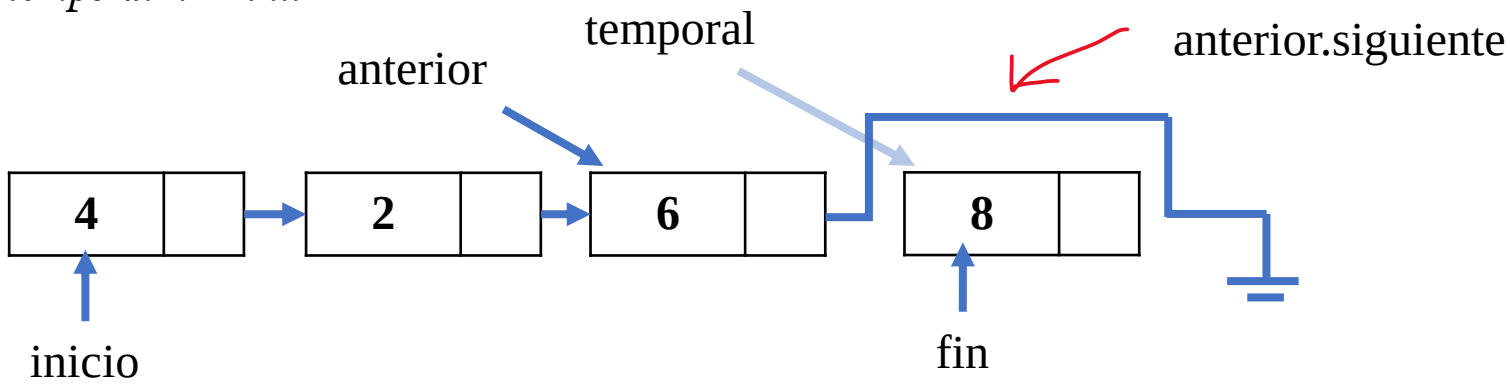
2°



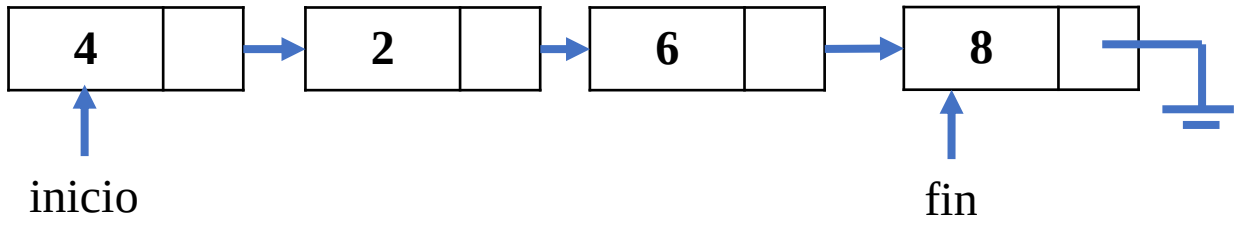
1. Si la Lista No esta vacía entonces
   1. Si inicio es igual a fin Y elemento igual a inicio de dato Entonces
      1. Apuntar inicio y fin a nulo
   2. Si No Si elemento igual a inicio de dato Entonces
      1. Apuntar inicio a inicio de siguiente
   3. Si No
      1. Crear dos Nodos, anterior y temporal
      2. Apuntar anterior a inicio
      3. Apuntar temporal a inicio de siguiente
      4. Mientras temporal sea diferente de nulo y temporal de dato diferente de elemento Hacer
         1. Apuntar anterior a anterior de siguiente
         2. Apuntar temporal a temporal de siguiente
      5. Si temporal es diferente de nulo Entonces
         1. Apuntar anterior de siguiente a temporal de siguiente
         2. Si temporal es igual a fin Entonces
            1. Apuntar fin a anterior

Fin del bucle: *temporal.dato == elemento*

Si: *temporal != null*



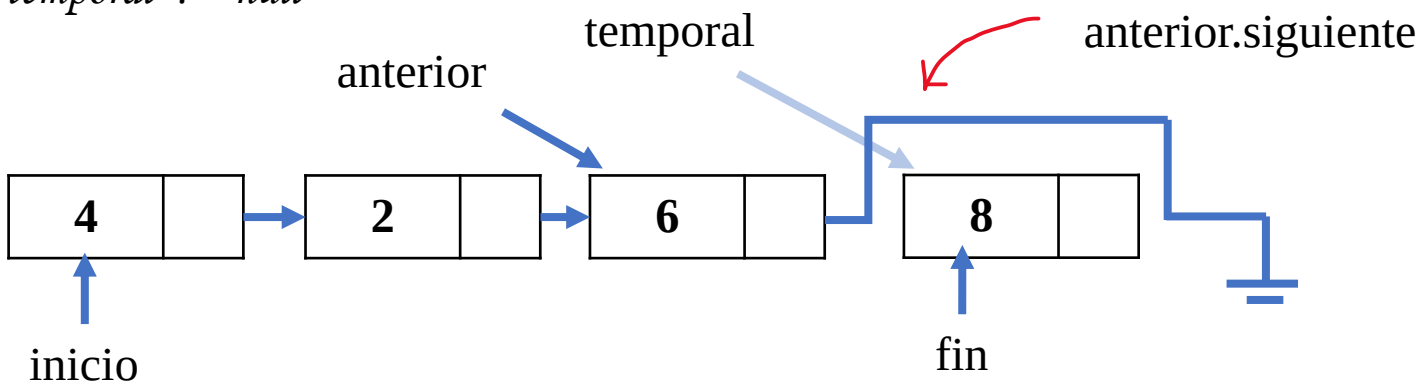
Cuarto caso: elemento = 8



Cuando el elemento a eliminar se encuentra en el ultimo nodo.

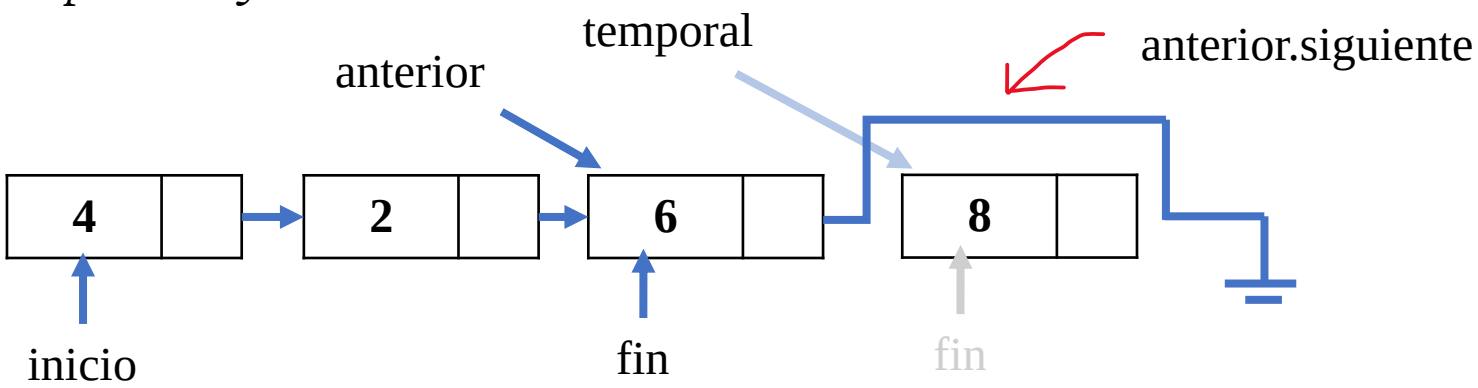
Fin del bucle: *temporal.dato == elemento*

Si: *temporal != null*

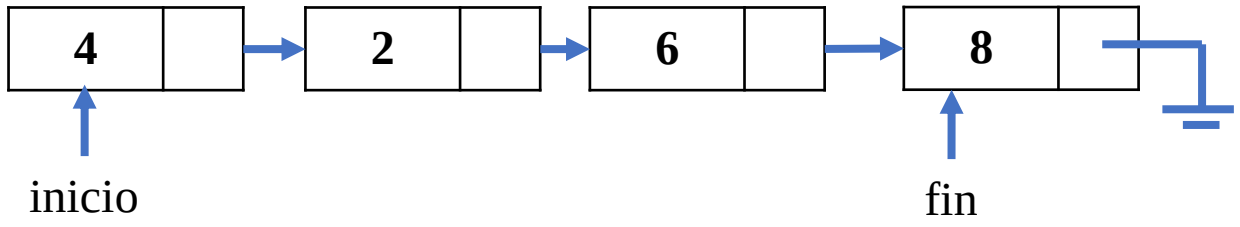


1. Si la Lista No esta vacía entonces
   1. Si inicio es igual a fin Y elemento igual a inicio de dato Entonces
      1. Apuntar inicio y fin a nulo
   2. Si No Si elemento igual a inicio de dato Entonces
      1. Apuntar inicio a inicio de siguiente
   3. Si No
      1. Crear dos Nodos, anterior y temporal
      2. Apuntar anterior a inicio
      3. Apuntar temporal a inicio de siguiente
      4. Mientras temporal sea diferente de nulo y temporal de dato diferente de elemento Hacer
         1. Apuntar anterior a anterior de siguiente
         2. Apuntar temporal a temporal de siguiente
      5. Si temporal es diferente de nulo Entonces
         1. Apuntar anterior de siguiente a temporal de siguiente
         2. Si temporal es igual a fin Entonces
            1. Apuntar fin a anterior

Si: *temporal == fin*



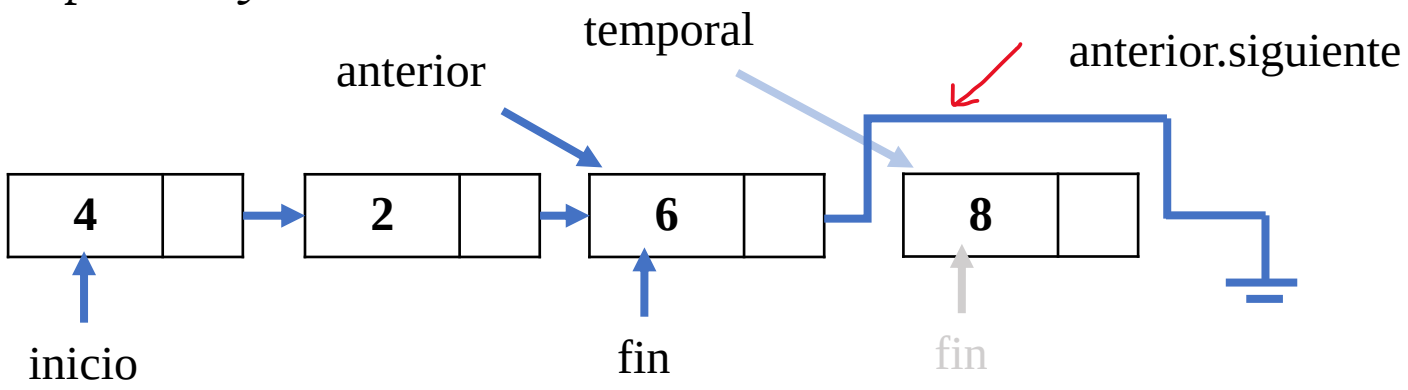
Cuarto caso: elemento = 8



Cuando el elemento a eliminar se encuentra en el ultimo nodo.

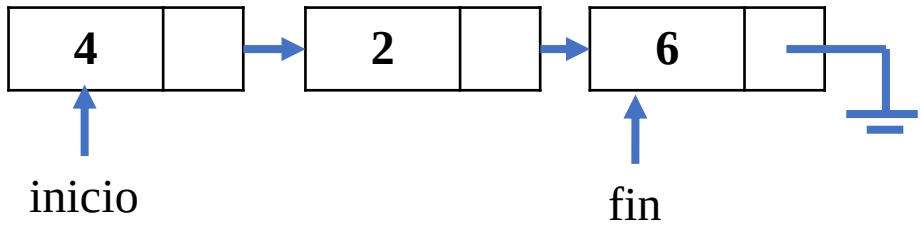
Fin del bucle: *temporal.dato == elemento*

Si: *temporal == fin*

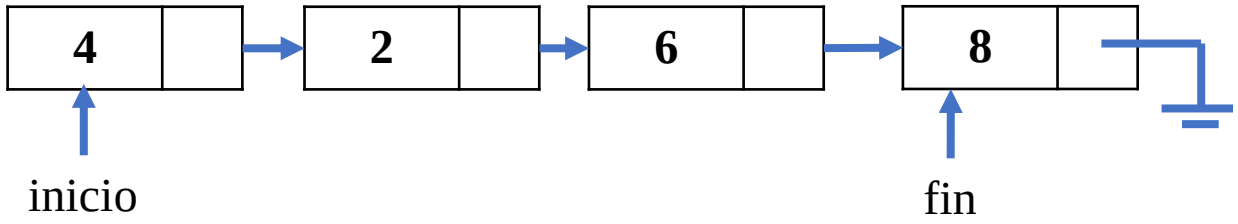


1. Si la Lista No esta vacía entonces
   1. Si inicio es igual a fin Y elemento igual a inicio de dato Entonces
      1. Apuntar inicio y fin a nulo
   2. Si No Si elemento igual a inicio de dato Entonces
      1. Apuntar inicio a inicio de siguiente
   3. Si No
      1. Crear dos Nodos, anterior y temporal
      2. Apuntar anterior a inicio
      3. Apuntar temporal a inicio de siguiente
      4. Mientras temporal sea diferente de nulo y temporal de dato diferente de elemento Hacer
         1. Apuntar anterior a anterior de siguiente
         2. Apuntar temporal a temporal de siguiente
      5. Si temporal es diferente de nulo Entonces
         1. Apuntar anterior de siguiente a temporal de siguiente
         2. Si temporal es igual a fin Entonces
            1. Apuntar fin a anterior

La lista queda de la siguiente manera:



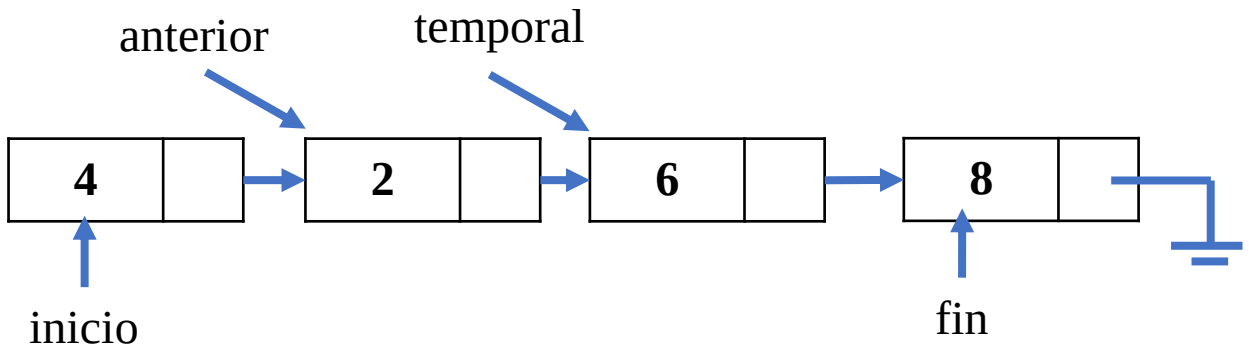
Quinto caso: elemento = 9



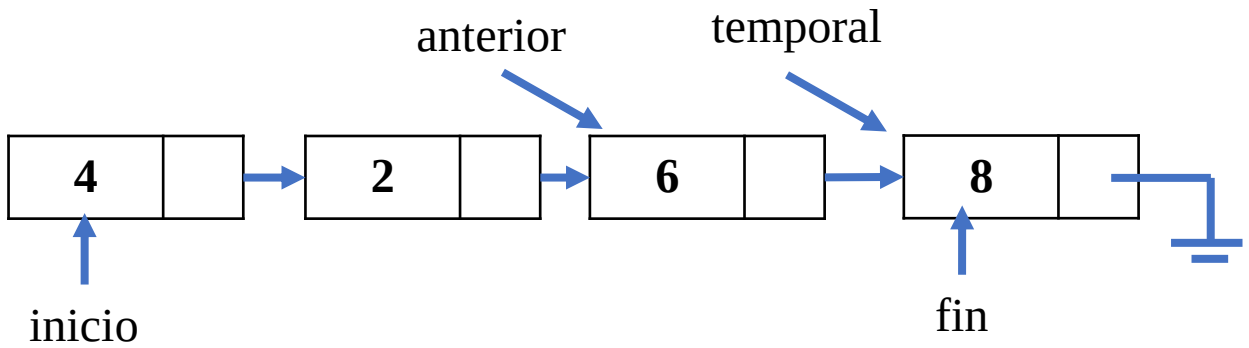
Cuando el elemento a eliminar no se encuentra en la lista.

Bucle: *temporal != null y temporal.dato != elemento*

1°

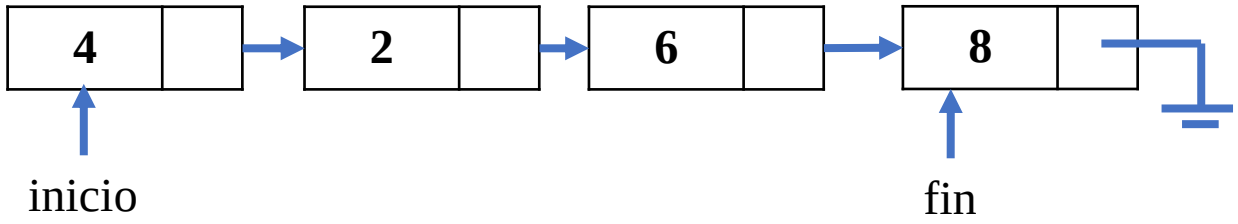


2°



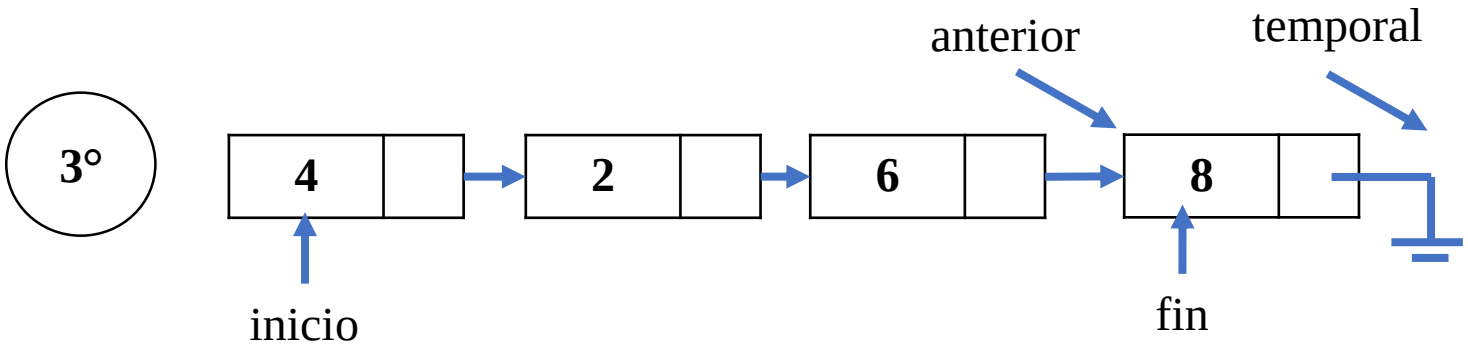
1. Si la Lista No esta vacía entonces
    1. Si inicio es igual a fin Y elemento igual a inicio de dato Entonces
        1. Apuntar inicio y fin a nulo
    2. Si No Si elemento igual a inicio de dato Entonces
        1. Apuntar inicio a inicio de siguiente
    3. Si No
        1. Crear dos Nodos, anterior y temporal
        2. Apuntar anterior a inicio
        3. Apuntar temporal a inicio de siguiente
        4. Mientras temporal sea diferente de nulo y temporal de dato diferente de elemento Hacer
            1. Apuntar anterior a anterior de siguiente
            2. Apuntar temporal a temporal de siguiente
        5. Si temporal es diferente de nulo Entonces
            1. Apuntar anterior de siguiente a temporal de siguiente
            2. Si temporal es igual a fin Entonces
                1. Apuntar fin a anterior

Quinto caso: elemento = 9



Cuando el elemento a eliminar no se encuentra en la lista.

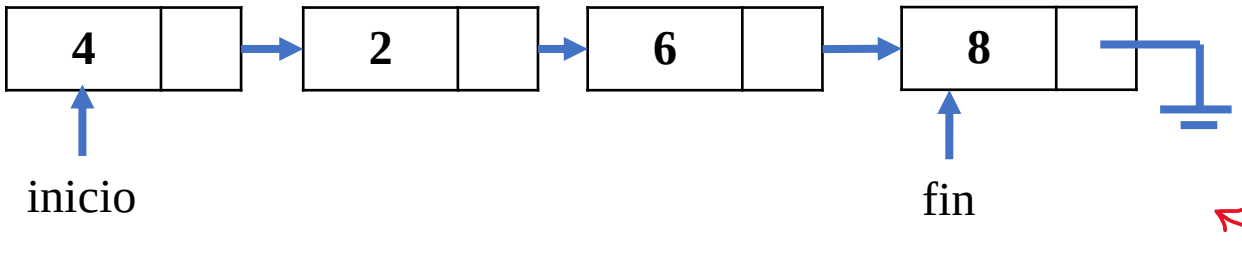
Bucle: *temporal != null y temporal.dato != elemento*



1. Si la Lista No esta vacía entonces
   1. Si inicio es igual a fin Y elemento igual a inicio de dato Entonces
      1. Apuntar inicio y fin a nulo
   2. Si No Si elemento igual a inicio de dato Entonces
      1. Apuntar inicio a inicio de siguiente
   3. Si No
      1. Crear dos Nodos, anterior y temporal
      2. Apuntar anterior a inicio
      3. Apuntar temporal a inicio de siguiente
      4. Mientras temporal sea diferente de nulo y temporal de dato diferente de elemento Hacer
         1. Apuntar anterior a anterior de siguiente
         2. Apuntar temporal a temporal de siguiente
      5. Si temporal es diferente de nulo Entonces
         1. Apuntar anterior de siguiente a temporal de siguiente
         2. Si temporal es igual a fin Entonces
            1. Apuntar fin a anterior

Fin del bucle: *temporal = null*

Sale del condicional



La lista queda así:

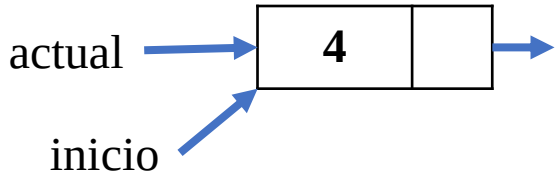
```java
@Override
public void insertarEn(int indice, String elemento) {
    Nodo nuevoNodo = new Nodo( d: elemento);
    Nodo anterior, actual;
    anterior = null;
    actual = inicio;

    if (indice == 0){
        nuevoNodo.ap_siguiente = inicio;
        inicio = nuevoNodo;
    }
    int i=1;
    while (actual != null && i<indice){
        anterior = actual;
        actual = actual.ap_siguiente;
        i++;
    }
    if(actual == null){
        anterior.ap_siguiente = nuevoNodo;
    } else{
        nuevoNodo.ap_siguiente = actual;
        anterior.ap_siguiente = nuevoNodo;
    }
}
```

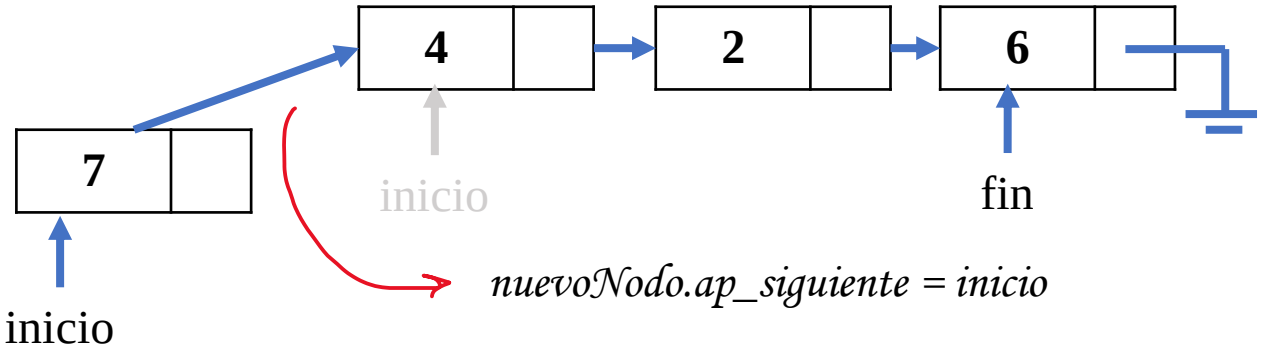
Nuevo nodo:

Nodo anterior:

Nodo actual:



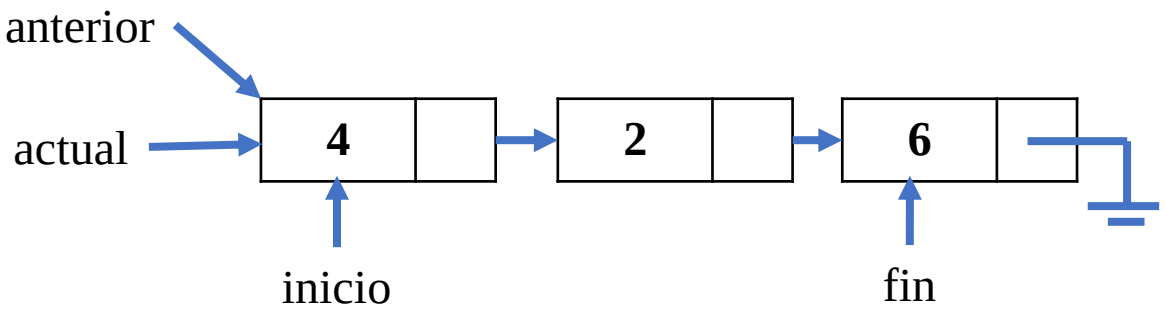
Primer caso: indice = 0

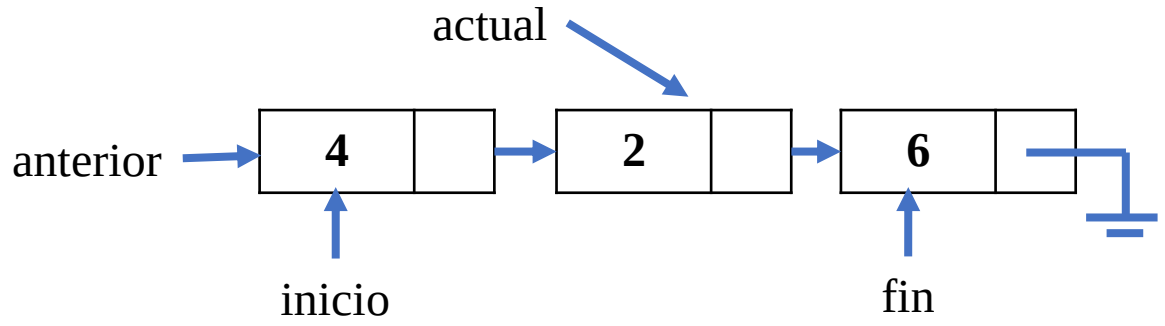


Segundo caso: indice = 2

Bucle: *actual != null y i < indice*

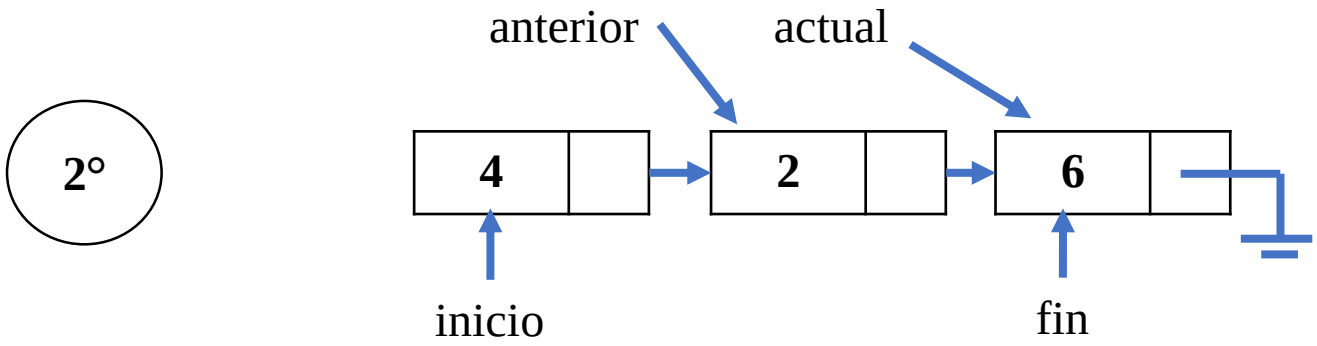
1°





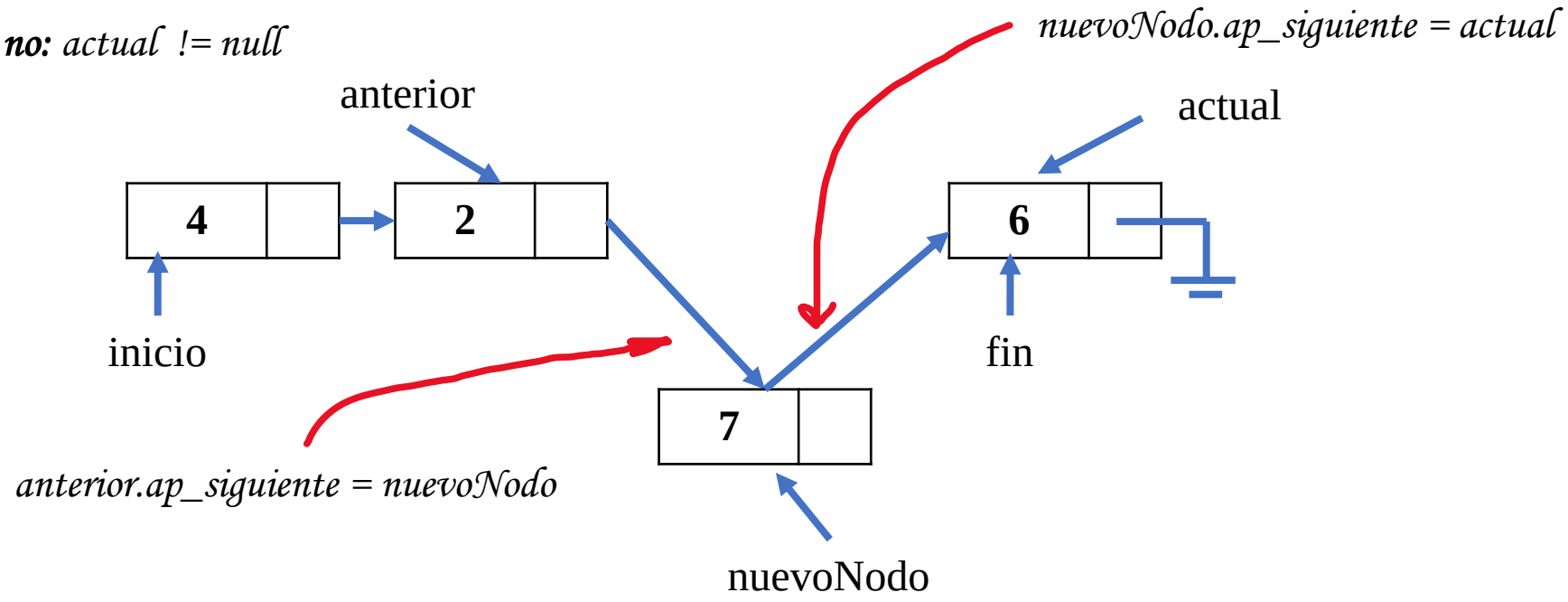
Segundo caso: indice = 2

Bucle: *actual != null y i < indice*



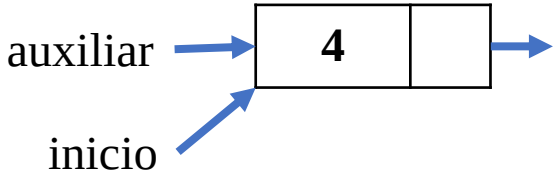
Fin del bucle: i = indice

Si no: *actual != null*



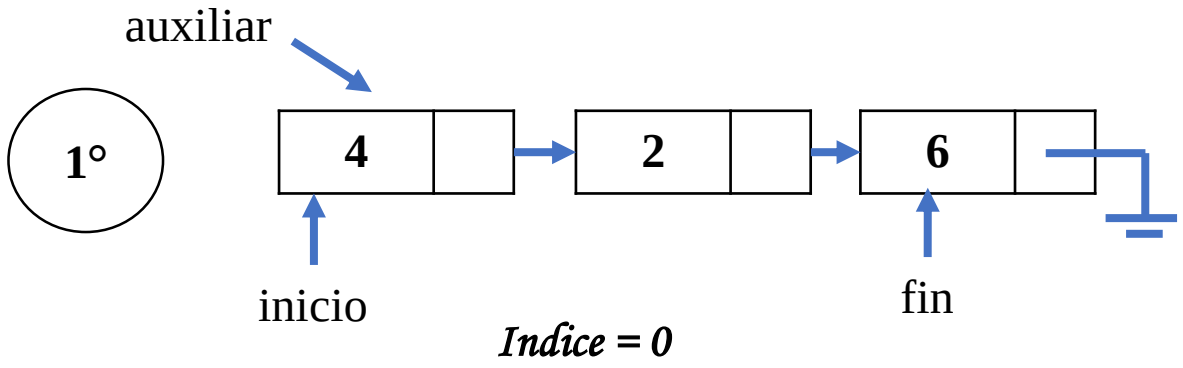
```java
@Override
public int Tamanio() {
    int indice = 0;
    Nodo auxiliar = inicio;
    while (auxiliar != null){
        auxiliar = auxiliar.ap_siguiente;
        indice++;
    }
    return indice;
}
```

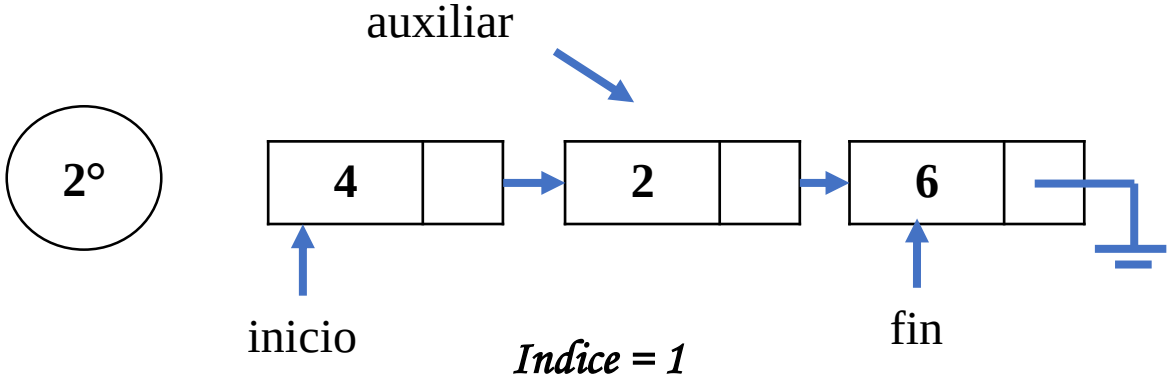
Nodo auxiliar:

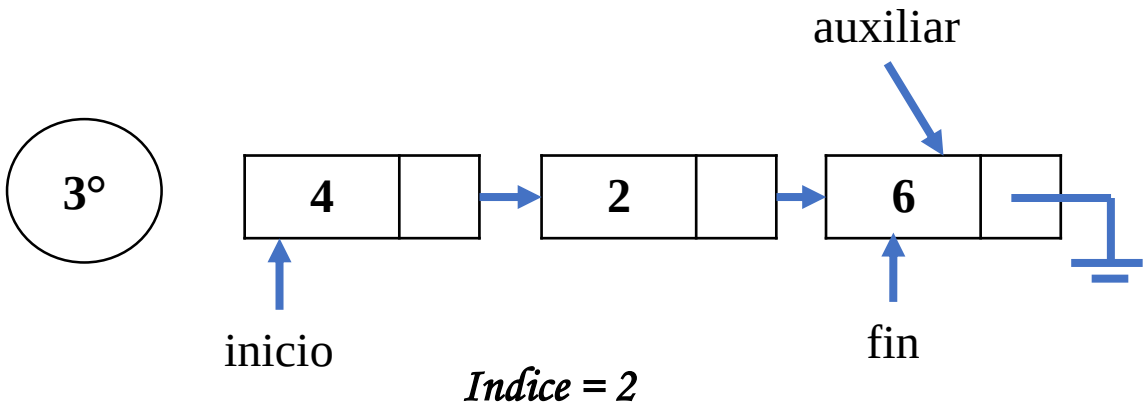


Procedimiento

Bucle: *auxiliar != null*





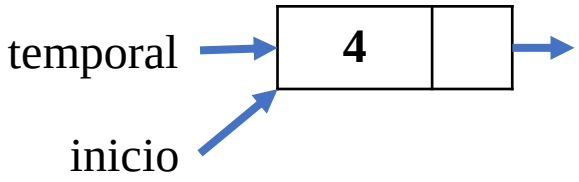


Fin del bucle: *auxiliar == null*

Retorna: *índice = 2*

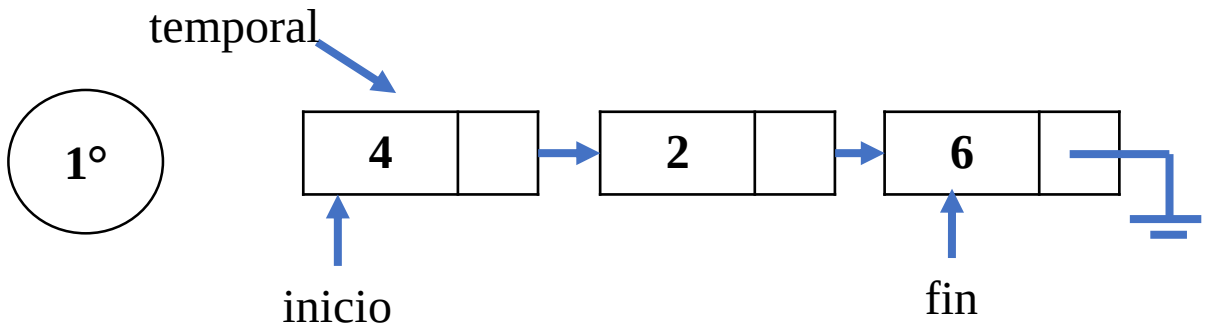
```java
@Override
public boolean contiene(String elemento) {
    Nodo temporal = inicio;
    while(temporal≠null && !temporal.dato.equals( anObject: elemento)){
        temporal=temporal.ap_siguiente;
    }
    return temporal≠null;
}
```

Nodo temporal:



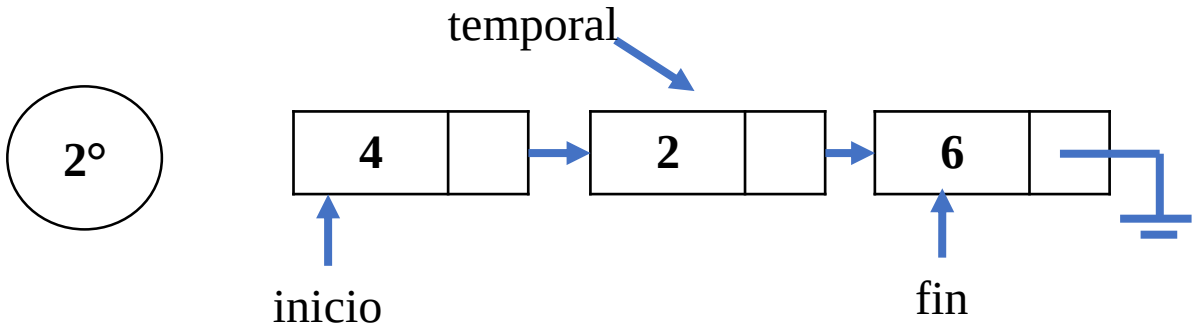
Procedimiento: elemento = 6

Bucle: *temporal != null y temporal.dato != elemento*



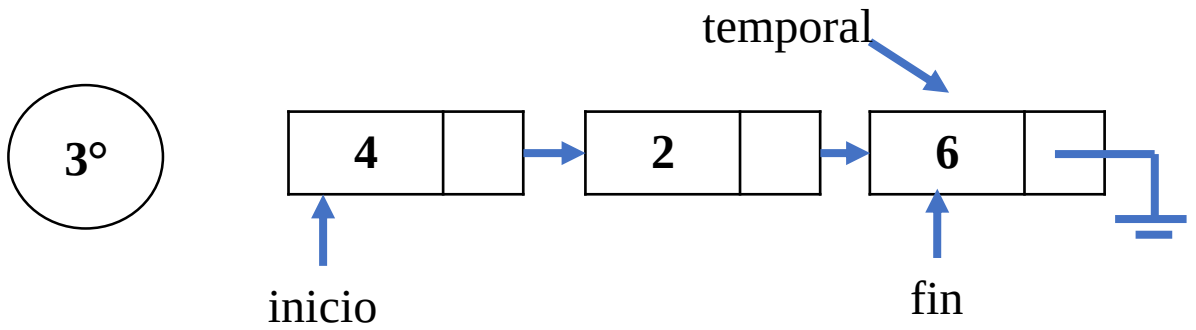
Temporal.dato != elemento

Bucle: *temporal != null y temporal.dato != elemento*



Temporal.dato != elemento

Bucle: *temporal != null y temporal.dato != elemento*



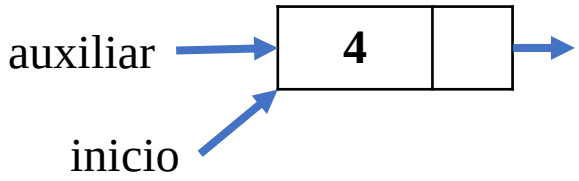
Temporal.dato == elemento

Fin del bucle: *temporal == elemento*

Retorno: *Verdadero*

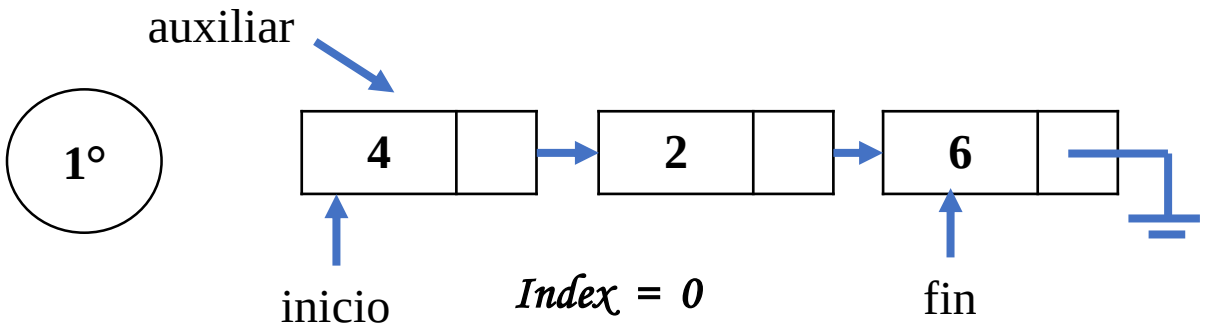
```java
@Override
public int obtenerIndice(String elemento) {
    int index = 0;
    Nodo auxiliar = inicio;
    while(auxiliar != null){
        if(auxiliar.dato.equals( anObject: elemento)){
            return index;
        }
        auxiliar = auxiliar.ap_siguiente;
        index++;
    }
    return -1;
}
```

Nodo auxiliar:

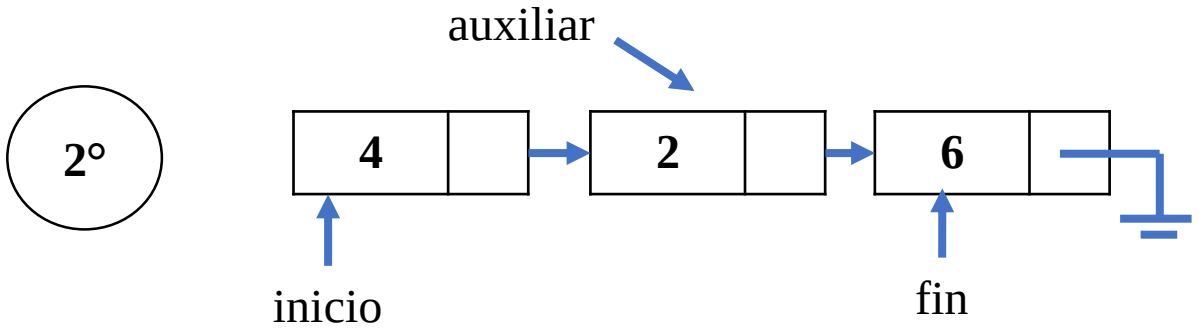


Primer caso: elemento = 6

Bucle: *auxiliar != null*

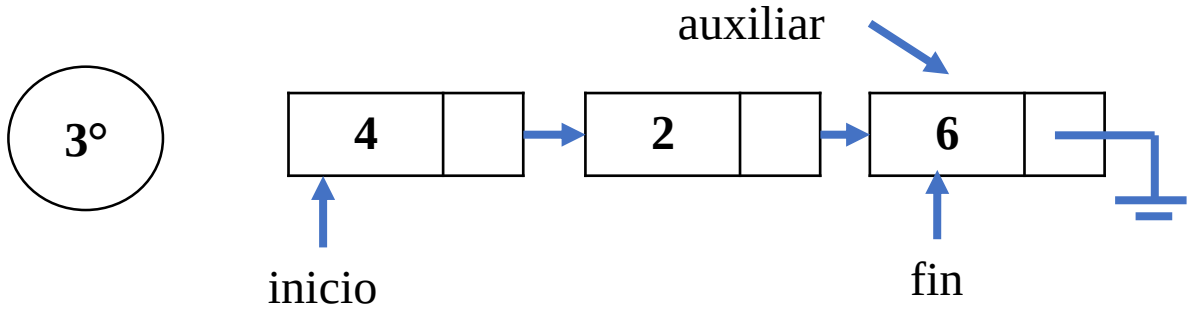


Bucle: *auxiliar != null*



Index = 1 ***auxiliar.dato != elemento***

Bucle: *auxiliar != null*



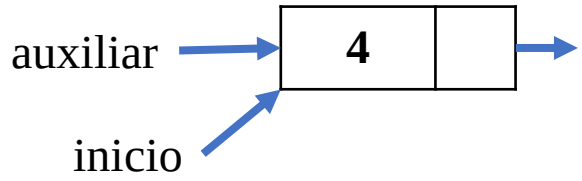
Index = 2 ***auxiliar.dato == elemento***

Fin del bucle: *auxiliar == elemento*

Retorno: index = 2

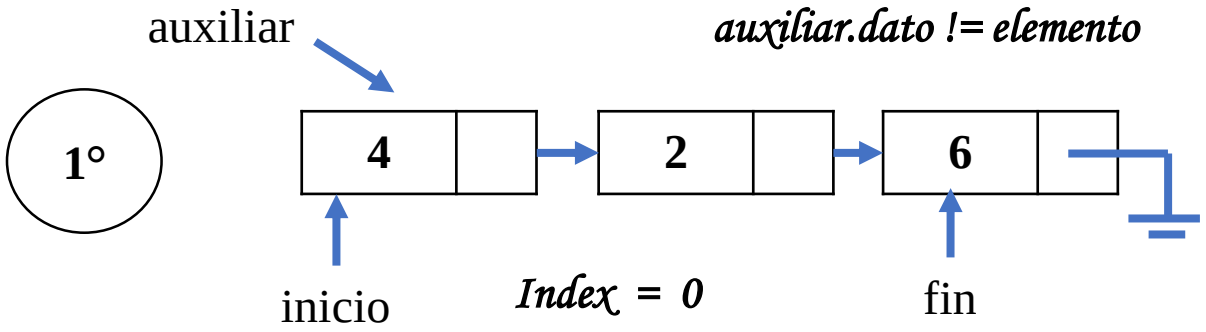
```java
@Override
public int obtenerIndice(String elemento) {
    int index = 0;
    Nodo auxiliar = inicio;
    while(auxiliar != null){
        if(auxiliar.dato.equals( anObject: elemento)){
            return index;
        }
        auxiliar = auxiliar.ap_siguiente;
        index++;
    }
    return -1;
}
```

Nodo auxiliar:



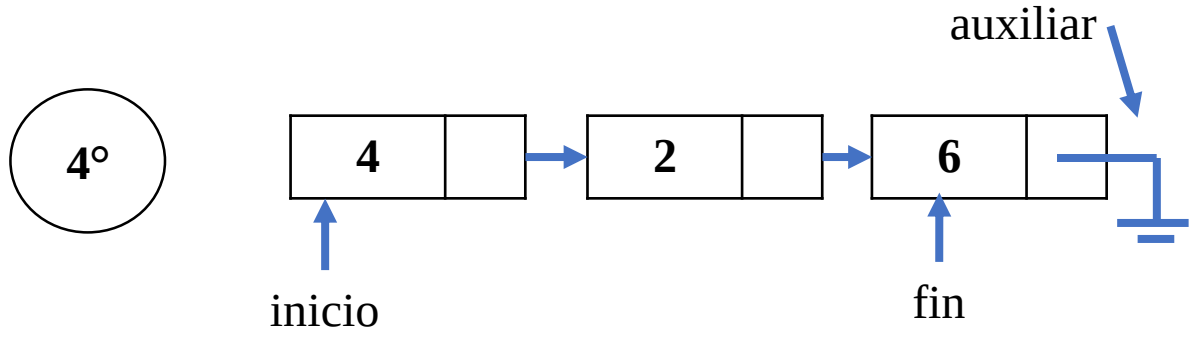
Segundo caso: elemento = 9

Bucle: *auxiliar != null*



...

Bucle: *auxiliar != null*



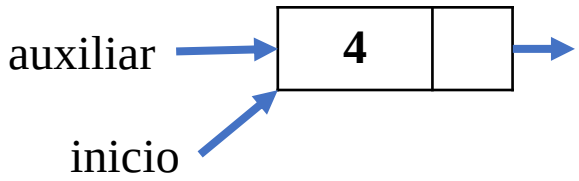
Index = 3 ***auxiliar == null***

Fin del bucle: *auxiliar == null*

Retorno: -1

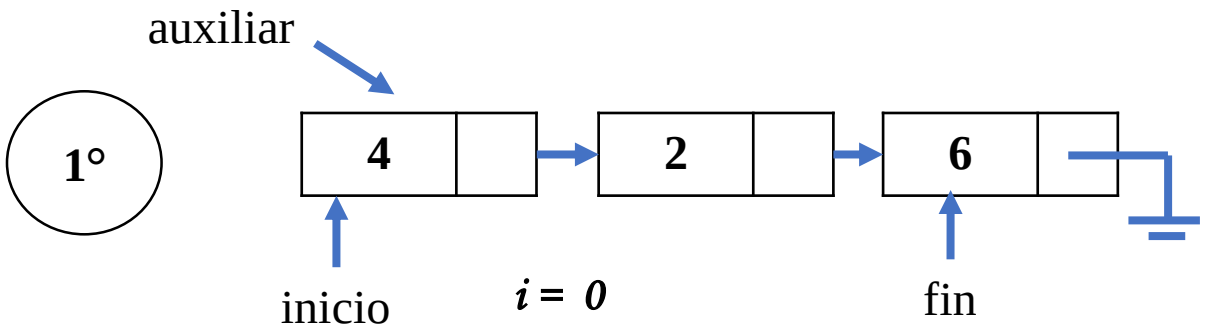
```java
@Override
public String obtenerElemento(int indice) {
    int i = 0;
    Nodo auxiliar = inicio;
    while(auxiliar != null){
        if(i == indice){
            return auxiliar.dato;
        }
        i++;
    }
    return null;
}
```

Nodo auxiliar:

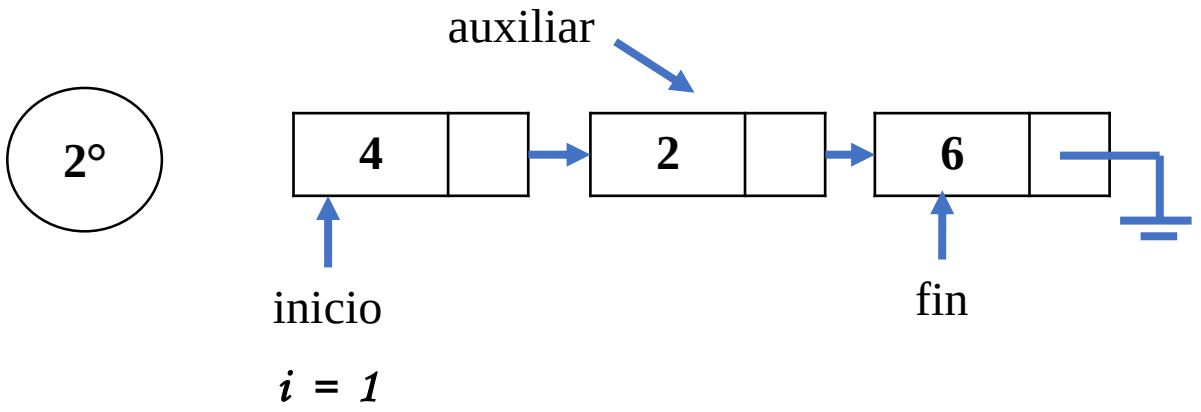


Primer caso: índice = 1

Bucle: *auxiliar != null*



Bucle: *auxiliar != null*



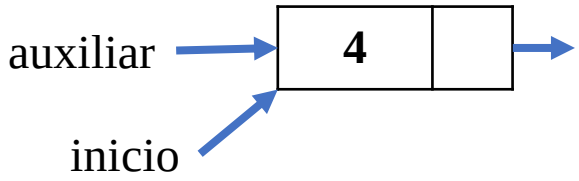
i = 1

Fin del bucle: *el índice coincide con el iterador (i)*

Retorno: *auxiliar.dato = 2*

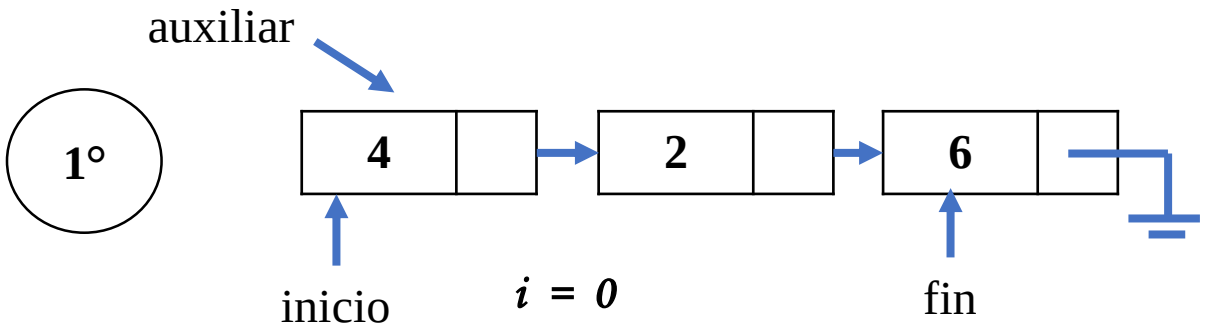
```java
@Override
public String obtenerElemento(int indice) {
    int i = 0;
    Nodo auxiliar = inicio;
    while(auxiliar != null){
        if(i == indice){
            return auxiliar.dato;
        }
        i++;
    }
    return null;
}
```

Nodo auxiliar:



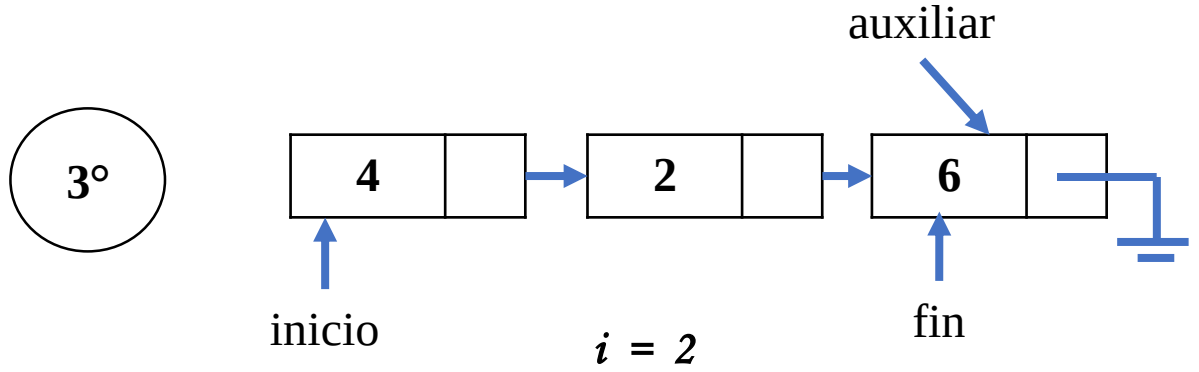
Segundo caso: índice = 3

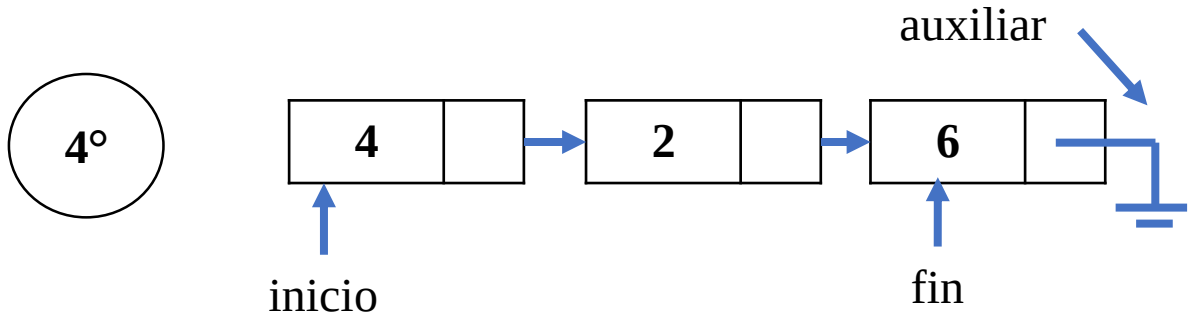
Bucle: *auxiliar != null*



...

Bucle: *auxiliar != null*





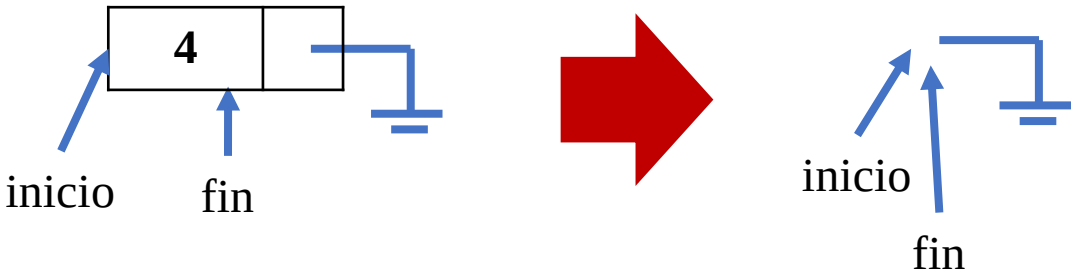
i = 3 ***auxiliar == null***

Fin del bucle: *auxiliar == null*

Retorno: *null*

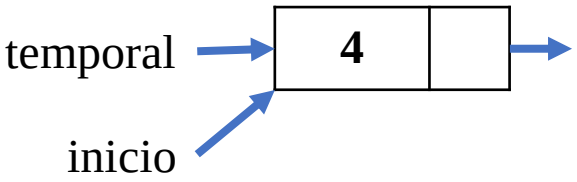
```java
@Override
public void eliminarElUltimo() {
    if (inicio == fin) {
        inicio = fin = null;
    } else {
        Nodo temporal = inicio;
        while (temporal.ap_siguiente != fin) {
            temporal = temporal.ap_siguiente;
        }
        fin = temporal;
        fin.ap_siguiente = null;
    }
}
```

Primer caso: inicio == final

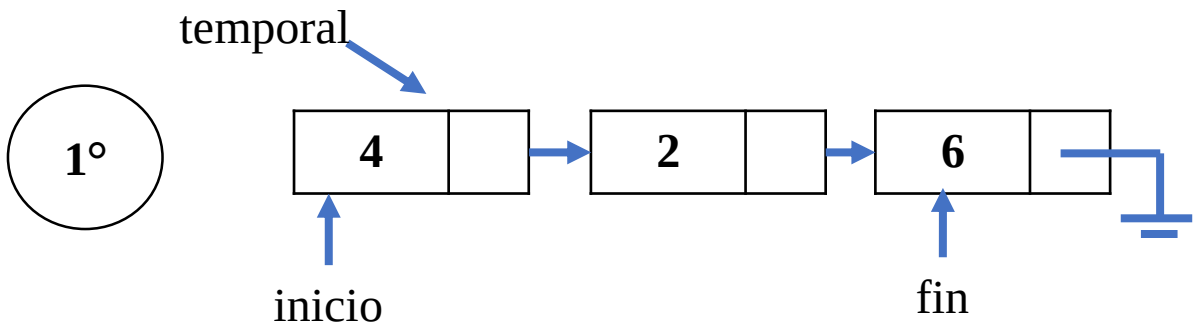


Segundo caso: inicio != final

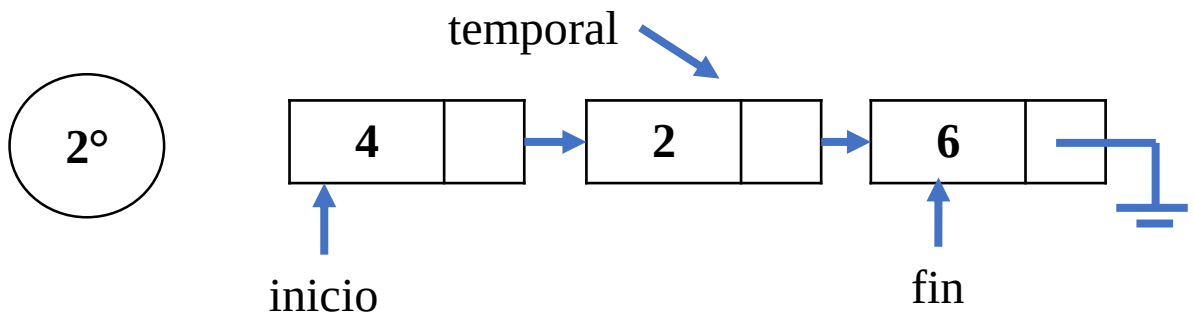
Nodo temporal:



Bucle: *temporal.ap_siguiente != fin*



temporal.ap_siguiente != fin



temporal.ap_siguiente == fin

Fin del bucle: *temporal.ap_siguiente == fin*