

# Introducción a la Algorítmica y Programación (3300)

Prof. Ariel Ferreira Szpiniak - aferreira@exa.unrc.edu.ar

Departamento de Computación  
Facultad de Cs. Exactas, Fco-Qcas y Naturales  
Universidad Nacional de Río Cuarto

## Teoría 15

### Estructuras de Datos

#### Estructuras dinámicas de datos

#### Listas encadenadas: simples y dobles

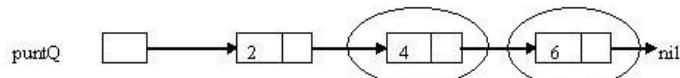


2018 Prof. Ariel Ferreira Szpiniak

1

## Listas simplemente encadenadas

Excepto para el caso de particular de inserción antes del primer elemento y de supresión del primer elemento de la lista, el resto de las inserciones y supresiones son similares, independientemente del lugar donde se realicen las mismas, debido a que el elemento se encuentra entre dos dobletes o es el último doblete, en cuyo caso como veremos se trata de la misma forma que entre dobletes.



A los efectos de realizar un análisis exhaustivo de las operaciones de inserción y supresión de elementos sobre listas simplemente encadenadas, dividiremos el análisis en dos partes: por un lado la **inserción** y por otro la **supresión**. A su vez cada una de ellas será analizada en dos situaciones distintas que llamaremos: **a la cabeza** cuando la operación se realice en el primer lugar de la lista, y **normal** cuando la operación se realice en cualquier otro lugar de la lista.



2018 Prof. Ariel Ferreira Szpiniak

3

## Listas simplemente encadenadas

- Las listas simplemente encadenadas se refieren al hecho de que los elementos a encadenar son dobletes, es decir, tienen dos campos, uno para la información y otro para la dirección del próximo elemento.
- A continuación analizaremos de que manera podemos realizar las inserciones y supresiones de elementos para manipular dicha lista.
- Veremos también que pueden darse diferentes situaciones a la hora de insertar o suprimir un elemento dependiendo del lugar donde se encuentra ubicado.
- De allí surge que existe una situación particular tanto para la inserción como para la supresión de elementos. Esta situación tiene que ver con la inserción o supresión de un elemento al comienzo de la lista. El comienzo de una lista recibe el nombre de **cabeza**.
- Decimos que se trata de una situación especial porque en el caso de la inserción de un nuevo elemento, éste debe insertarse entre una variable de tipo puntero, que apunta al primer elemento, y un doblete, el actual elemento. Lo mismo sucede con la supresión.



2018 Prof. Ariel Ferreira Szpiniak

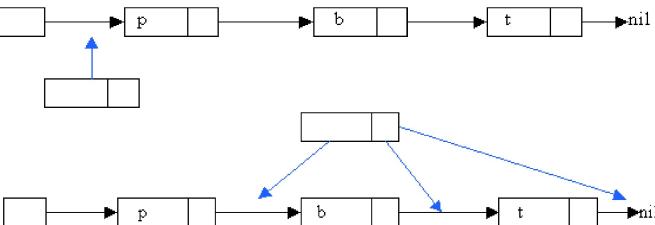
2

## Listas simplemente encadenadas

### Inserción:

- A la cabeza
- 

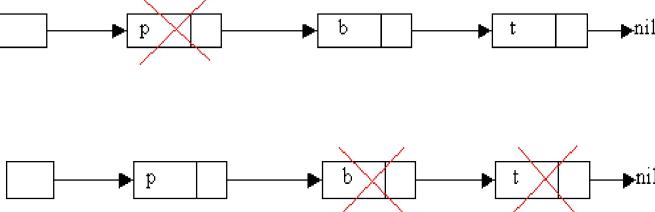
- Normal



### Supresión:

- A la cabeza
- 

- Normal



2018 Prof. Ariel Ferreira Szpiniak

4

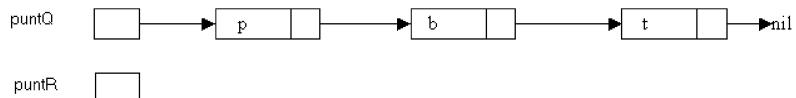
# Listas simplemente encadenadas

## Inserción de un elemento

Para analizar esta operación supondremos tener definida una estructura dinámica del tipo **TelemCar** y dos variables, **puntQ**, **puntR** de tipo **puntero a TelemCar**.

TelemCar =*<info ε Caracter, next ε puntero a TelemCar>*  
**puntQ, puntR ε puntero a TelemCar**

Supondremos además, como precondition, que ya existen 3 dobletes con los valores p, b, t donde la variable de tipo puntero **puntQ** apunta al primero de ellos.



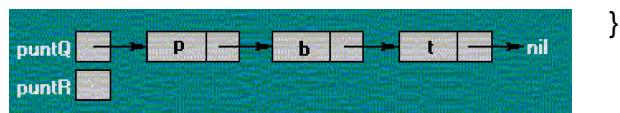
La variable de tipo puntero **puntR** será utilizada para la creación del nuevo doblete que se desea insertar.



# Listas simplemente encadenadas

## Inserción de un elemento a la cabeza

{Precondición:



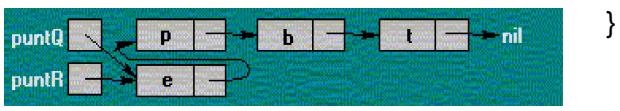
Obtener(puntR)

(^puntR).info <- 'e'

(^puntR).next <- puntQ

puntQ <- puntR

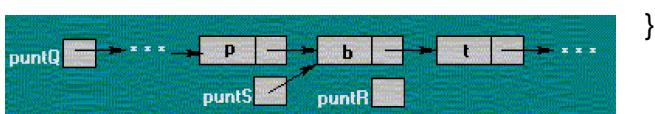
{Poscondición:



# Listas simplemente encadenadas

## Inserción normal de un elemento

{Precondición:



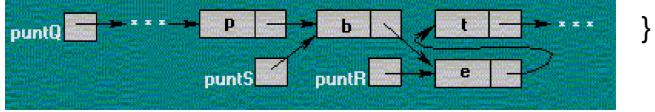
Obtener(puntR)

(^puntR).next <- (^puntS).next

(^puntR).info <- 'e'

(^puntS).next <- puntR

{Poscondición:



¿Qué sucede si el elemento debe insertarse al final de la lista?



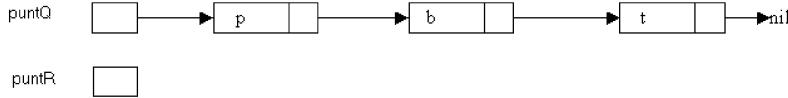
# Listas simplemente encadenadas

## Supresión de un elemento

Para analizar esta operación también supondremos tener definida una estructura dinámica del tipo **TelemCar** y dos variables, **puntQ, puntR** de tipo **puntero a TelemCar**.

TelemCar =*<info ε Caracter, next ε puntero a TelemCar>*  
**puntQ, puntR ε puntero a TelemCar**

Supondremos además, como precondition, que ya existen 3 dobletes con los valores p, b, t donde la variable de tipo puntero **puntQ** apunta al primero de ellos.



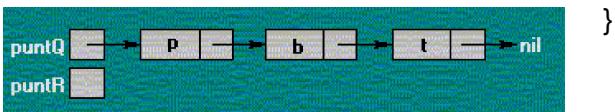
La variable de tipo puntero **puntR** será utilizada para la creación del nuevo doblete que se desea insertar.



# Listas simplemente encadenadas

## Supresión de un elemento a la cabeza

{Precondición:



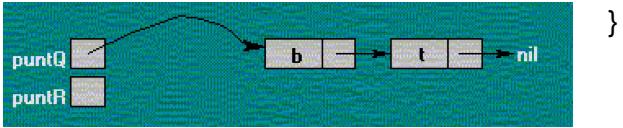
}

puntR <- puntQ

puntQ <- (^puntQ).next

Liberar(puntR)

{Poscondición:



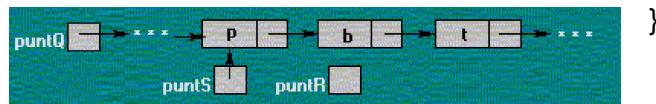
}



# Listas simplemente encadenadas

## Supresión normal de un elemento

{Precondición:



}

puntR <- (^puntS).next

(^puntS).next <- (^puntR).next

Liberar(puntR)

{Poscondición:

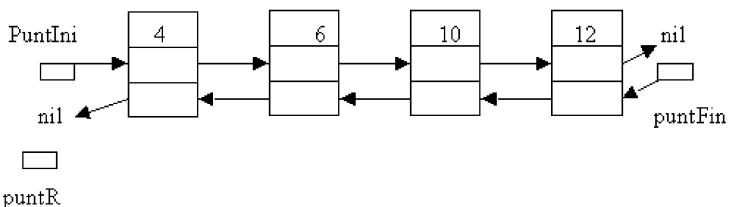


}

¿Qué sucede si el elemento debe eliminarse al final de la lista?



# Listas doblemente encadenadas



puntR

Al igual que con las listas simplemente encadenadas, dividiremos el análisis en dos partes iguales: por un lado la **inserción** y por el otro la **supresión**.

A su vez cada una de ellas será analizada en dos situaciones distintas que llamaremos: **a la cabeza**, cuando la operación se realice en el primer lugar de la lista y **normal**, cuando la operación se realice en cualquier otro lugar de la lista.

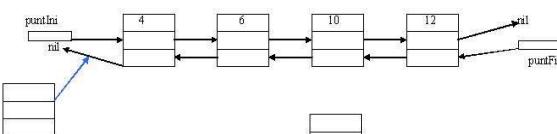
La inserción y supresión al final no será tratada puesto que es similar a la realizada a la cabeza, con la única modificación del nombre del puntero que apunta al elemento.



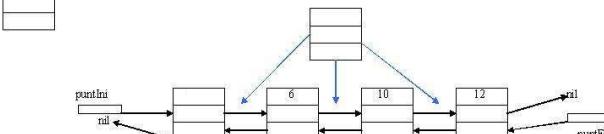
# Listas doblemente encadenadas

## Inserción:

- A la cabeza

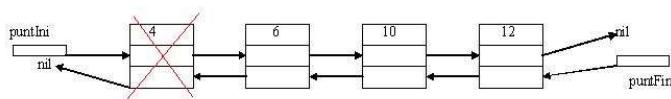


- Normal

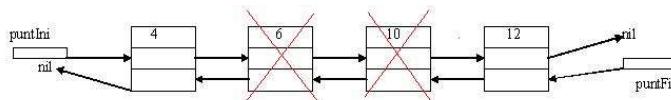


## Supresión:

- A la cabeza



- Normal



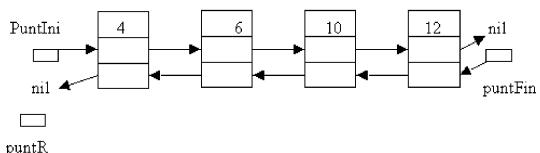
# Listas doblemente encadenadas

## Inserción de un elemento

Para analizar esta operación supondremos tener definida una estructura dinámica del tipo **TelemEnt** y tres variables, **puntIni**, **puntFin**, y **puntR** de tipo **puntero a TelemEnt**.

**TelemEnt** = <info ε Caracter, back ε puntero a TelemEnt, next ε puntero a TelemEnt>  
**puntIni**, **puntFin**, **puntR** ε puntero a TelemEnt

Supondremos además, como precondition, que ya existen 4 tripletes con los valores 4, 6, 10 y 12, donde la variable de tipo numérico **puntIni** apunta al primero de ellos y **puntFin** apunta al último de ellos:



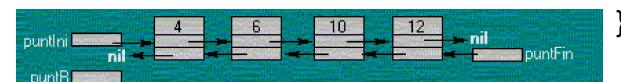
La variable de tipo puntero **puntR** será utilizada para la creación del nuevo triplete que se desea insertar.



# Listas doblemente encadenadas

## Inserción de un elemento a la cabeza

{Precondición:



Obtener(**puntR**)

(^**puntR**).info <- 2

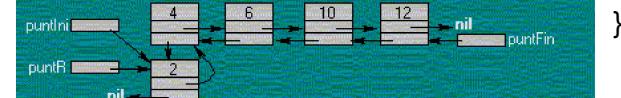
(^**puntR**).back <- **puntIni**

(^**puntR**).next <- nil

(^**puntIni**).back <- **puntR**

**puntIni** <- **puntR**

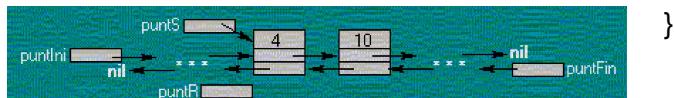
{Poscondición:



# Listas doblemente encadenadas

## Inserción normal de un elemento

{Precondición:



Obtener(**puntR**)

(^**puntR**).info <- 2

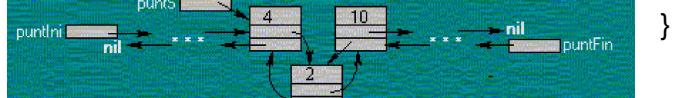
(^**puntR**).back <- **puntS**

(^**puntR**).next <- (^**puntS**).next

(^(^**puntS**).next).back <- **puntR**

(^**puntS**).next <- **puntR**

{Poscondición:



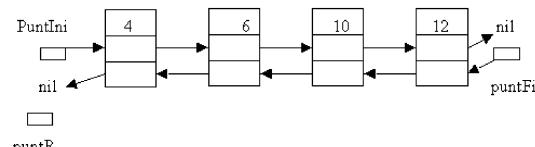
# Listas doblemente encadenadas

## Supresión de un elemento

Para analizar esta operación supondremos tener definida una estructura dinámica del tipo **TelemEnt** y tres variables, **puntIni**, **puntFin**, y **puntR** de tipo **puntero a TelemEnt**.

**TelemEnt** = <info ε Caracter, back ε puntero a TelemEnt, next ε puntero a TelemEnt>  
**puntIni**, **puntFin**, **puntR** ε puntero a TelemEnt

Supondremos además, como precondition, que ya existen 4 tripletes con los valores 4, 6, 10 y 12, donde la variable de tipo numérico **puntIni** apunta al primero de ellos y **puntFin** apunta al último de ellos:



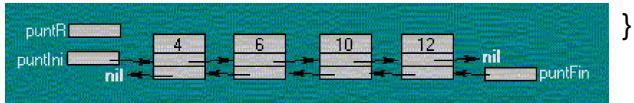
La variable de tipo puntero **puntR** será utilizada para la creación del nuevo triplete que se desea insertar.



# Listas doblemente encadenadas

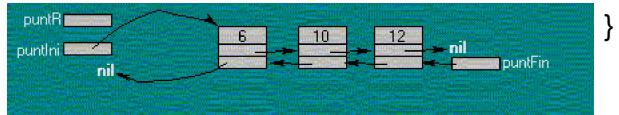
## Supresión de un elemento a la cabeza

{Precondición:



```
puntR <- PuntInI  
(^(^puntInI).next).back <- nil  
puntInI <- (^puntInI).next  
Liberar(puntR)
```

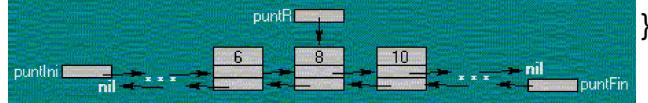
{Poscondición:



# Listas doblemente encadenadas

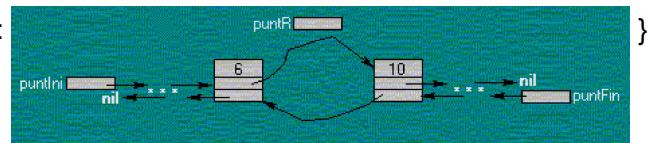
## Supresión normal de un elemento

{Precondición:



```
(^(^puntR).back).next <- (^puntR).next  
(^(^puntR).next).back <- (^puntR).back  
Liberar(puntR)
```

{Poscondición:



# Listas simplemente encadenadas

## Elemento ficticio

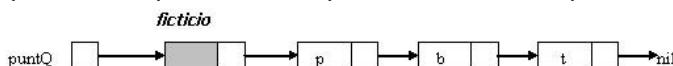
- Para evitar el caso especial de inserciones y supresiones a la cabeza, y a la cola en el caso de estructuras doblemente encadenadas, se presentará a continuación la idea de implementar la lista con la ayuda de un elemento ficticio.
- El **elemento ficticio** es un elemento que **no contiene datos útiles** pero que se usa en la implementación de la misma a los efectos de **unificar las sentencias de inserción y supresión**.
- Analizaremos únicamente el uso de ficticio para listas simplemente encadenadas, dado que para listas doblemente encadenadas es similar.
- Para el caso de listas simplemente encadenadas, utilizaremos **ficticio en la cabeza** de la lista. De esta manera, el elemento ficticio es siempre el primero de lista, con lo cual tanto la **inserción como la supresión de elemento se realiza entre dos dobletes**, es decir que estaríamos en todos los casos frente a lo que previamente denominaremos **inserción o supresión normal**.



# Listas simplemente encadenadas

## Elemento ficticio

Por ejemplo la lista "pbt" utilizada previamente, se implementaría:

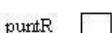
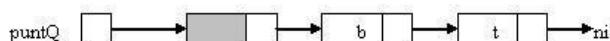


Para analizar el tema de elemento ficticio supondremos tener definida:

TelemCar = <info ε Caracter, next ε puntero a TelemCar>

puntQ, puntR ε puntero a TelemCar

Supondremos además, como precondición, que ya existen 2 dobletes con los valores 'b' y 't', donde la variable de tipo puntero **puntQ** apunta al primero de ellos:



La variable de tipo puntero **puntR** será utilizada para la creación del nuevo doblete que se desea insertar.



# Listas simplemente encadenadas

## Elemento ficticio. Inserción de un elemento

El primer lugar analizaremos dos situaciones y compararemos el conjunto de sentencias obtenidas para realizar dichas inserciones:

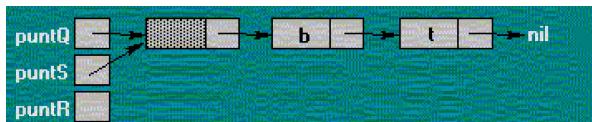
- Una sobre como insertar un nuevo doblete con el valor 'e' al comienzo de la lista, lo que implica, en éste caso, que sea a la cabeza.
- Otra sobre como insertar un nuevo doblete con el valor 'e' entre los dobletes que alojan a los valores 'b' y 't'. La situación sería similar si el nuevo doblete se insertaría en cualquier lugar entre el segundo y el último. Es oportuno destacar que las sentencias utilizadas serían las mismas para insertar un doblete al final de la lista.



# Listas simplemente encadenadas

## Elemento ficticio. Inserción de un elemento al comienzo

{Precondición:



}

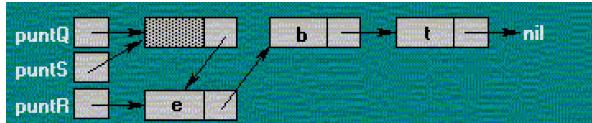
Obtener(puntR)

(^puntR).info <- 'e'

(^puntR).next <- (^puntS).next

(^puntS).next <- puntR

{Poscondición:

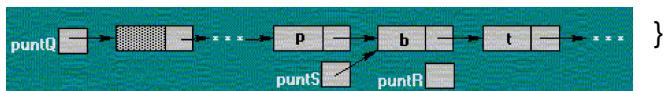


}

# Listas simplemente encadenadas

## Elemento ficticio. Inserción de un elemento en cualquier otro lugar

{Precondición:



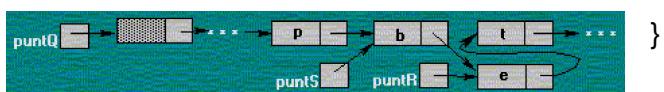
Obtener(puntR)

(^puntR).info <- 'e'

(^puntR).next <- (^puntS).next

(^puntS).next <- puntR

{Poscondición:



# Listas simplemente encadenadas

## Elemento ficticio. Inserción de un elemento al comienzo

Obtener(puntR)

(^puntR).info <- 'e'

(^puntR).next <- (^puntS).next

(^puntS).next <- puntR

## Elemento ficticio. Inserción de un elemento en cualquier otro lugar

Obtener(puntR)

(^puntR).info <- 'e'

(^puntR).next <- (^puntS).next

(^puntS).next <- puntR

Los dos tipos de inserciones requieren de las mismas sentencias.

No es necesario tener la precaución de identificar cuando estamos ante la presencia de una inserción al comienzo de la lista y cuando no.



# Listas simplemente encadenadas

## Elemento ficticio. Supresión de un elemento

El primer lugar analizaremos dos situaciones y compararemos el conjunto de sentencias obtenidas para realizar dichas inserciones:

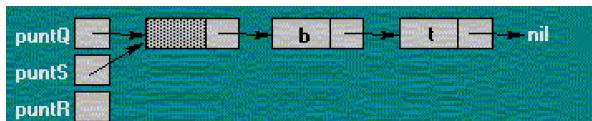
- Una sobre como eliminar el doblete con el valor 'e' al comienzo de la lista, lo que implica, en éste caso, que sea a la cabeza.
- Otra sobre como eliminar el nuevo doblete con el valor 'b'. La situación sería similar si el doblete a eliminar estuviera ubicado en cualquier lugar entre el segundo y el último. Es oportuno destacar que las sentencias utilizadas serían las mismas para eliminar un doblete al final de la lista.



# Listas simplemente encadenadas

## Elemento ficticio. Supresión de un elemento al comienzo

{Precondición:



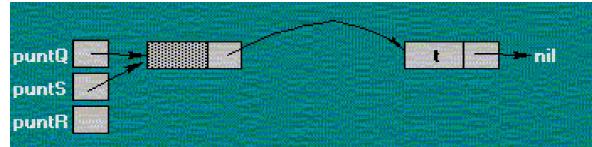
}

$puntR \leftarrow (^{puntS}).next$

$(^{puntS}).next \leftarrow (^{puntR}).next$

Liberar(puntR)

{Poscondición:

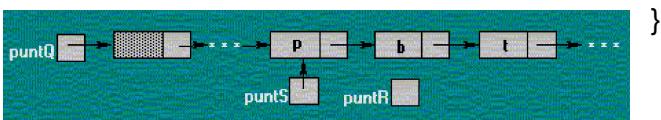


}

# Listas simplemente encadenadas

## Elemento ficticio. Supresión de un elemento en cualquier otro lugar

{Precondición:

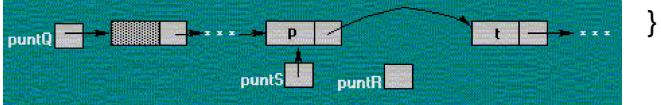


$puntR \leftarrow (^{puntS}).next$

$(^{puntS}).next \leftarrow (^{puntR}).next$

Liberar(puntR)

{Poscondición:



}

# Listas simplemente encadenadas

## Elemento ficticio. Supresión de un elemento al comienzo

$puntR \leftarrow (^{puntS}).next$

$(^{puntS}).next \leftarrow (^{puntR}).next$

Liberar(puntR)

## Elemento ficticio. Supresión de un elemento en cualquier otro lugar

$puntR \leftarrow (^{puntS}).next$

$(^{puntS}).next \leftarrow (^{puntR}).next$

Liberar(puntR)

Los dos tipos de supresiones requieren de las mismas sentencias.

No es necesario tener la precaución de identificar cuando estamos ante la presencia de una supresión al comienzo de la lista y cuando no.



# Bibliografía

- Scholl, P. y Peyrin, J.-P. “Esquemas Algorítmicos Fundamentales: Secuencias e iteración”.
- Biondi, J. y Clavel, G. “Introducción a la Programación. Tomo 1: Algorítmica y Lenguajes”.
- Clavel, G. y Biondi, J. “Introducción a la Programación. Tomo 2: Estructuras de Datos”.



Citar/Atribuir: Ferreira, Szpiniak, A. (2018). Teoría 15: Estructuras de Datos. Estructuras dinámicas de datos. Listas encadenadas Introducción a la Algorítmica y Programación (3300). Departamento de Computación. Facultad de Cs. Exactas, Fco-Qcas y Naturales. Universidad Nacional de Río Cuarto.

**Usted es libre para:**

Compartir: copiar y redistribuir el material en cualquier medio o formato.

Adaptar: remezclar, transformar y crear a partir del material.

El licenciatante no puede revocar estas libertades en tanto usted siga los términos de la licencia.

Bajo los siguientes términos:



**Atribución:** Usted debe darle crédito a esta obra de manera adecuada, proporcionando un enlace a la licencia, e indicando si se han realizado cambios. Puede hacerlo en cualquier forma razonable, pero no de forma tal que sugiera que usted o su uso tienen el apoyo del licenciatante.



**Compartir Igual:** Si usted mezcla, transforma o crea nuevo material a partir de esta obra, usted podrá distribuir su contribución siempre que utilice la misma licencia que la obra original.

<https://creativecommons.org/licenses/by-sa/2.5/ar/>

