Programación II
Implementaciones Dinámicas

2C 2023 TN – Ing. Elizabeth Barrera

Temas

- *Implementaciones dinámicas*
- *Pila*
- *Cola*
- *Cola con prioridad*
- *Conjunto*
- *Diccionario simple*
- *Diccionario múltiple*


Implementaciones
Dinámicas

Implementaciones dinámicas

Las implementaciones dinámicas se basan en el concepto de utilizar ***estructuras dinámicas*** *para implementar los TDA, a diferencia de las implementaciones estáticas que utilizan arreglos, que son estructuras estáticas.*

Aclaración

- *Cabe recordar que esta implementación **no afectará en absoluto el comportamiento** del TDA, es decir que todo aquel que haya usado el TDA con una implementación estática, lo podrá seguir usando de la misma forma, con una implementación dinámica, sin ninguna modificación.*
- *La implementación, como ya se ha visto, debe ser transparente al usuario.*


Pila

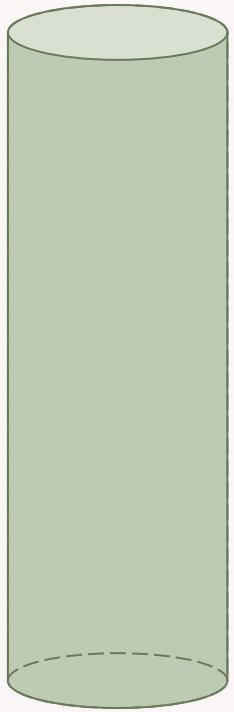
Pila - implementación dinámica

- Se mantendrá una cadena de nodos, con un ***puntero "origen" al tope de la pila***, que apuntará al siguiente, y así sucesivamente.
- Cada nodo corresponde a un elemento apilado.
- Si la lista está vacía, será un puntero null.
- Para agregar un nodo (apilar), se lo crea, y se agrega al comienzo de la cadena (***inserción al principio***).

Pila - implementación dinámica

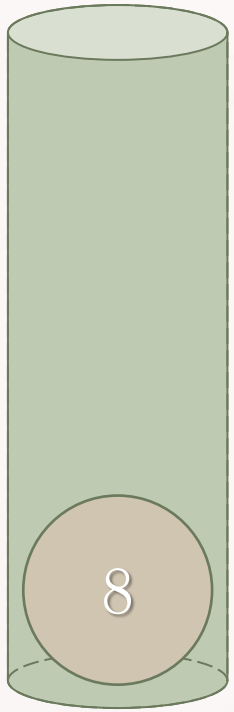
- *Para eliminar un nodo (desapilar), simplemente se elimina el primer nodo, dejando el puntero apuntando al segundo, si es que existe (**eliminar el origen**).*
- *O sea, cada operación apilar o desapilar cambiará el origen de la estructura enlazada.*

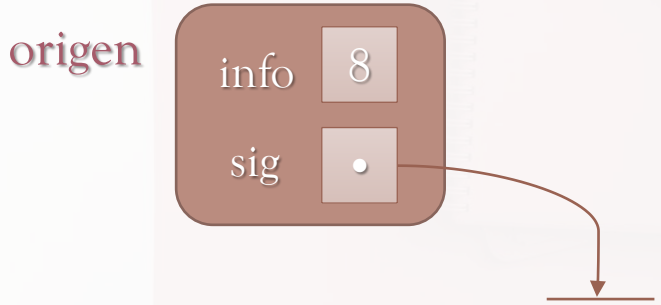
Pila - implementación dinámica



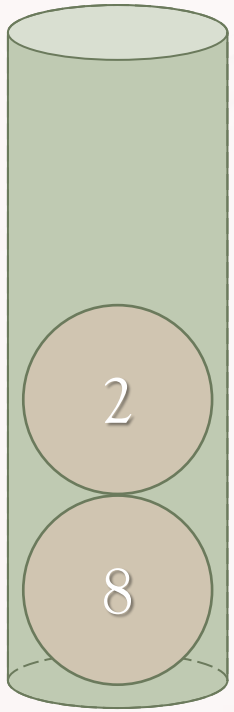
origen

Pila - implementación dinámica

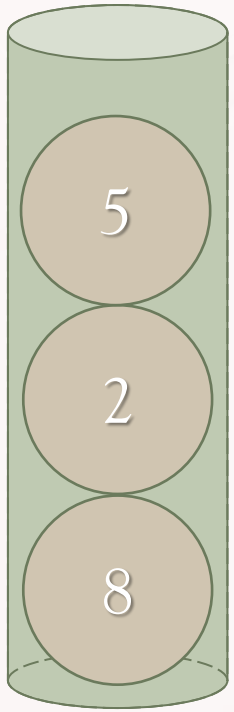


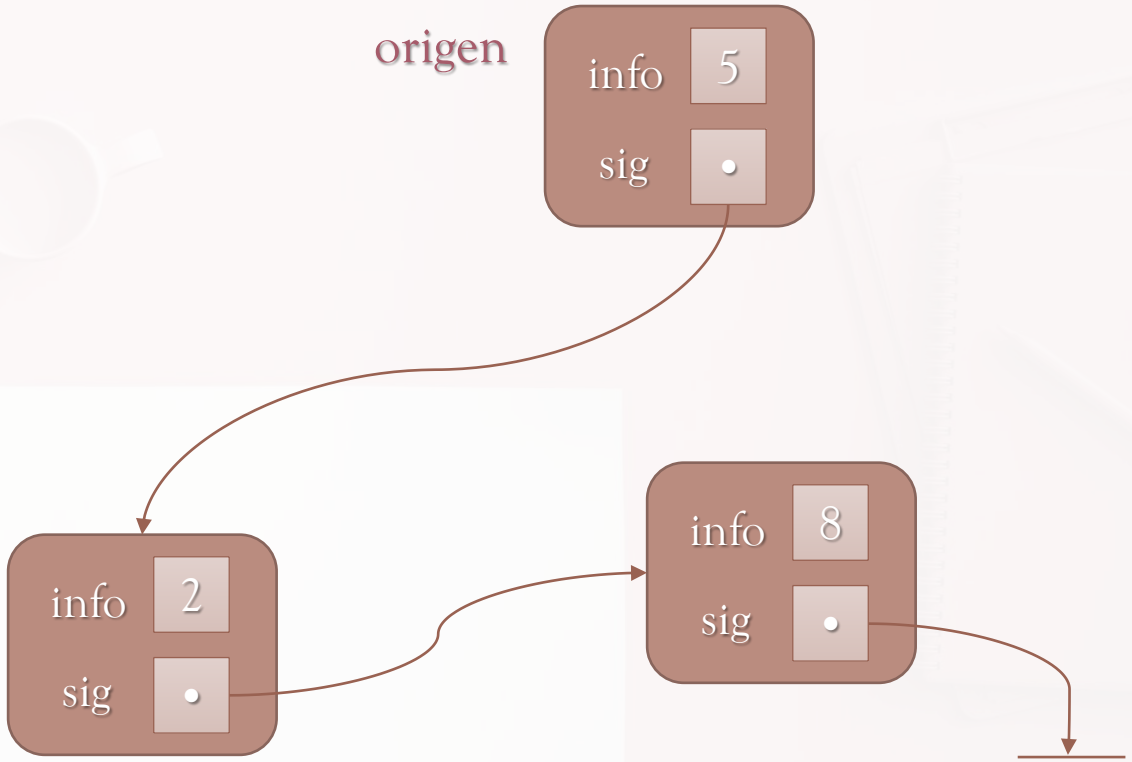


Pila - implementación dinámica

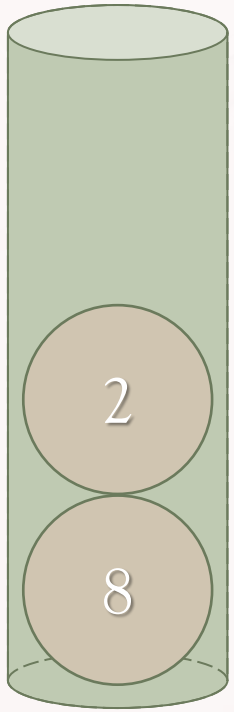


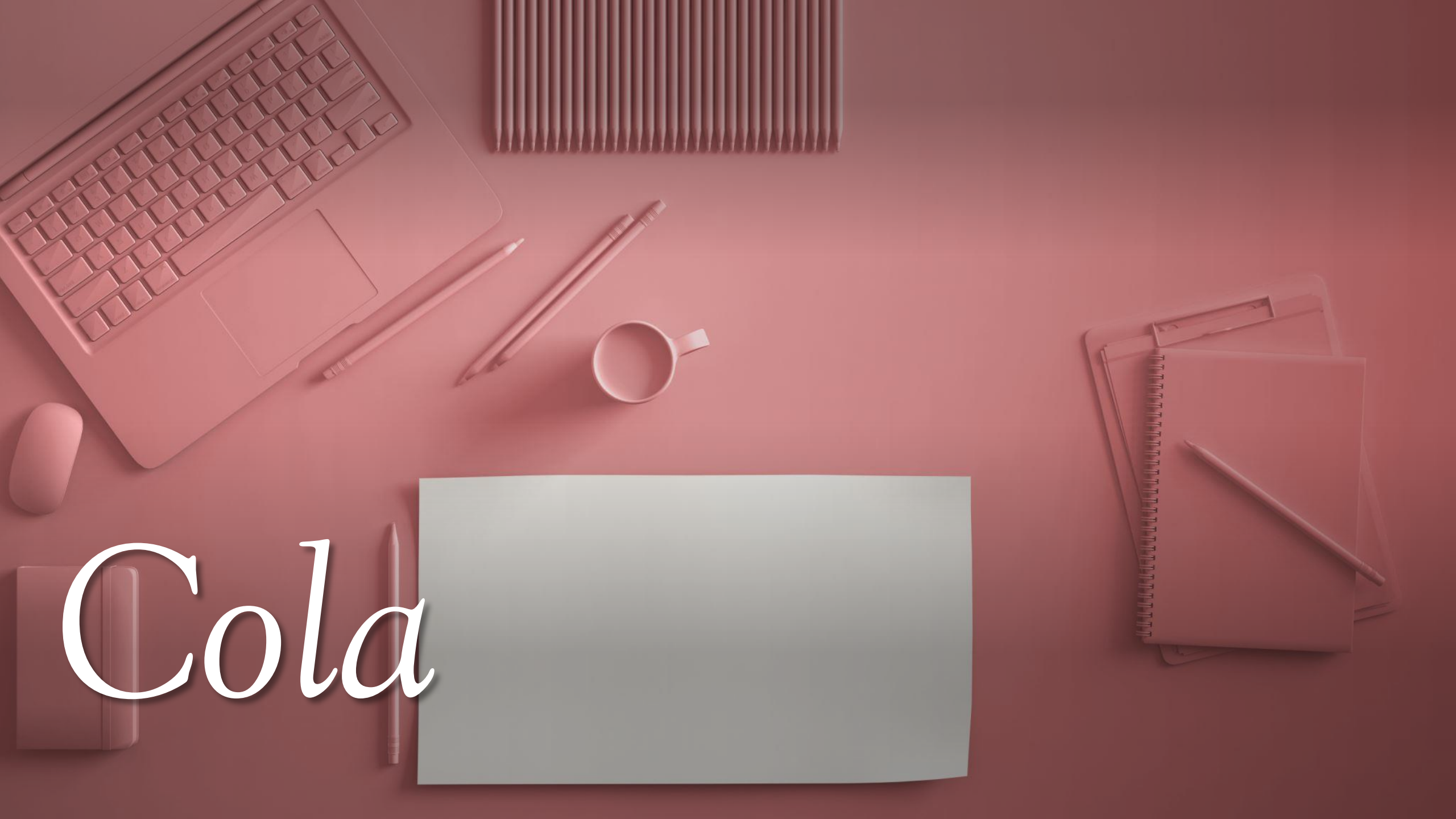
Pila - implementación dinámica





Pila - implementación dinámica




Cola

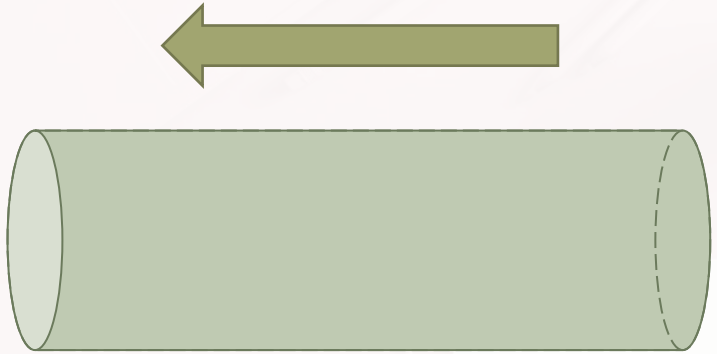
Cola - implementación dinámica

- Se mantendrá un **puntero "origen" al primer elemento**.
- Cada nodo corresponde a un elemento acolado.
- Si la lista está vacía, será un puntero null.
- Los nodos se agregarán al final de la lista. Se tendrá también la referencia al "último" elemento (**inserción al final con doble referencia**), para evitar recorrer toda la lista al acolar.

Cola - implementación dinámica

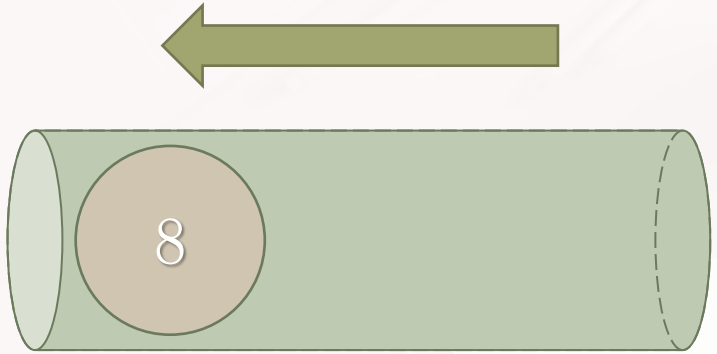
- *Para eliminar un nodo (desacolar), se elimina el primer nodo, dejando el puntero apuntando al segundo, si es que existe (**eliminar el origen**).*
- *Esto significa entonces que cada operación acolar cambiará la referencia al último elemento y cada operación desacolar cambiará el origen de la estructura enlazada.*

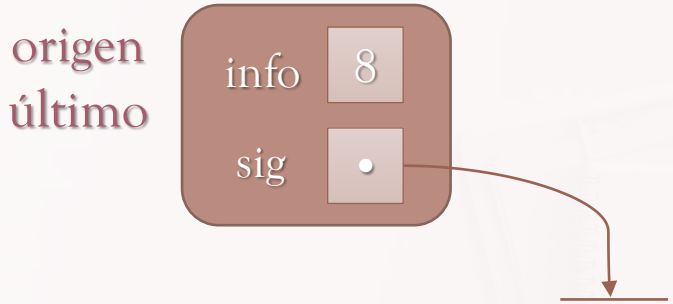
Cola - implementación dinámica



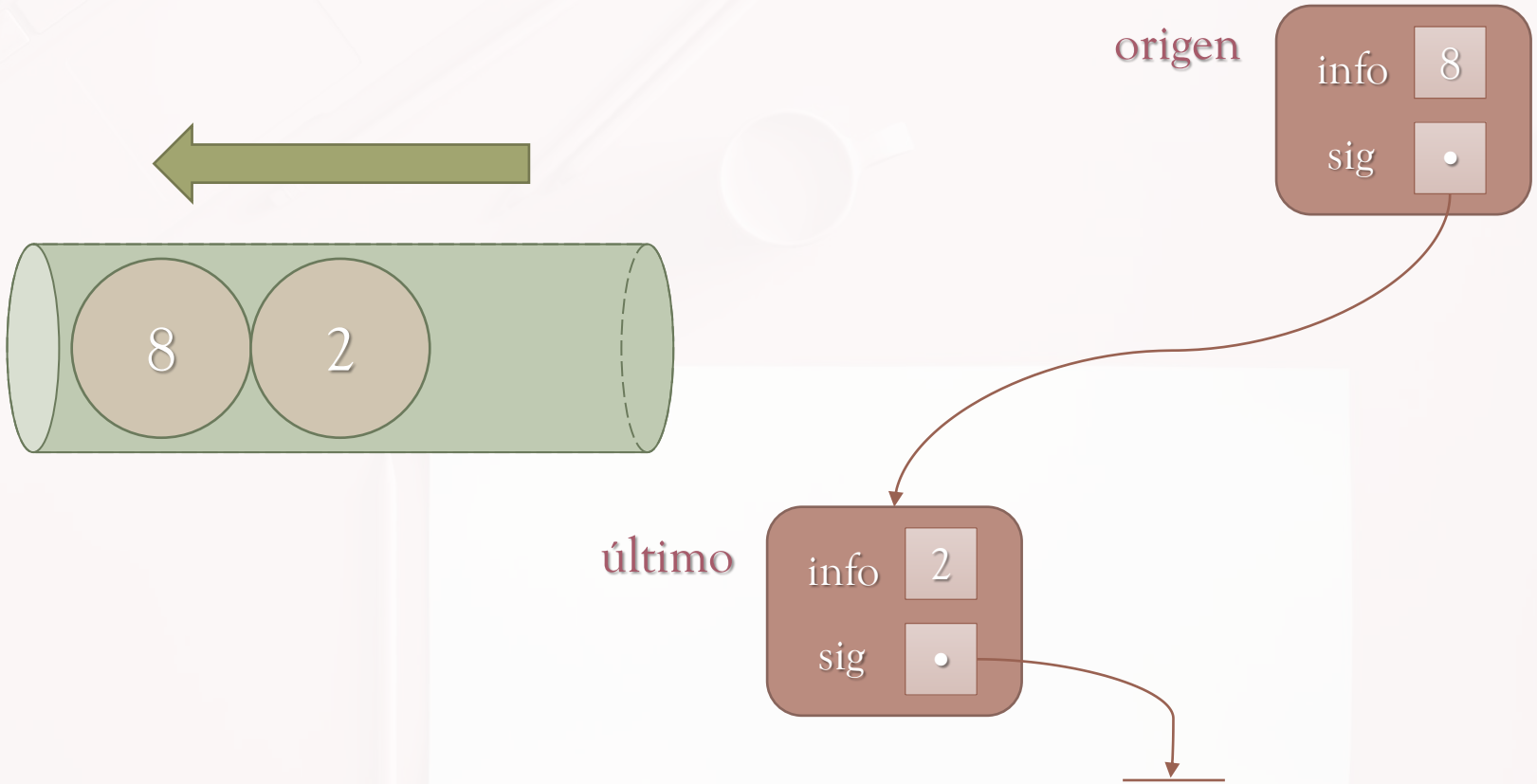
origen
último

Cola - implementación dinámica

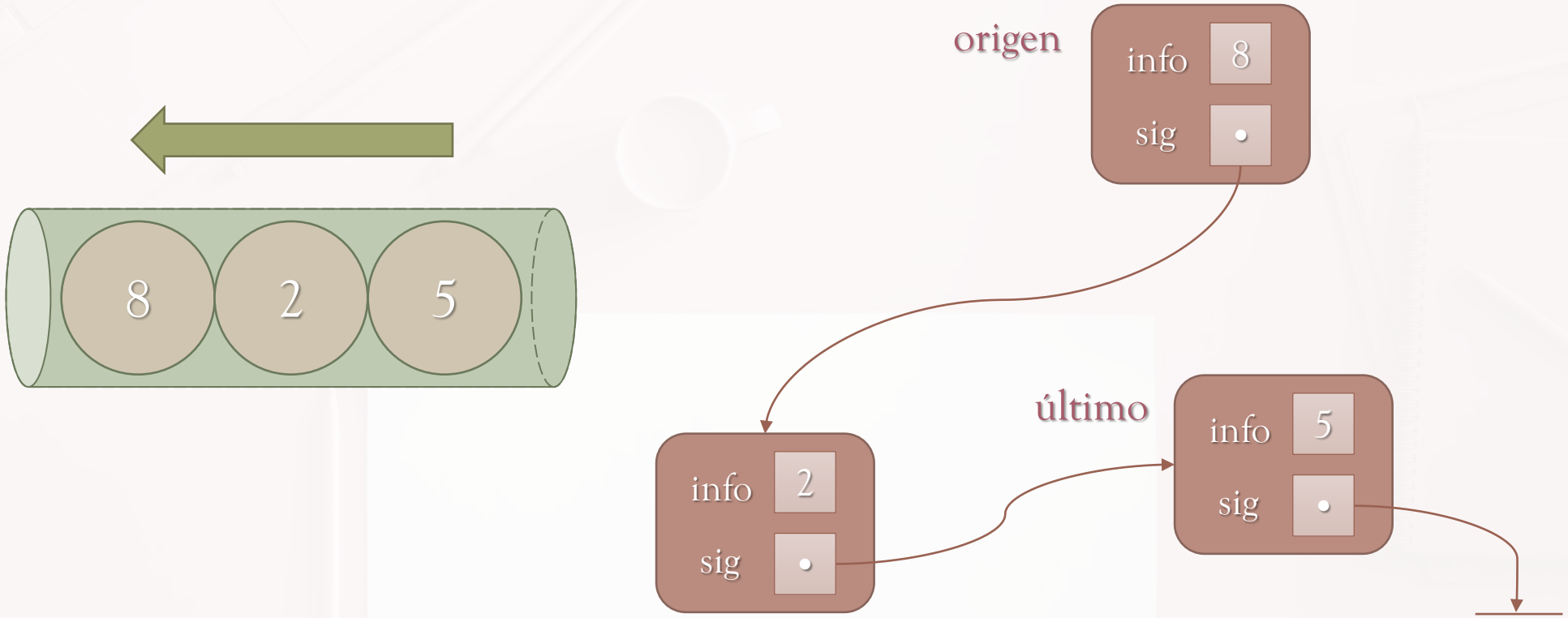




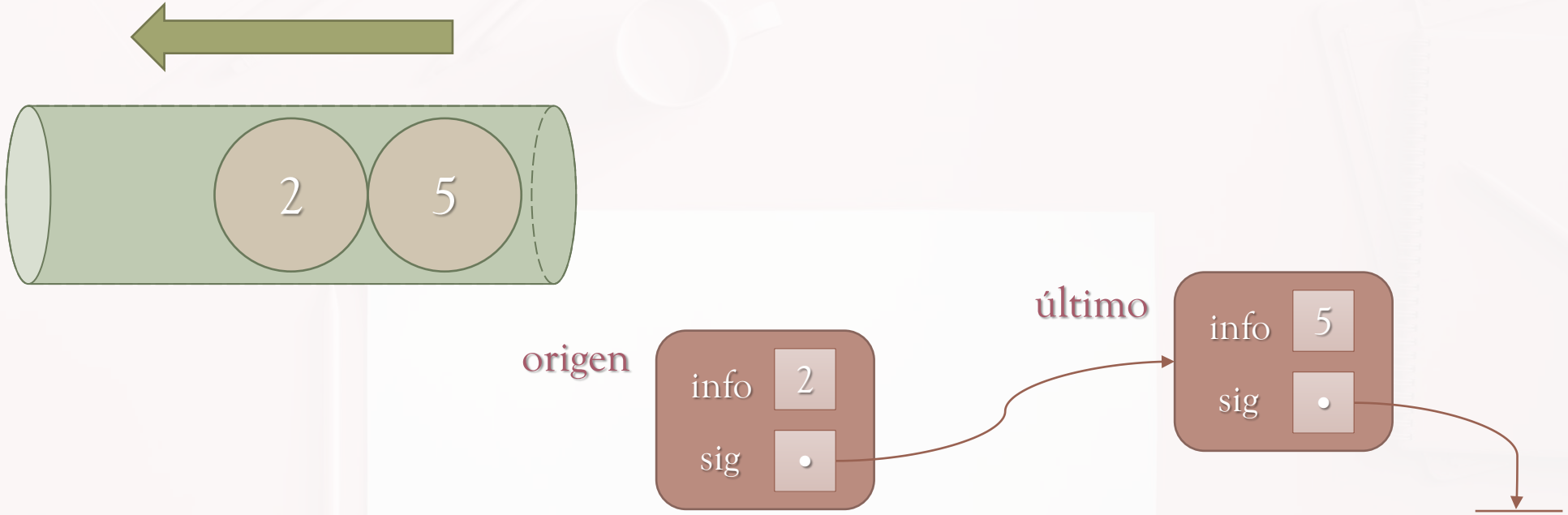
Cola - implementación dinámica



Cola - *implementación dinámica*



Cola - implementación dinámica




Cola con
Prioridad

Cola con prioridad - implementación dinámica

- La ***referencia*** de la estructura enlazada será el ***primer nodo "origen"*** (el elemento con mayor prioridad, que es el primero que se deberá eliminar). En este caso, no es necesaria una segunda referencia al último.
- Cada nodo corresponde a un elemento acolado.
- Si la lista está vacía, será un puntero null.

Cola con prioridad - implementación dinámica

- Los nodos se agregarán ordenadamente por prioridad, luego por orden de entrada (***inserción ordenada***).
- Para eliminar un nodo (desacolar), se elimina el primer nodo, dejando el puntero apuntando al segundo, si es que existe (***eliminar el origen***).

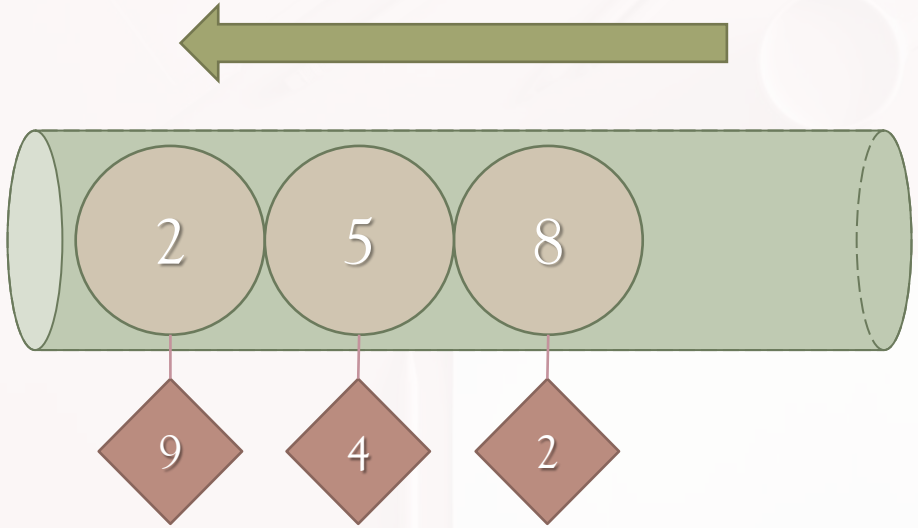
Cola con prioridad - implementación dinámica

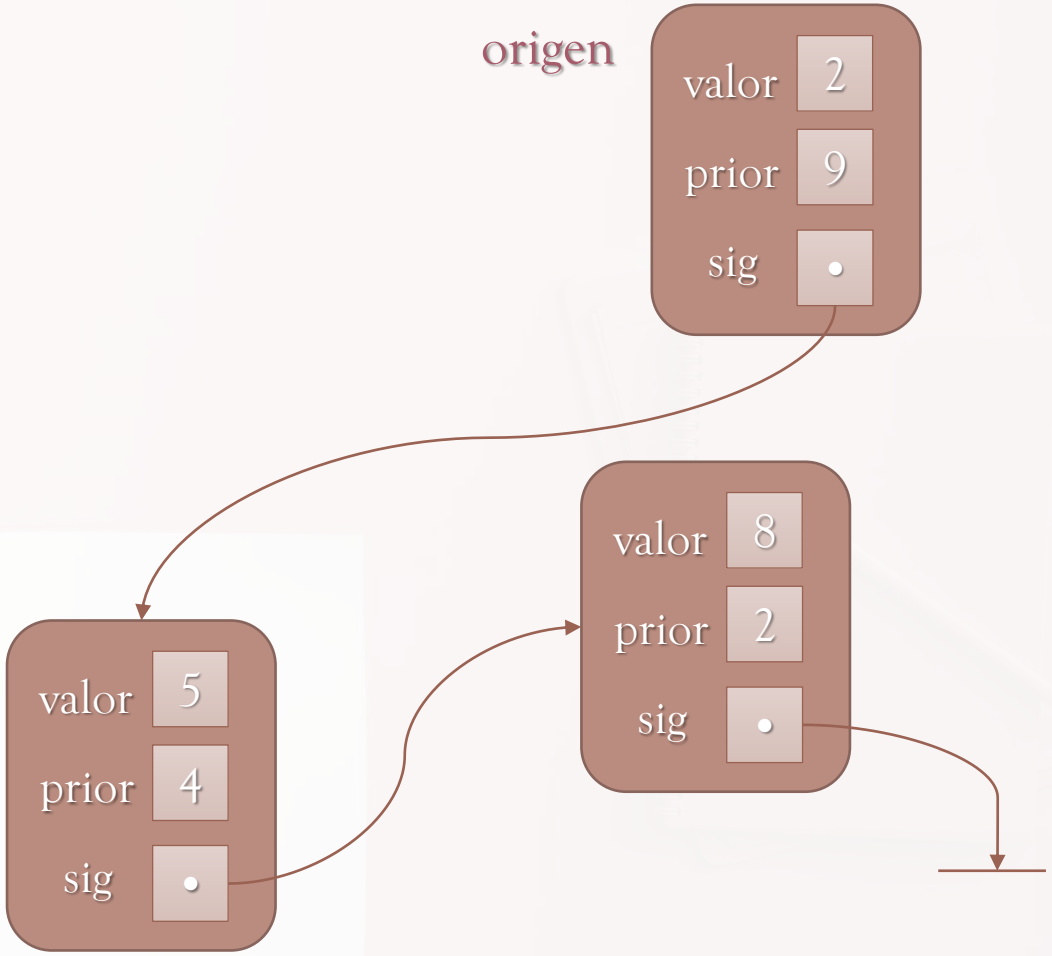
- Esto significa entonces que una operación acolarPrioridad no necesariamente cambiará la referencia al origen de la estructura enlazada, aunque cada operación desacolar sí.
- Se modifica el nodo, para que además del valor tenga la prioridad incluida.

NodoPrioridad

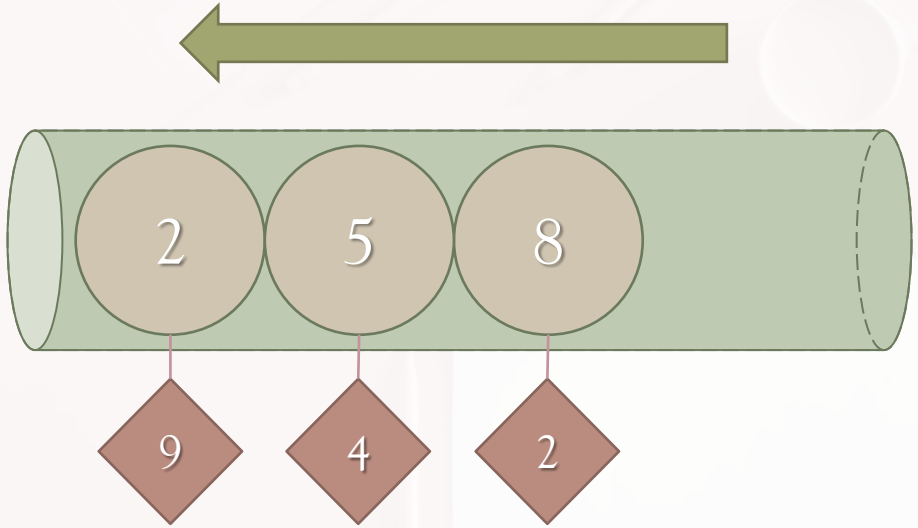
```
class NodoPrioridad {
    int info;
    int prioridad;
    NodoPrioridad sig;
}
```

Cola con prioridad - implementación dinámica

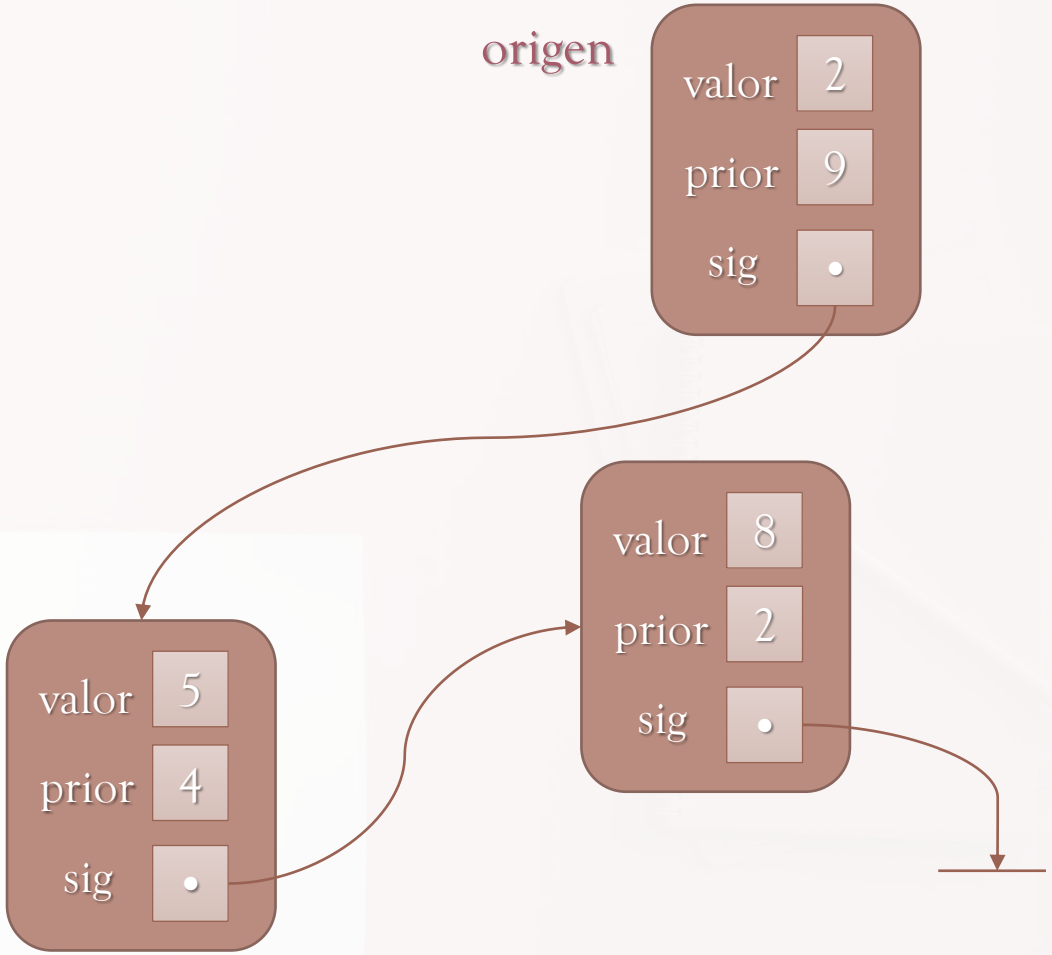




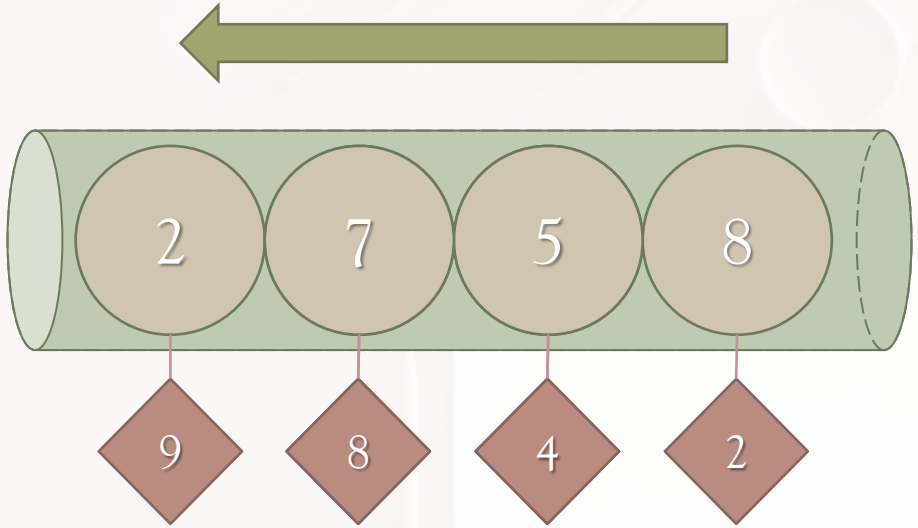
Cola con prioridad - implementación dinámica

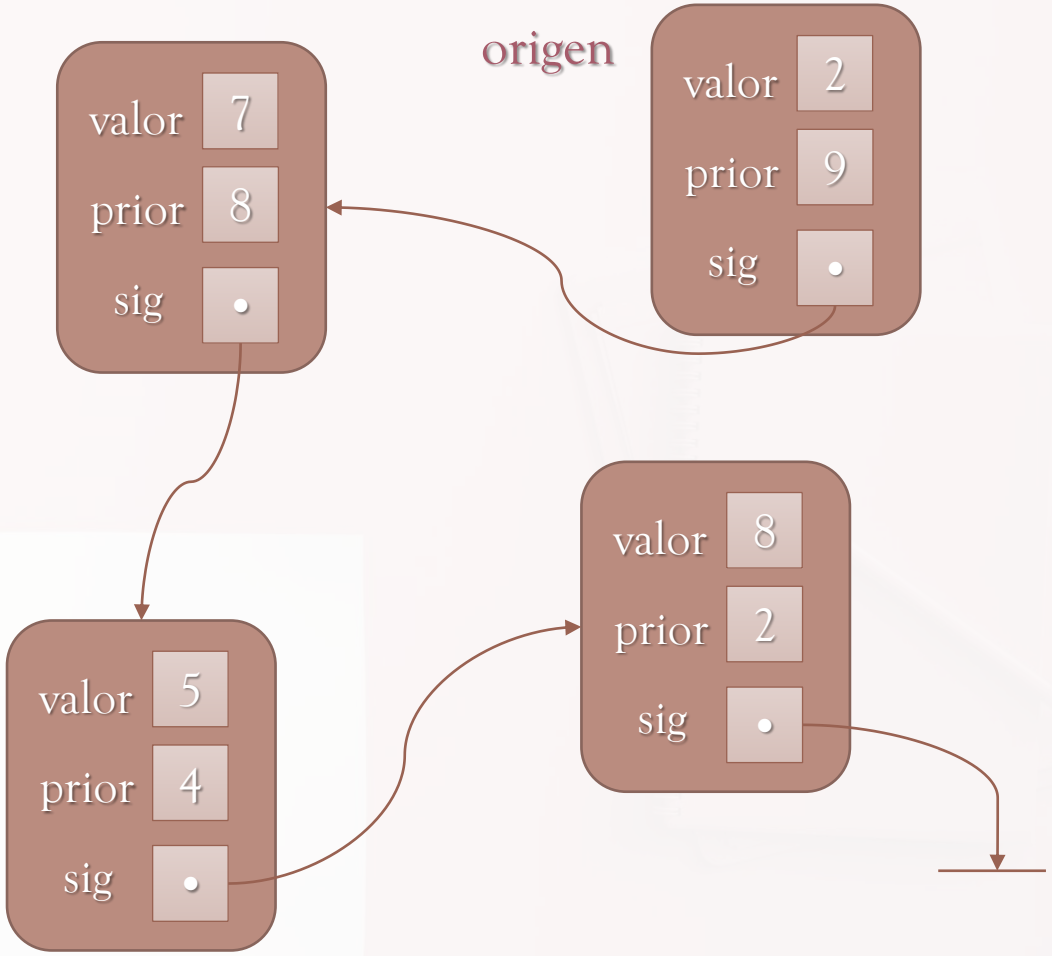


acolarPrioridad(7, 8)



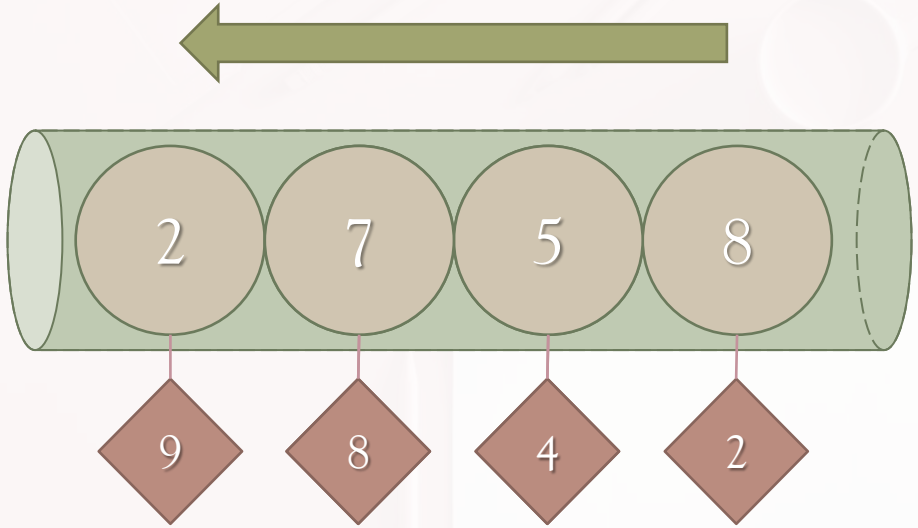
Cola con prioridad - implementación dinámica

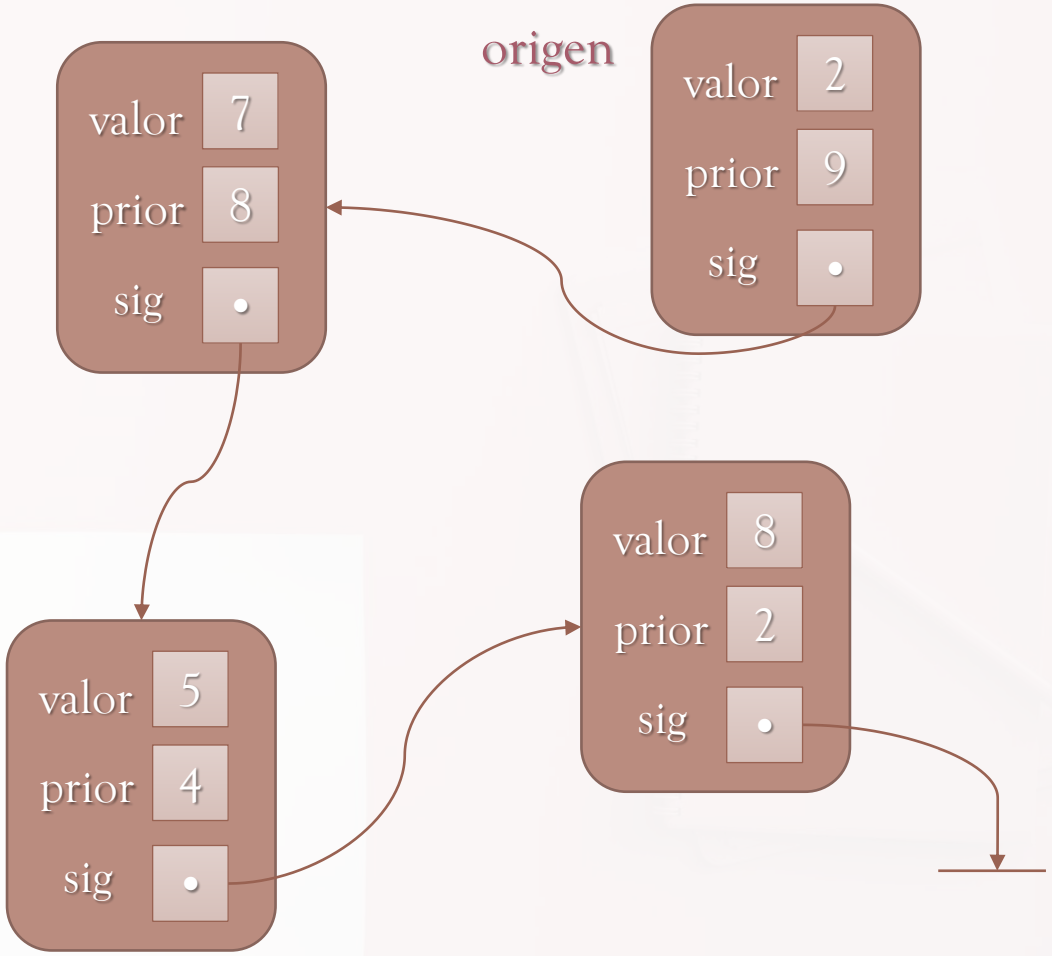




acolarPrioridad(7, 8)

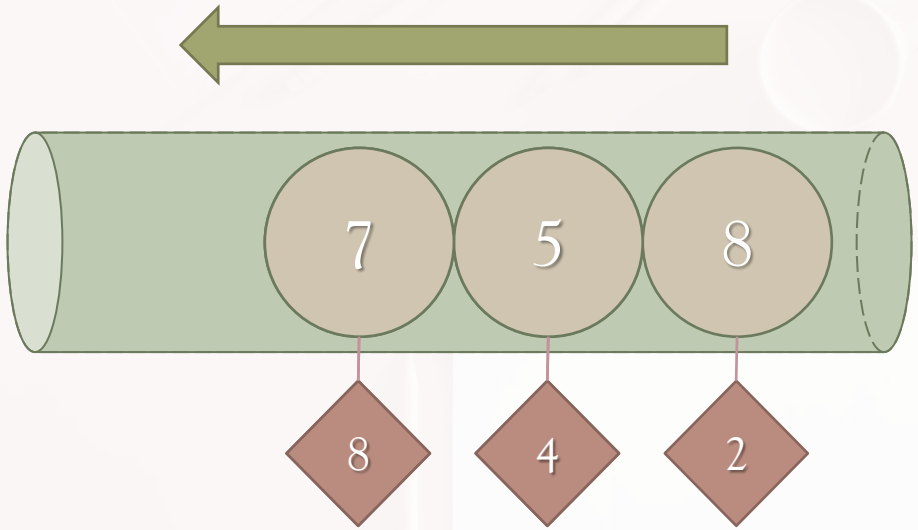
Cola con prioridad - implementación dinámica

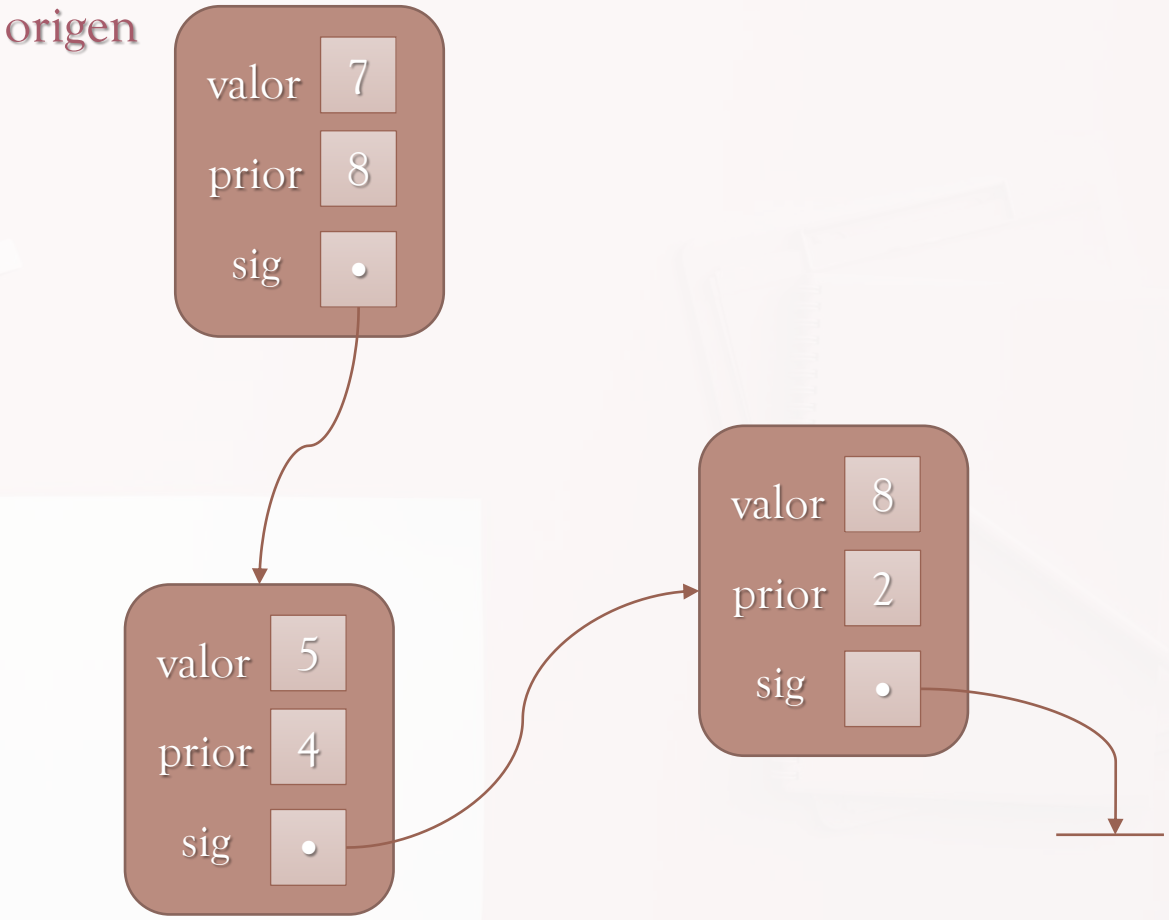




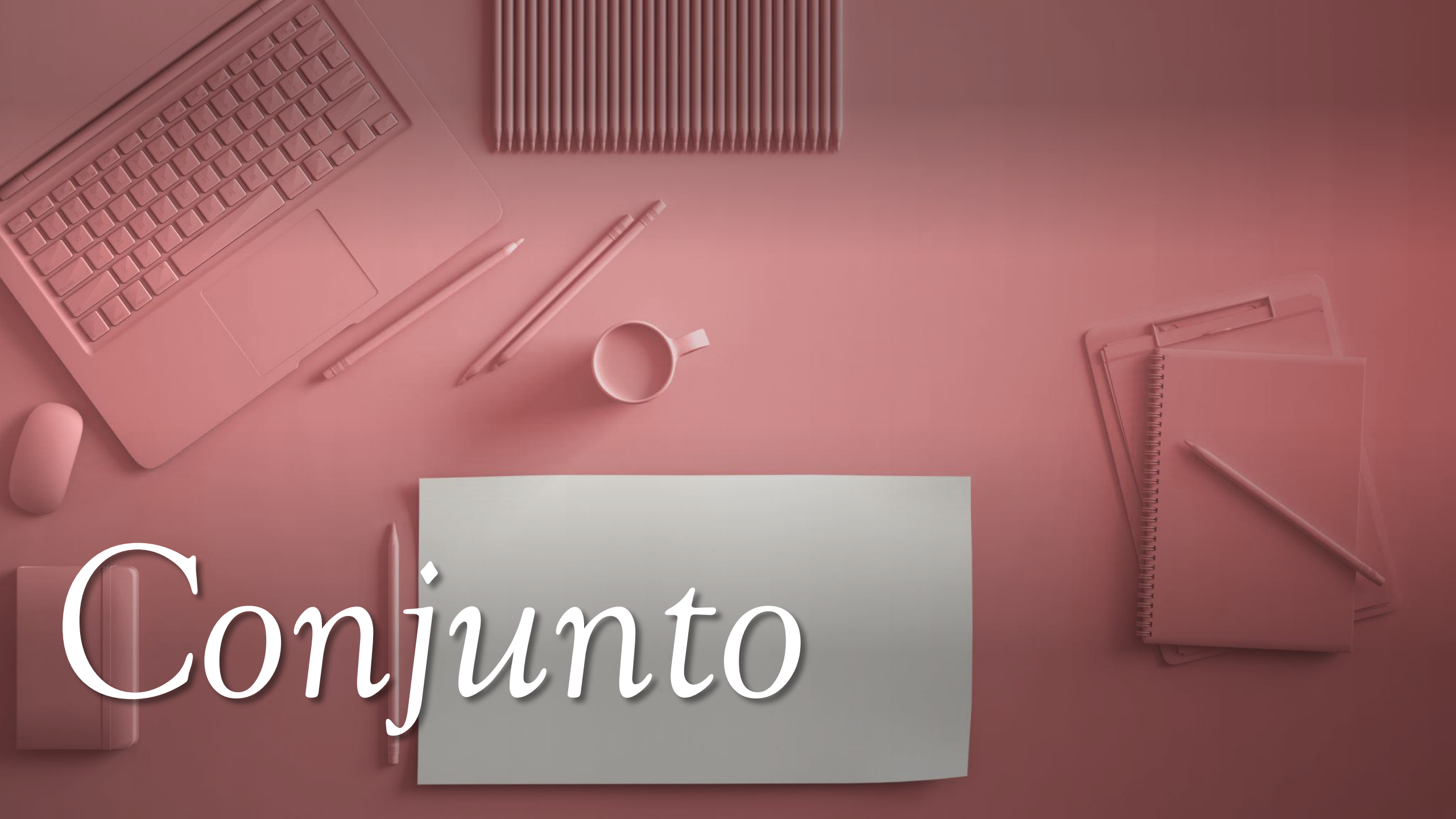
desacolar()

Cola con prioridad - implementación dinámica





desacolar()


Conjunto

Conjunto - implementación dinámica

- La ***referencia*** de la estructura enlazada será el ***primer nodo "origen"***.
- Cada nodo corresponde a un elemento agregado.
- Si la lista está vacía, será un puntero null.
- Dado que ***el orden no importa***, nos limitaremos a mantener la cadena de nodos.

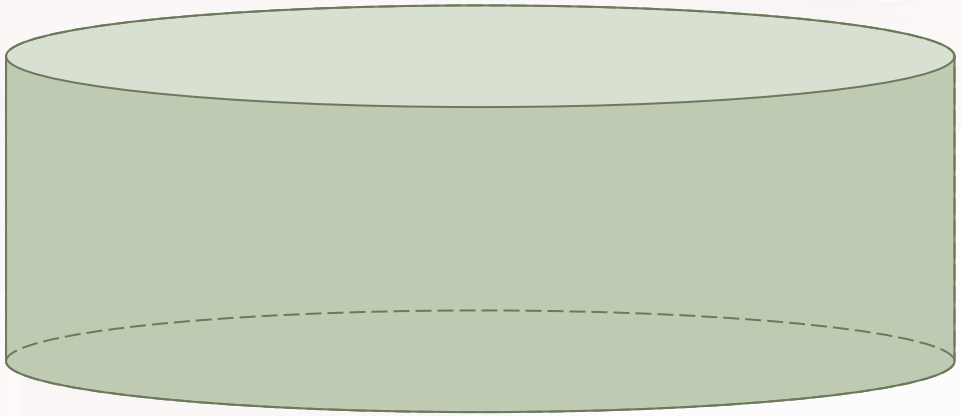
Conjunto - implementación dinámica

- Siempre se deberá ***recorrer*** toda la estructura, tanto para agregar como para eliminar elementos, y para determinar pertenencia.
- Los nodos se agregarán al principio (***inserción al principio***), siempre y cuando no existan ya. De esta forma, la referencia a la estructura cambia con cada elemento que se agrega al conjunto.

Conjunto - implementación dinámica

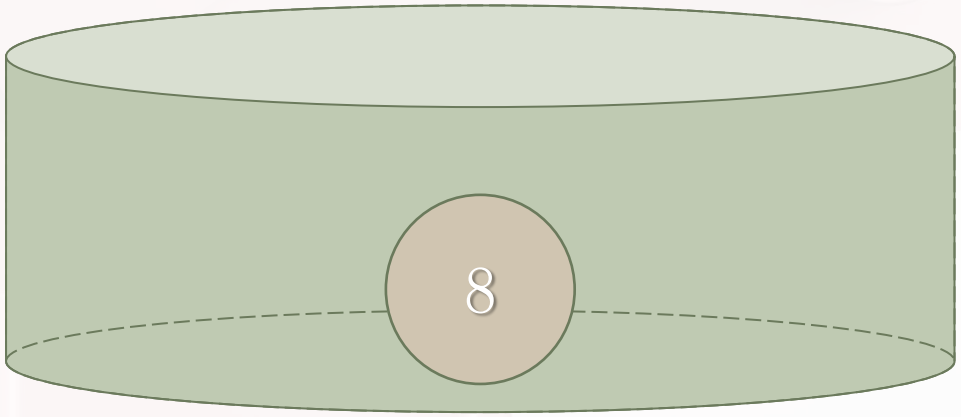
- *Para eliminar un nodo (sacar), se busca el mismo con un puntero auxiliar y se lo circunvala, dejándolo inaccesible (**eliminación** común).*

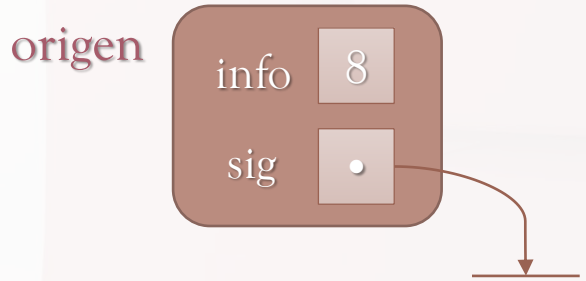
Conjunto - implementación dinámica



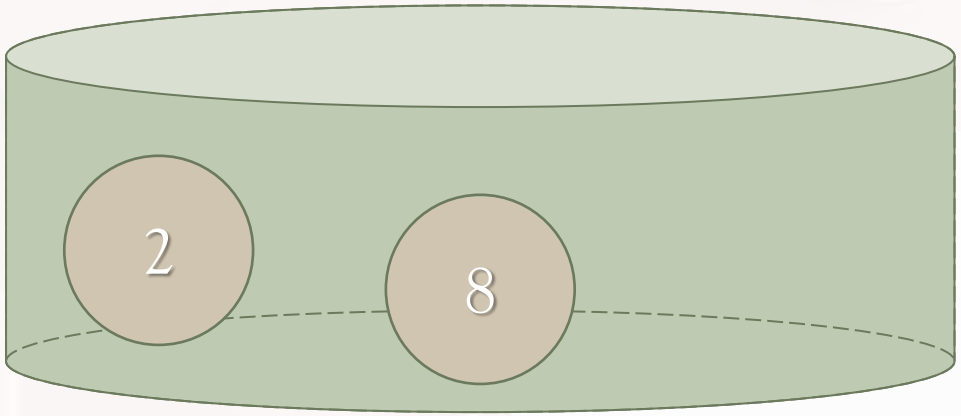
origen

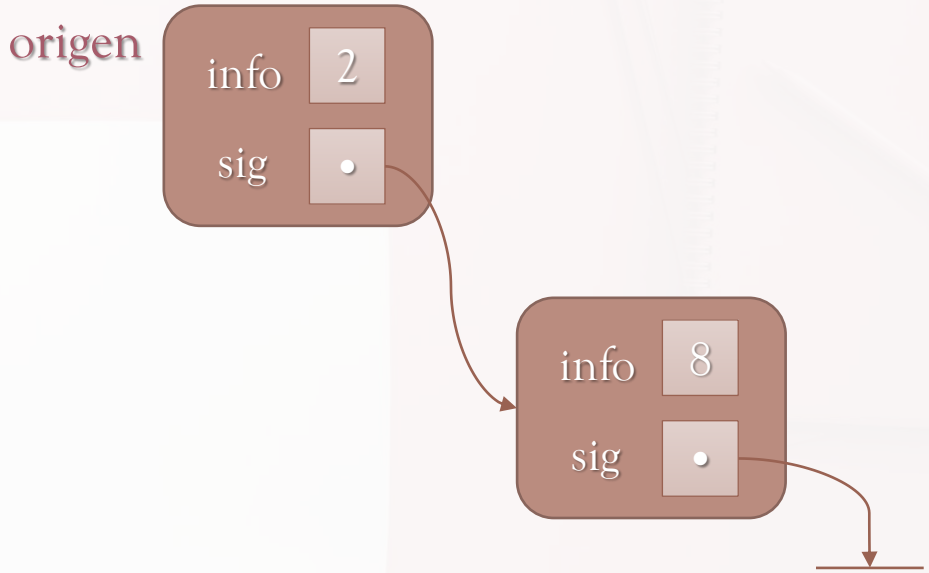
Conjunto - implementación dinámica



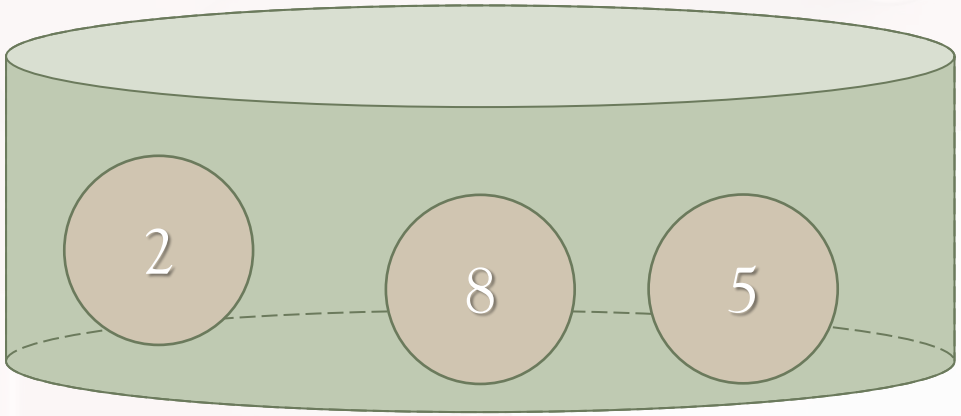


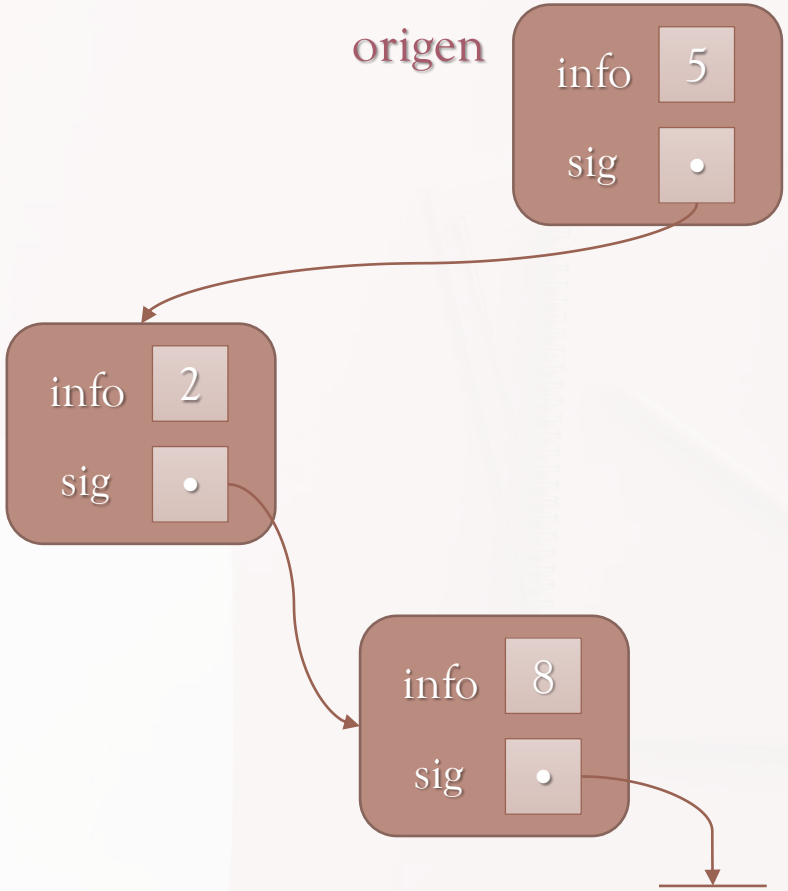
Conjunto - implementación dinámica



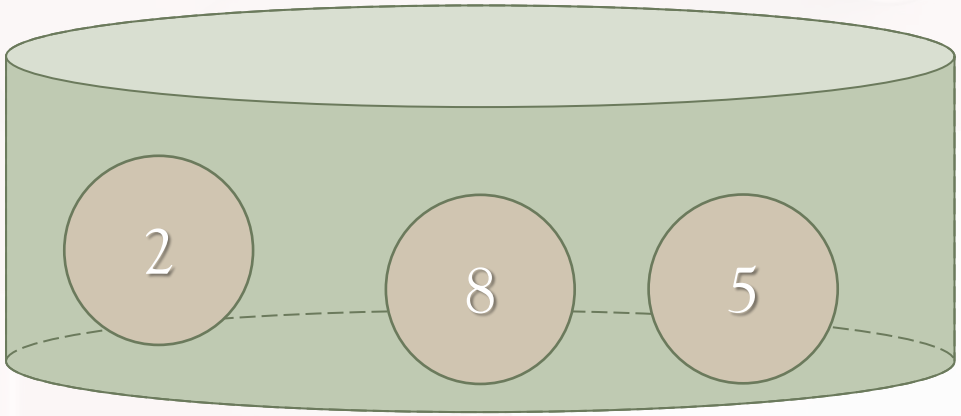


Conjunto - implementación dinámica

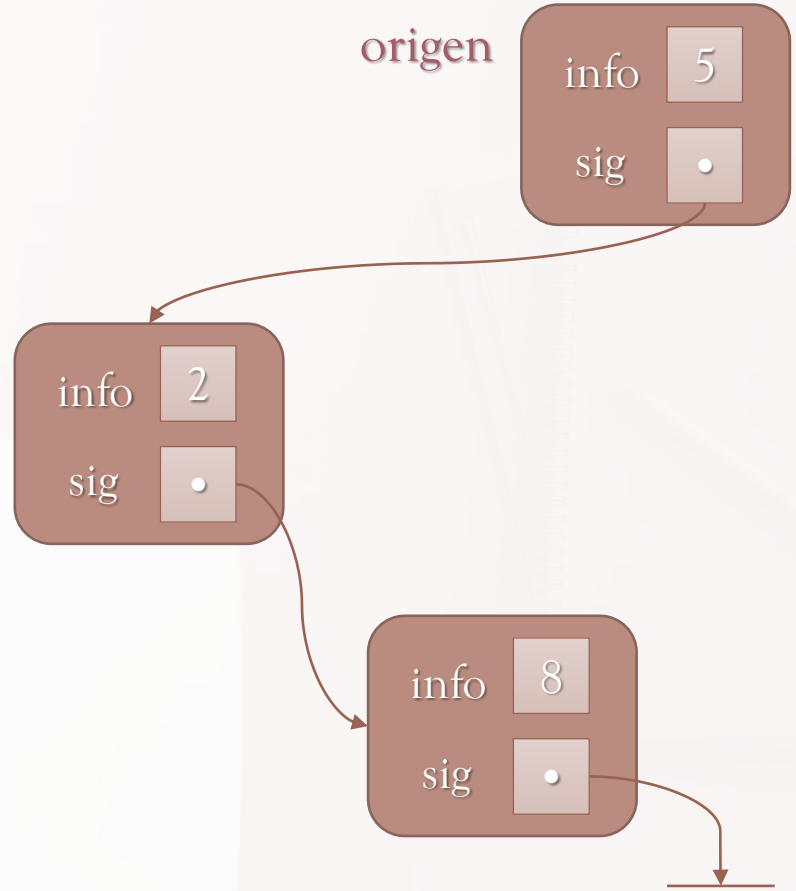




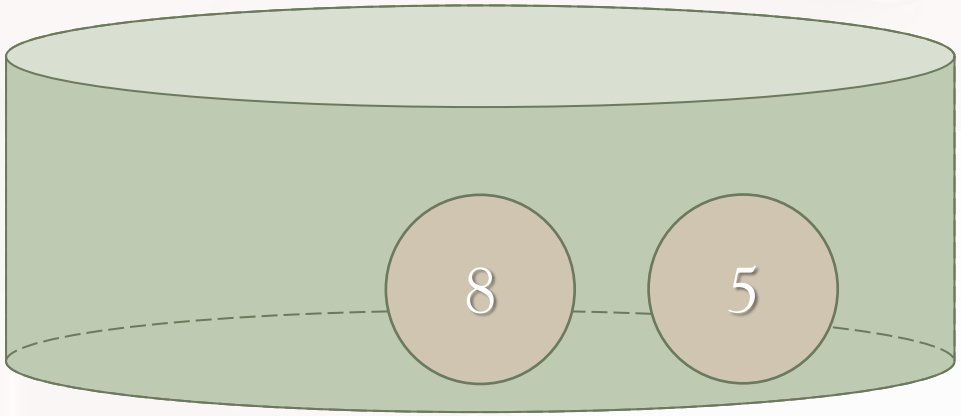
Conjunto - implementación dinámica



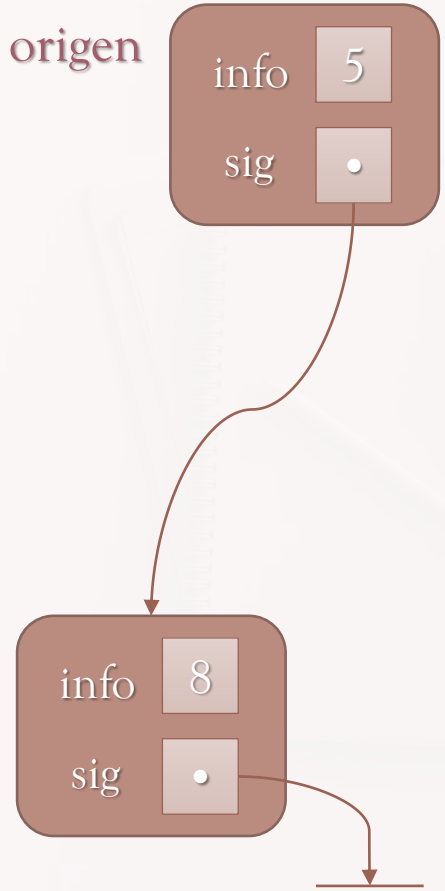
sacar(2)



Conjunto - implementación dinámica



sacar(2)




Diccionario
Simple

Diccionario simple - implementación dinámica

- *La **referencia** de la estructura enlazada será el **primer nodo "origen"**.*
- *Cada nodo corresponde a un elemento agregado.*
- *Si la lista está vacía, será un puntero null.*
- *Como **no tenemos orden**, nos limitamos a mantener el conjunto de nodos, y siempre deberá recorrerse para buscar un elemento o eliminarlo.*

Diccionario simple - implementación dinámica

- *Los nodos se agregarán al principio (**inserción al principio**), siempre y cuando no exista ya la clave; sino se buscará el nodo por clave y se sobrescribirá el valor. De esta forma, la referencia a la estructura cambia con cada elemento que se agrega al diccionario.*

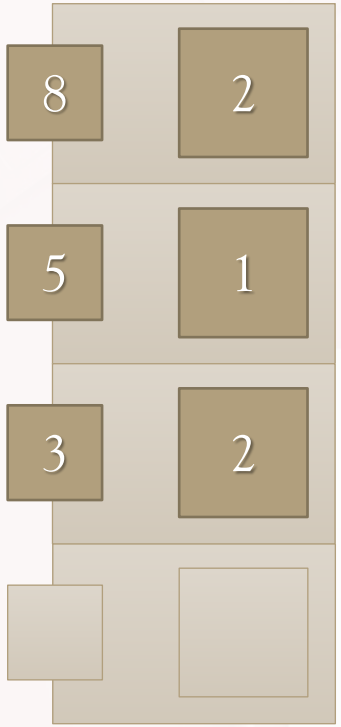
Diccionario simple - implementación dinámica

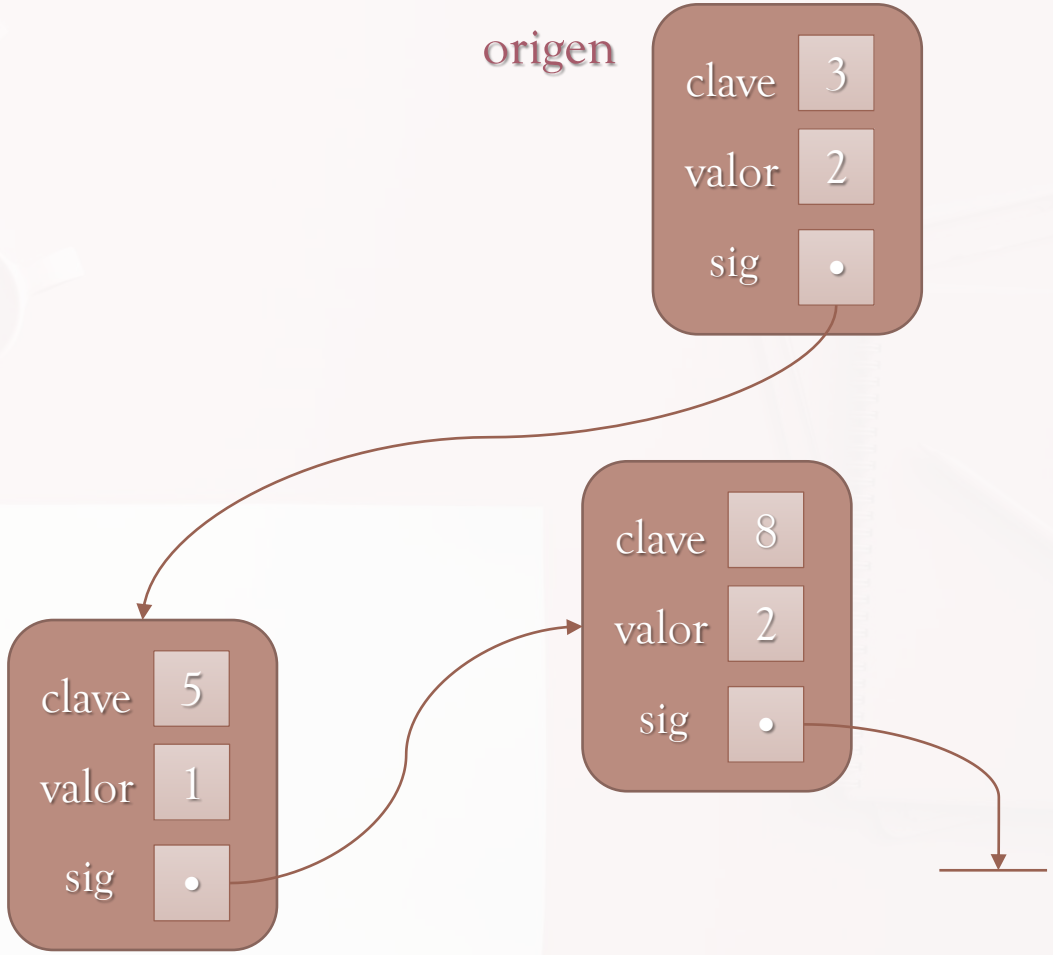
- *Para eliminar un nodo (eliminar), se busca la clave con un puntero auxiliar y se circunvala el nodo, dejándolo inaccesible (**eliminación** común).*
- *Los nodos contienen, además del valor, la clave.*

NodoClave

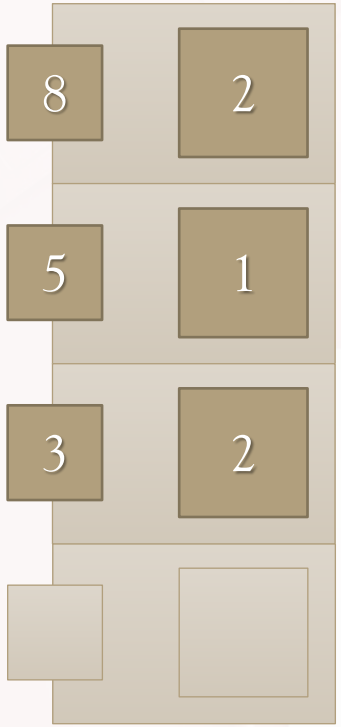
```
class NodoClave {
    int clave;
    int valor;
    NodoClave sigClave;
}
```

Diccionario simple - implementación dinámica

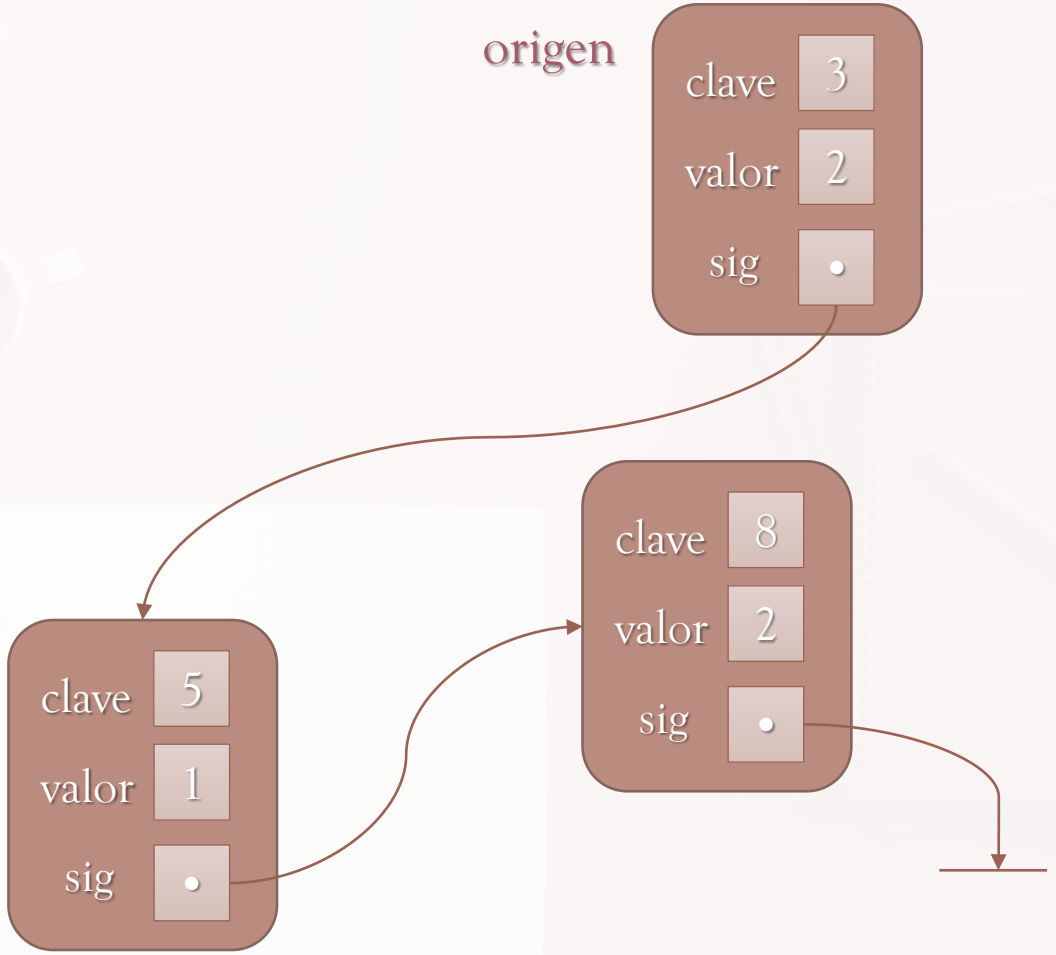




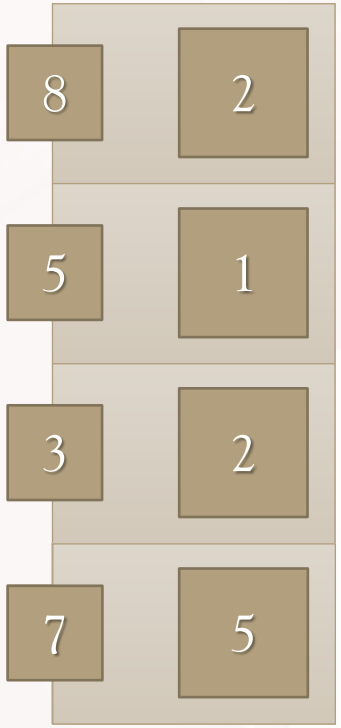
Diccionario simple - implementación dinámica



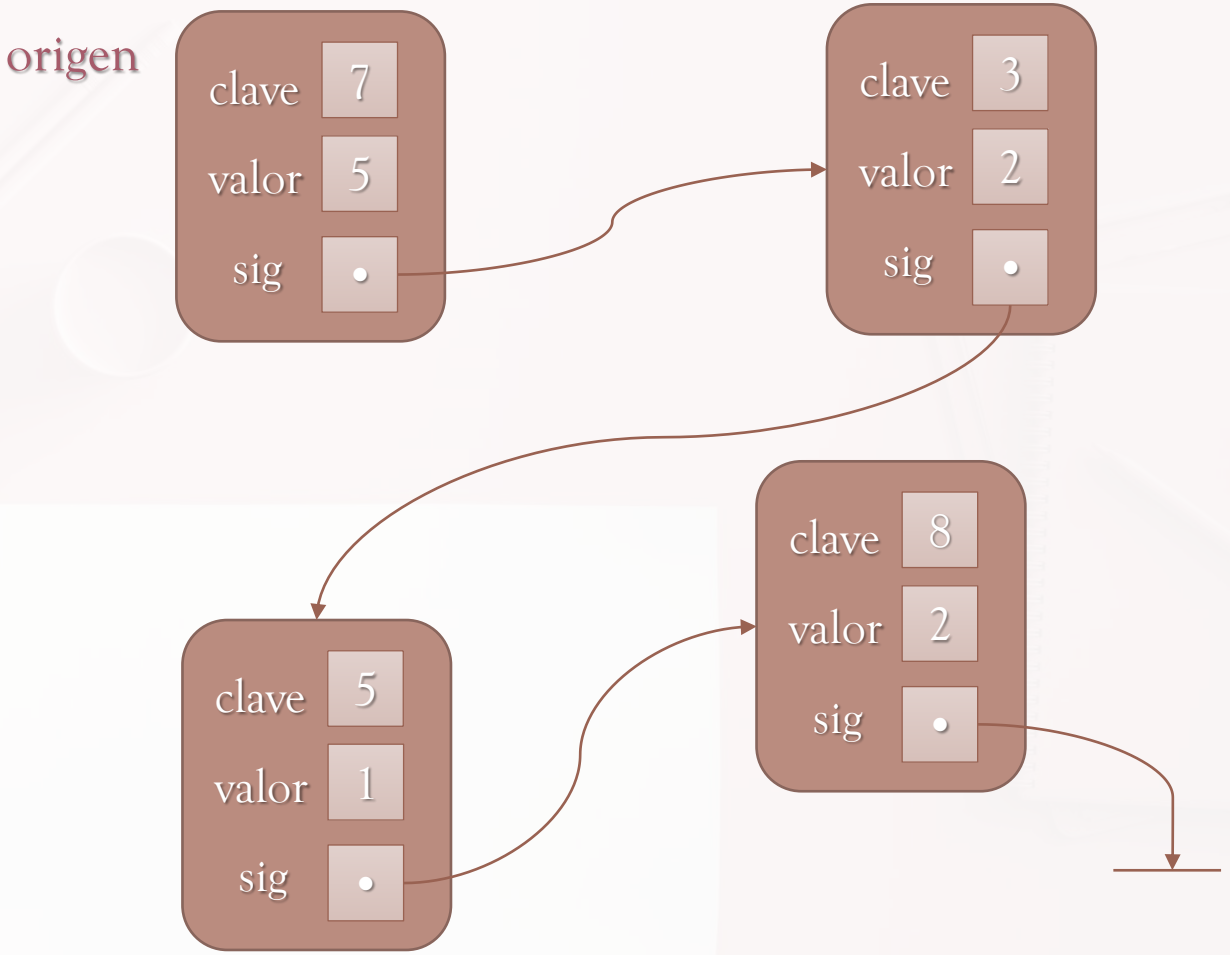
agregar(7, 5)



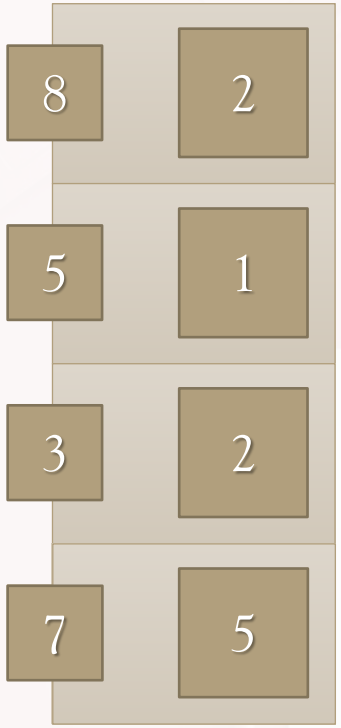
Diccionario simple - implementación dinámica



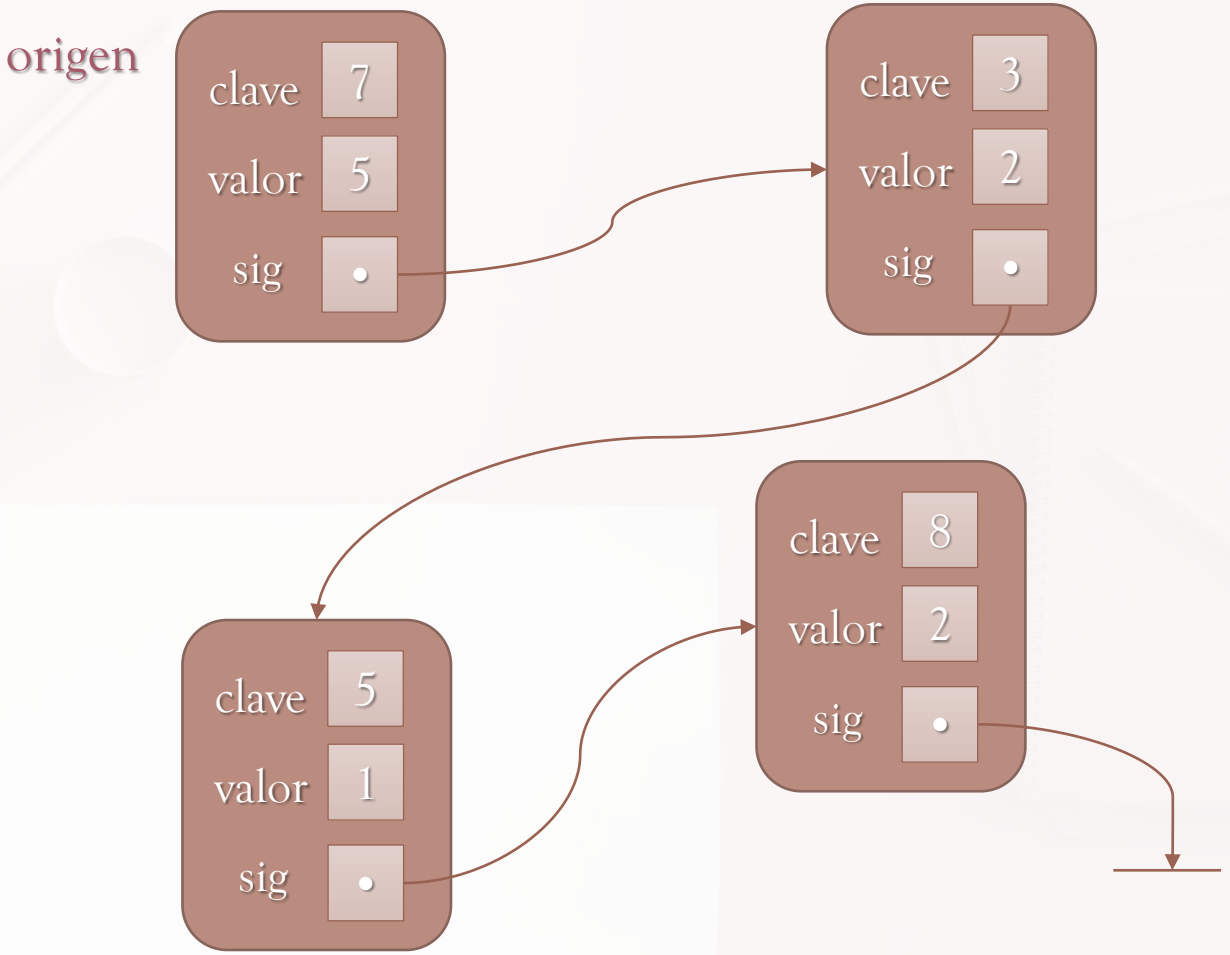
agregar(7, 5)



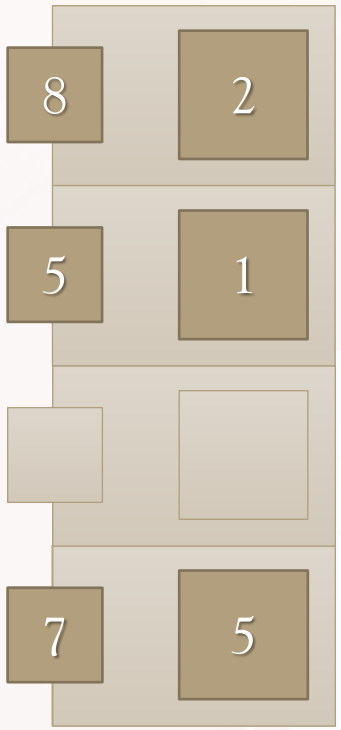
Diccionario simple - implementación dinámica



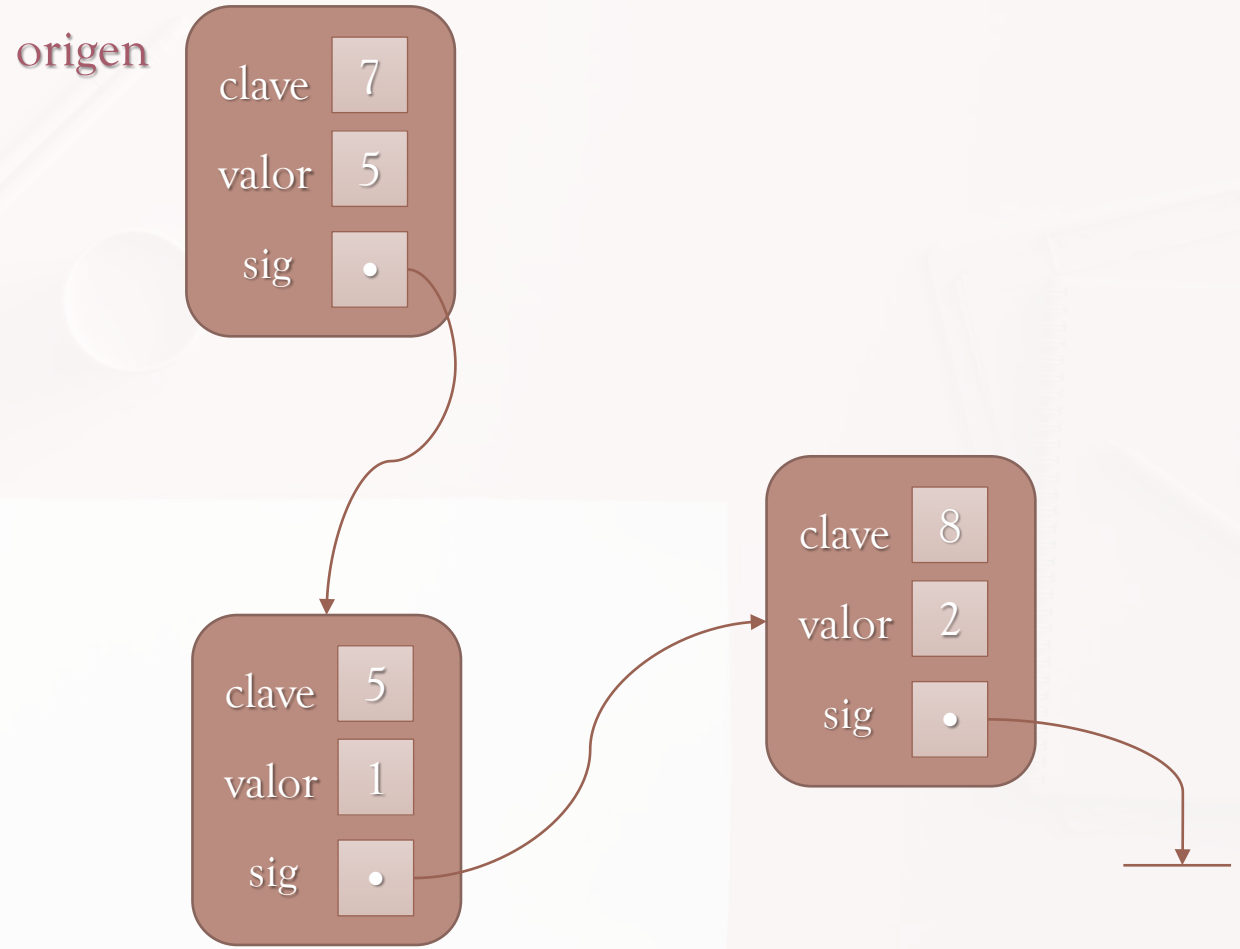
eliminar(3)



Diccionario simple - implementación dinámica



eliminar(3)




Diccionario
Múltiple

Diccionario múltiple - implementación dinámica

- *Existen **dos tipos de nodos**: claves y valores.*
- *Cada clave es el inicio de una lista enlazada de valores que corresponde a su conjunto de valores.*
- *Tendremos una **lista enlazada de nodos NodoClave**, que **se ramifica en varias listas de valores**, representados con nodos NodoValor.*

NodoClave

```
class NodoClave {
    int clave;
    NodoValor valores;
    NodoClave sigClave;
}
```

NodoValor

```
class NodoValor {
    int valor;
    NodoValor sigValor;
}
```

Diccionario múltiple - implementación dinámica

- *La **referencia** de la estructura enlazada de **claves** será el **primer nodo "origen"**.*
- *Cada nodo corresponde a un elemento agregado, ya sea una clave o un valor.*
- *Si la lista está vacía, será un puntero null.*
- *Como no tenemos orden, nos limitamos a mantener el **conjunto de nodos** (ya sea NodoClave o NodoValor).*

Diccionario múltiple - implementación dinámica

- *Siempre deberán **recorrerse** dichos conjuntos para buscar un elemento o eliminarlo.*
- *Las nuevas claves se agregan al principio de la lista de claves (**inserción al principio**), siempre y cuando la misma no exista ya. De esta forma, la referencia a la estructura cambia con cada clave que se agrega al diccionario.*

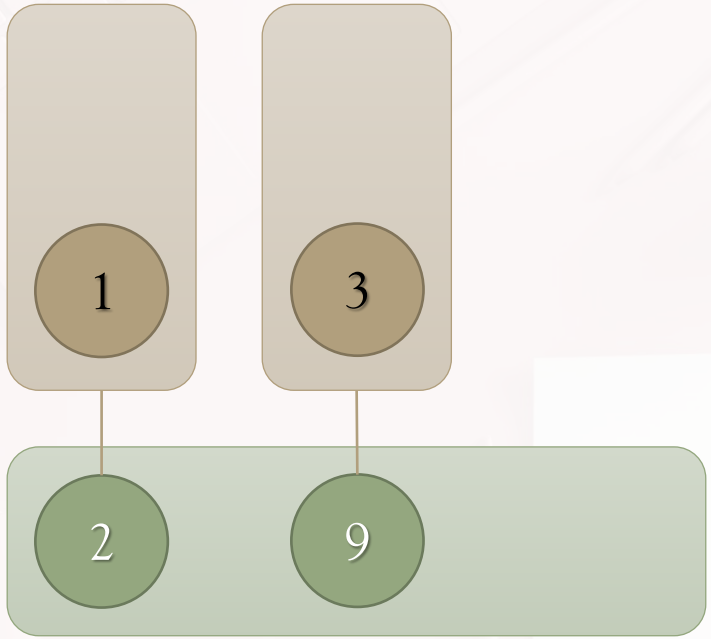
Diccionario múltiple - implementación dinámica

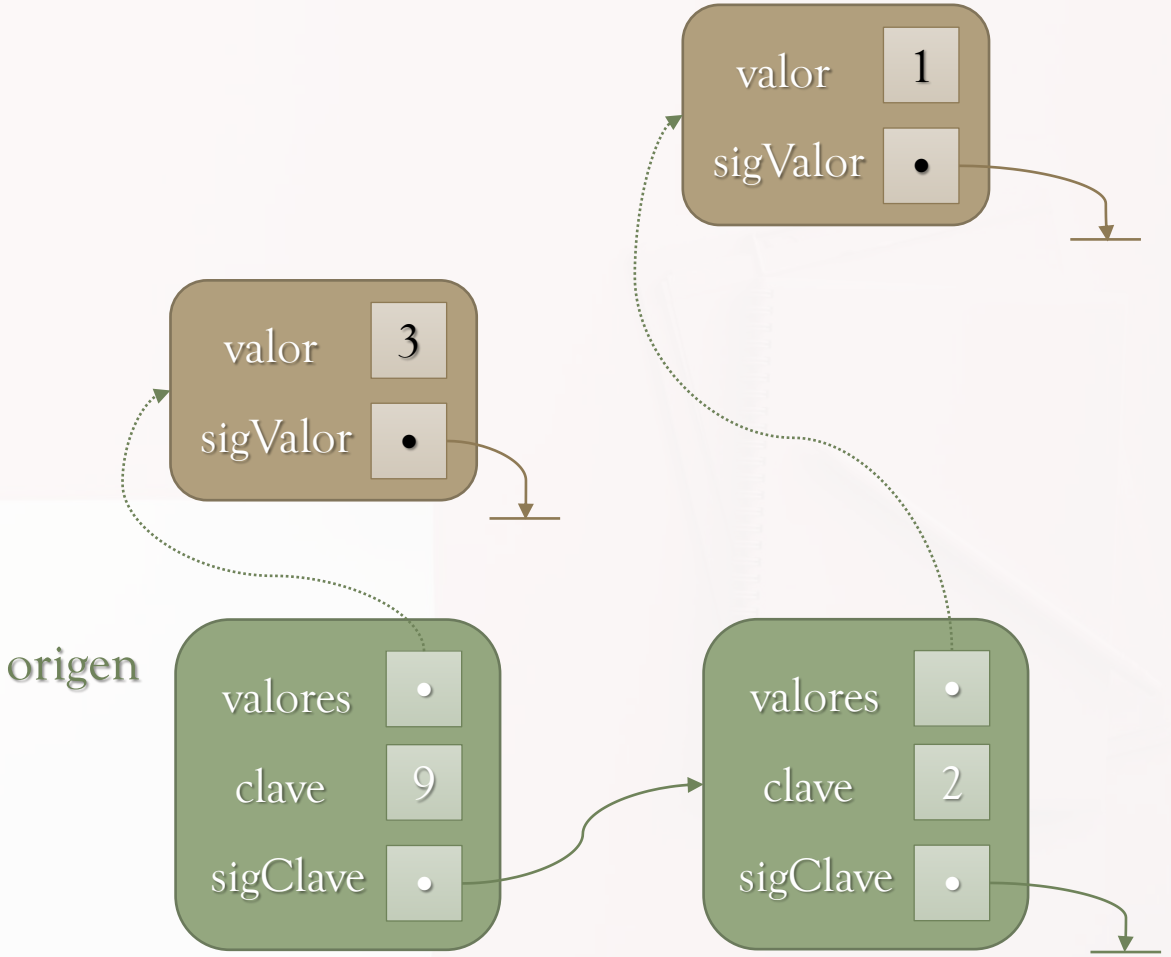
- *Cada clave tiene la referencia al primer nodo de su **lista de valores**.*
- *Los nuevos valores se agregan al principio de estas listas "valores" (**inserción al principio**), siempre y cuando los mismos no existan ya.*
- *Para eliminar una clave (eliminar), se busca la misma con un puntero auxiliar y se circunvala el nodo, dejándolo inaccesible (**eliminación** común).*

Diccionario múltiple - implementación dinámica

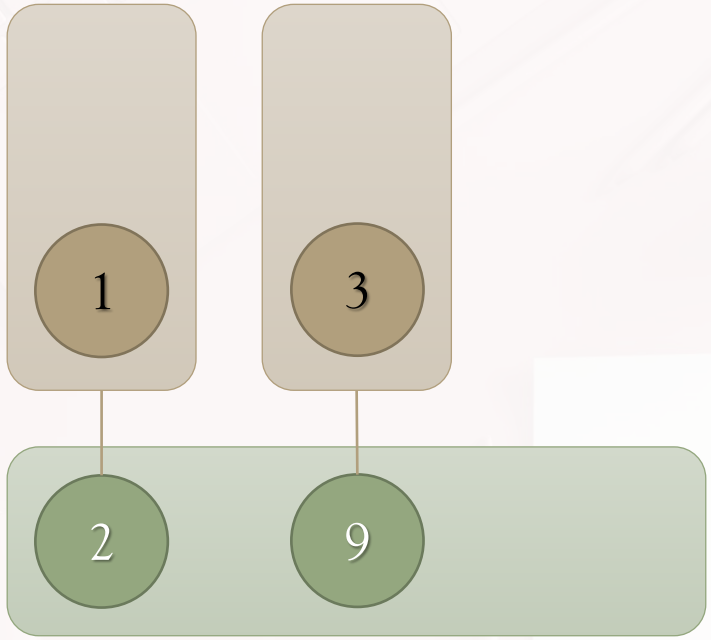
- *Para eliminar un valor de una clave (eliminarValor), se busca esta última con un puntero auxiliar, posteriormente se busca el valor con otro puntero auxiliar y se circunvala el nodo, dejándolo inaccesible (**eliminación** común). Si el conjunto de valores quedara vacío, se deberá eliminar la clave.*

Diccionario múltiple - implementación dinámica

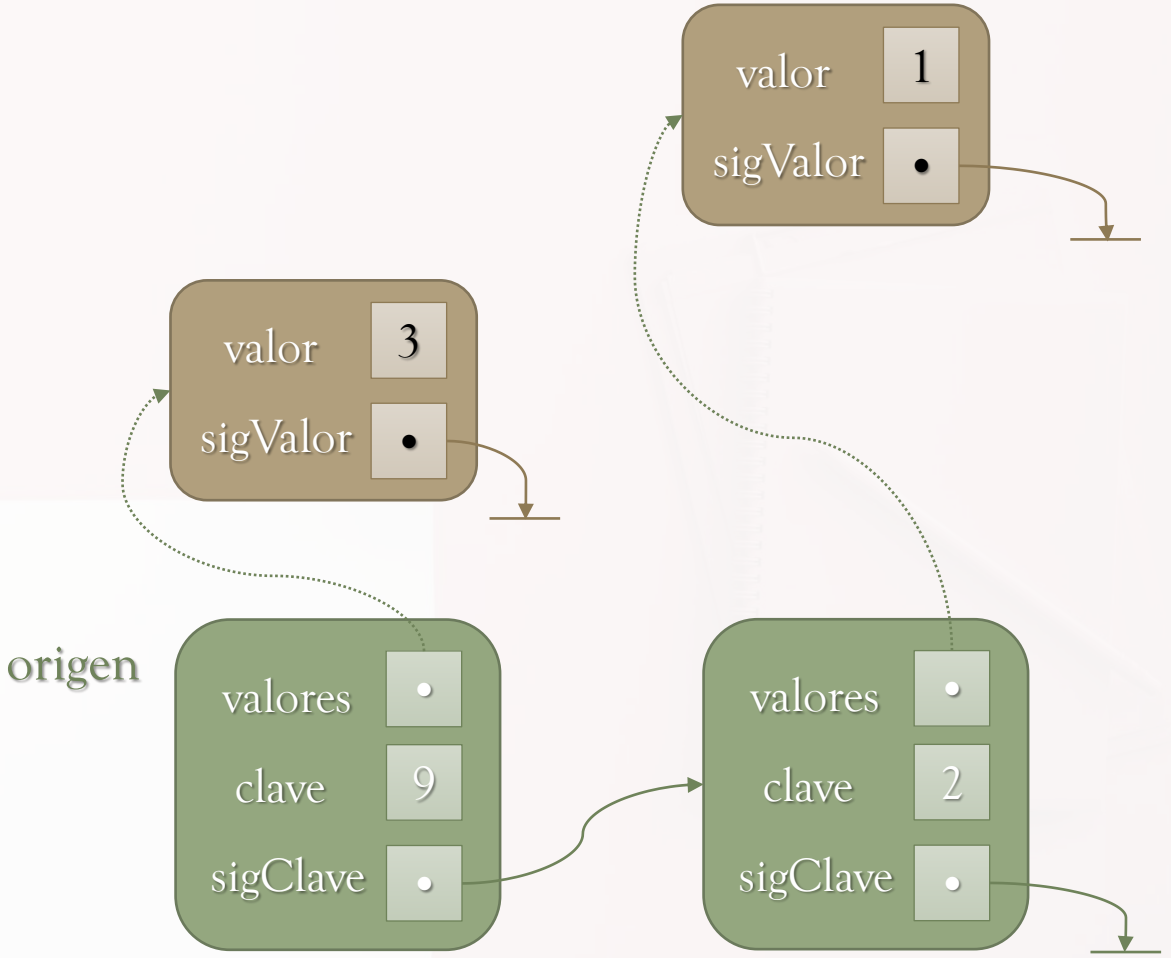




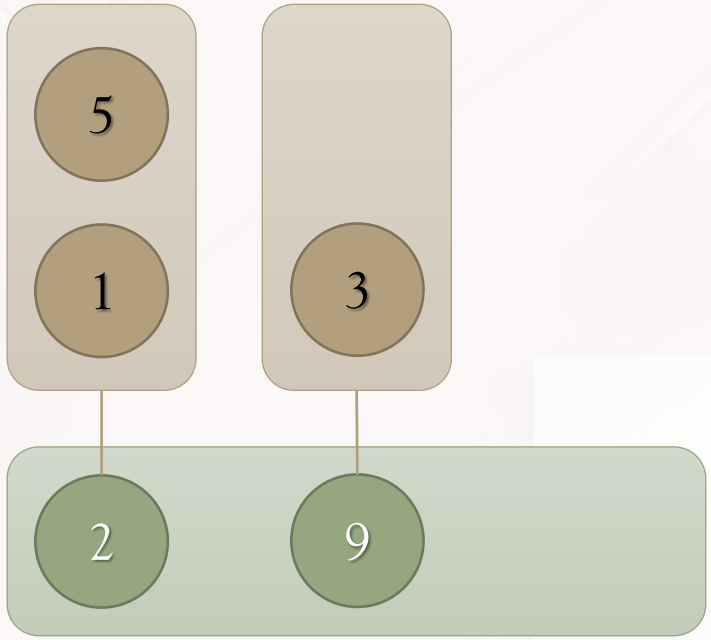
Diccionario múltiple - implementación dinámica



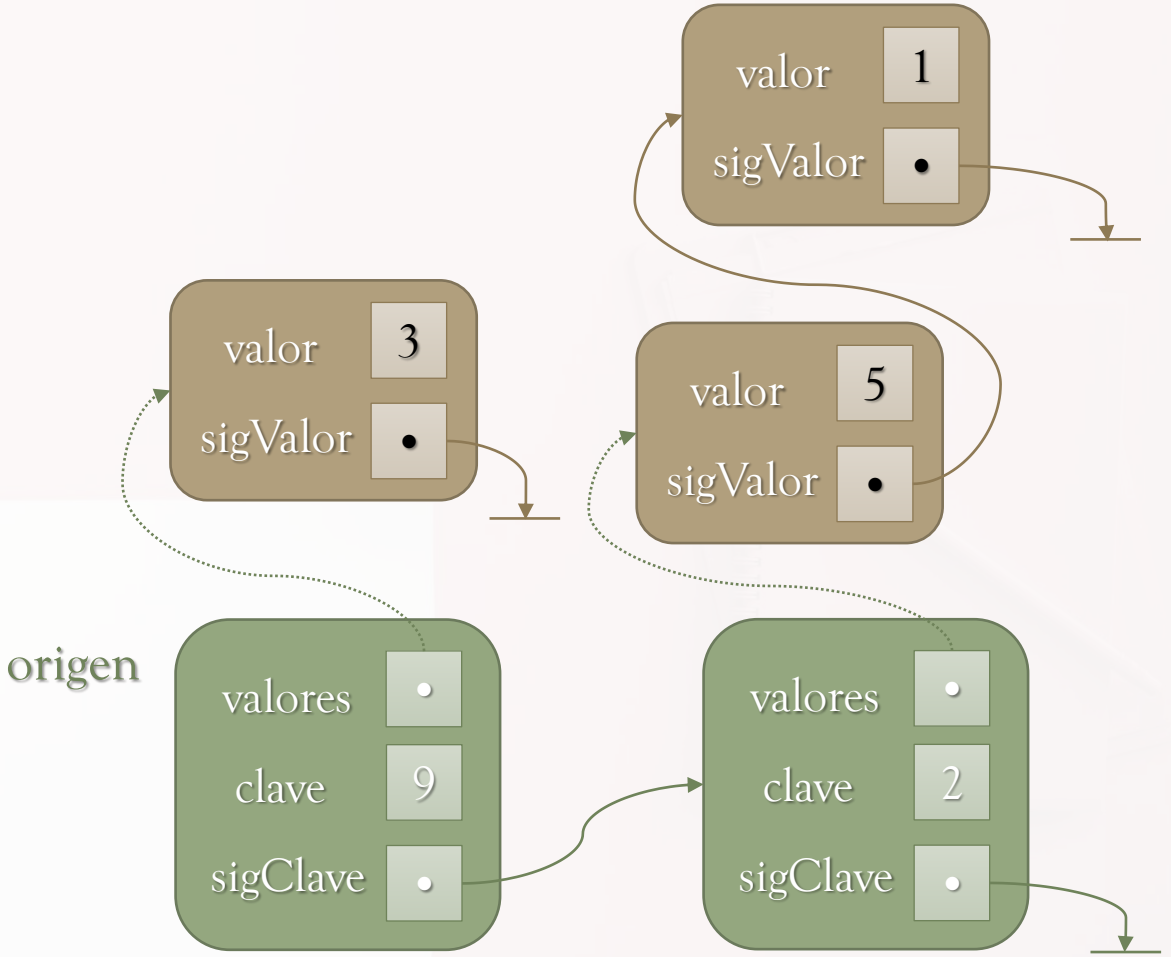
agregar(2, 5)



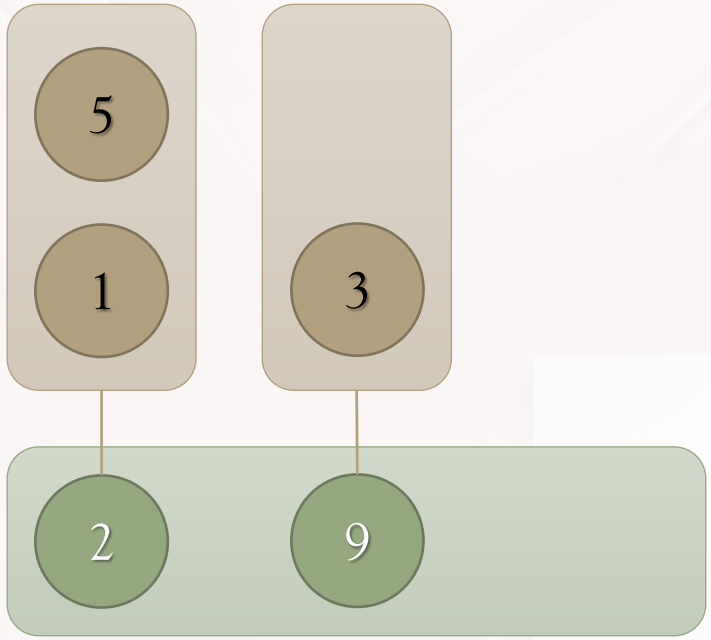
Diccionario múltiple - implementación dinámica



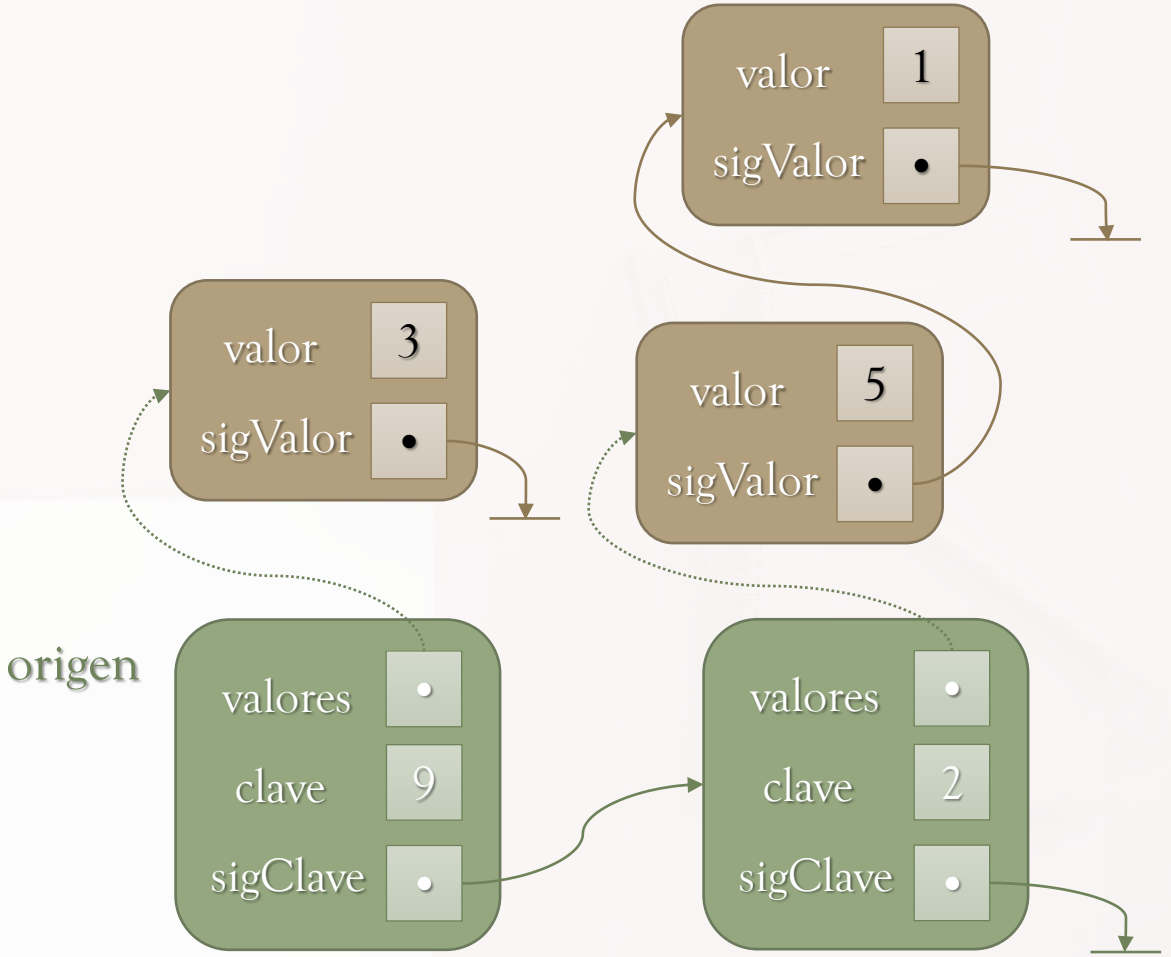
agregar(2, 5)



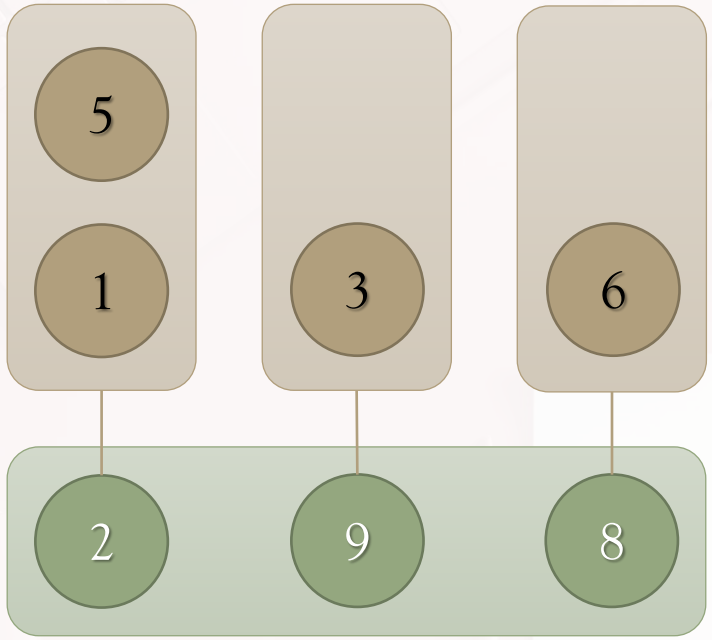
Diccionario múltiple - implementación dinámica

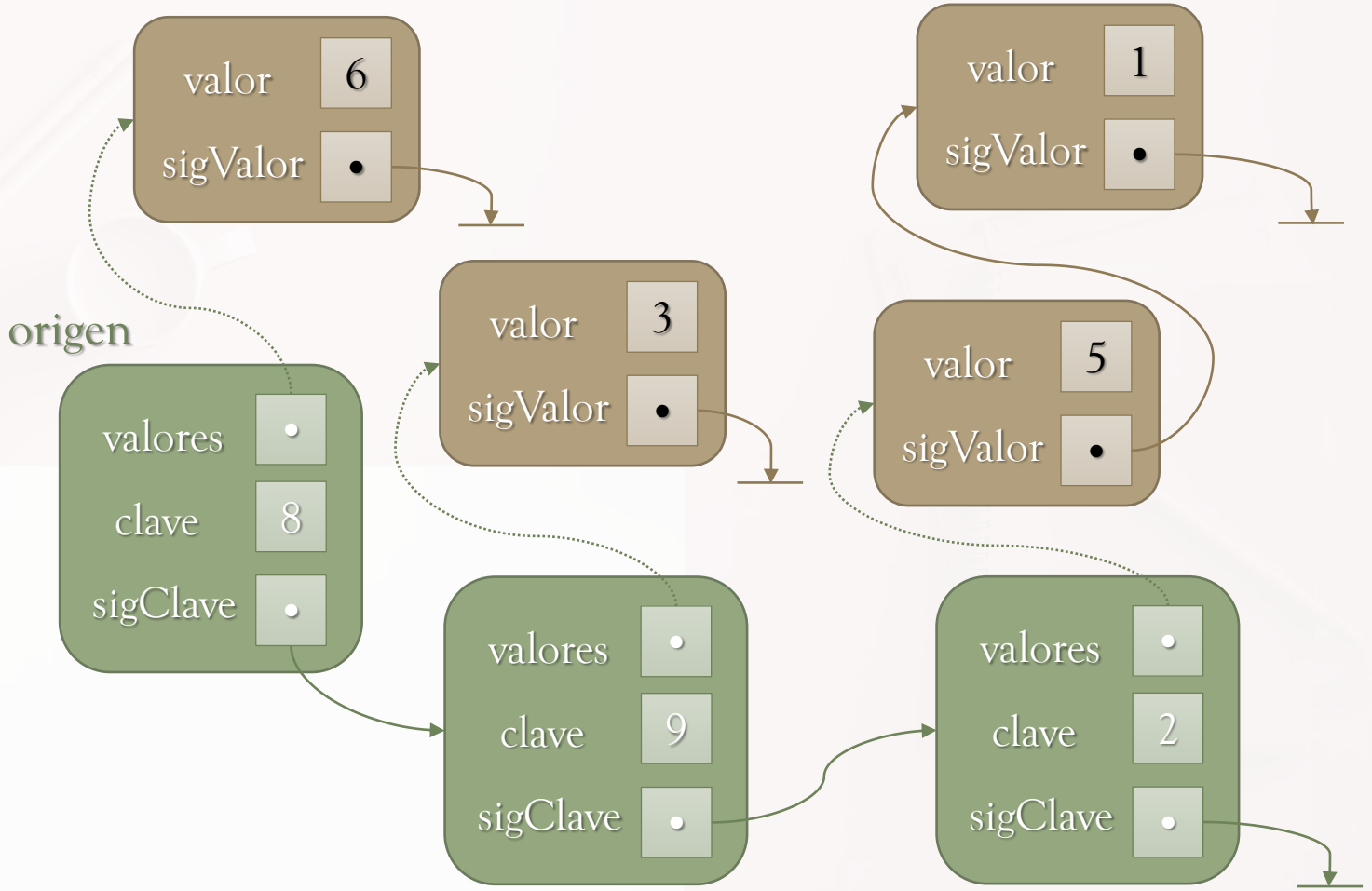


agregar(8, 6)



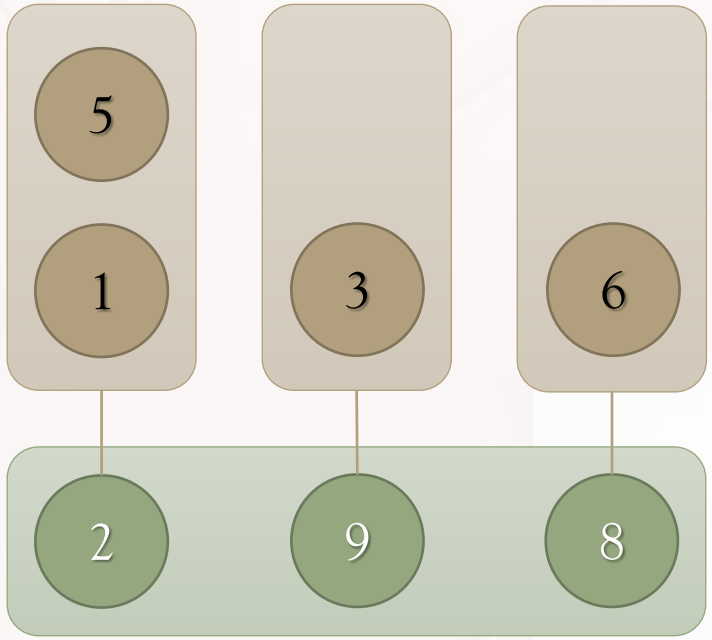
Diccionario múltiple - implementación dinámica

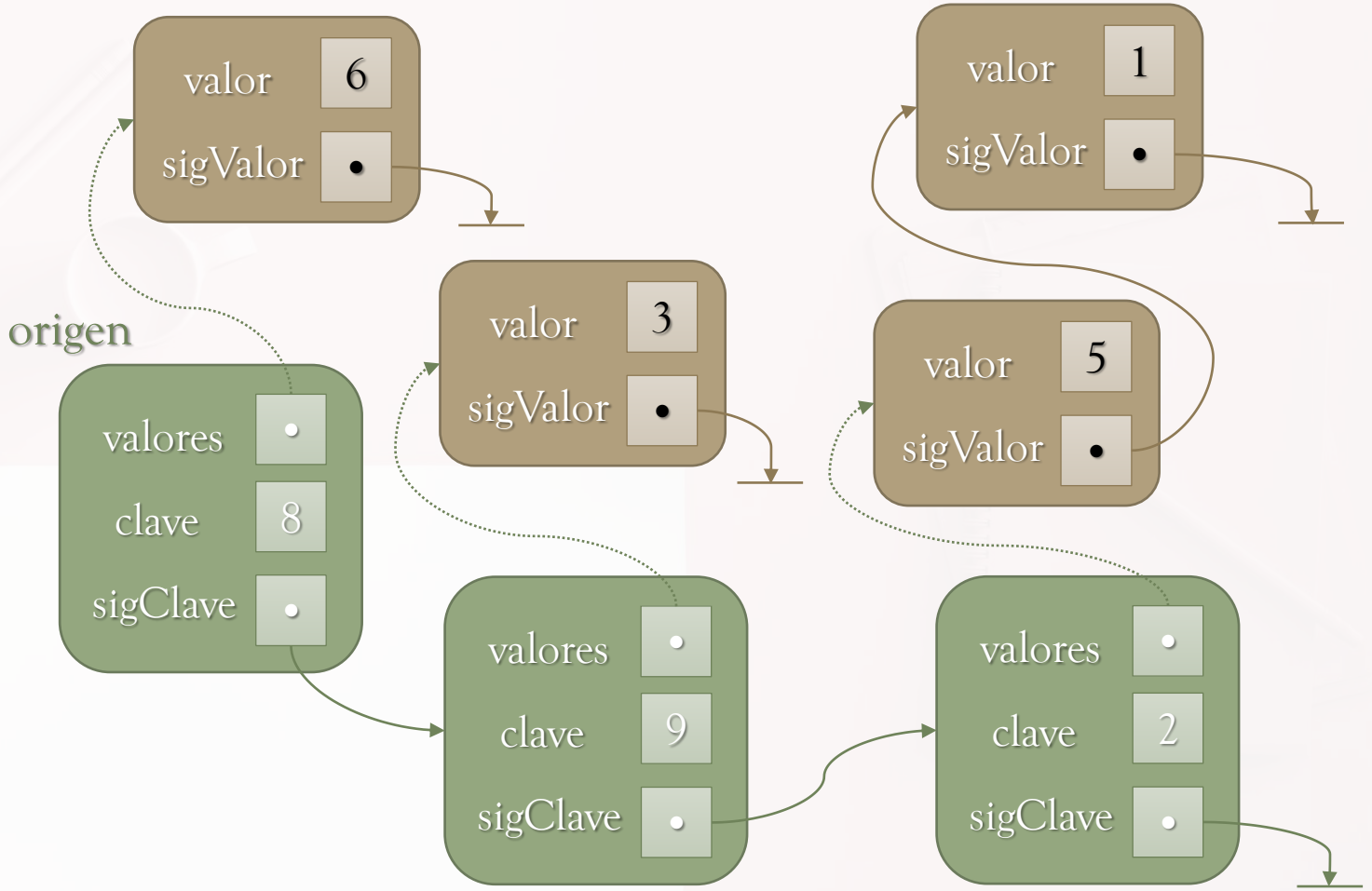




agregar(8, 6)

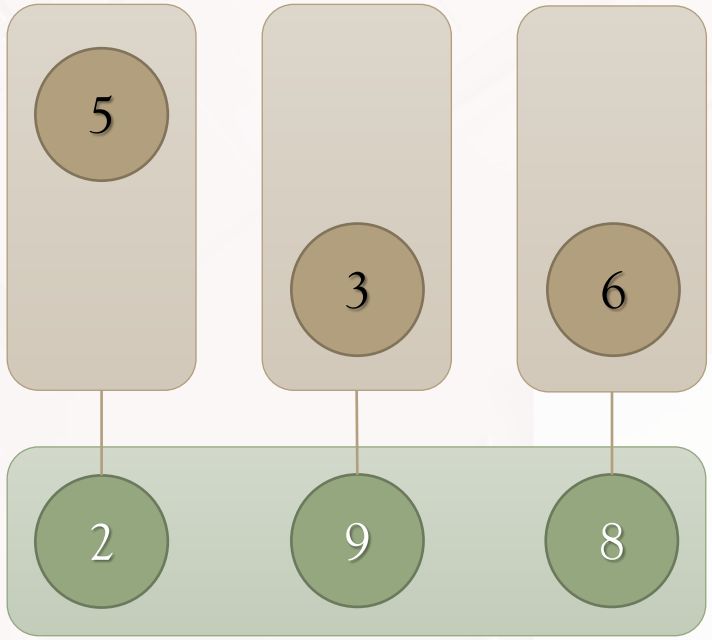
Diccionario múltiple - implementación dinámica

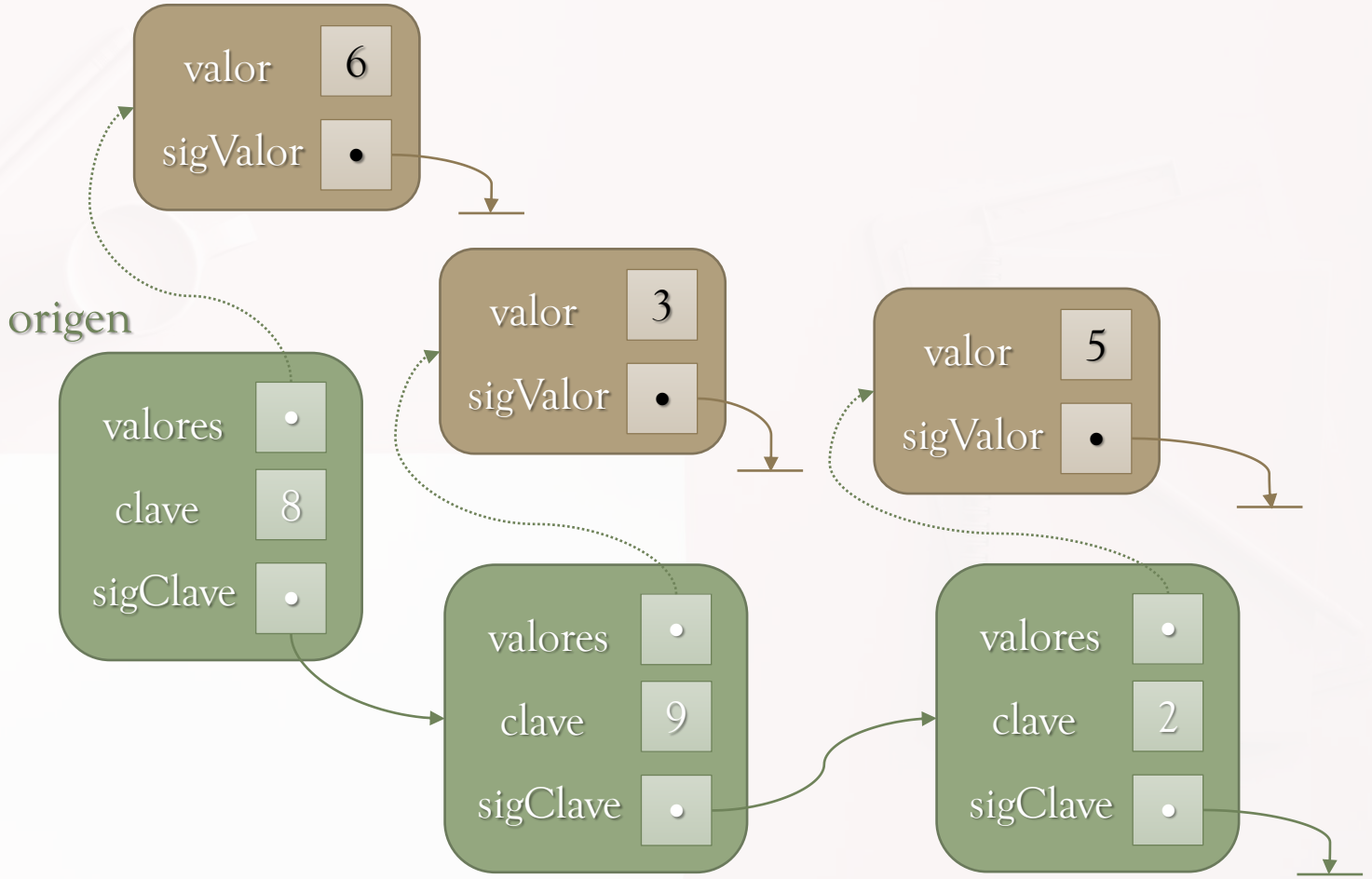




eliminarValor(2, 1)

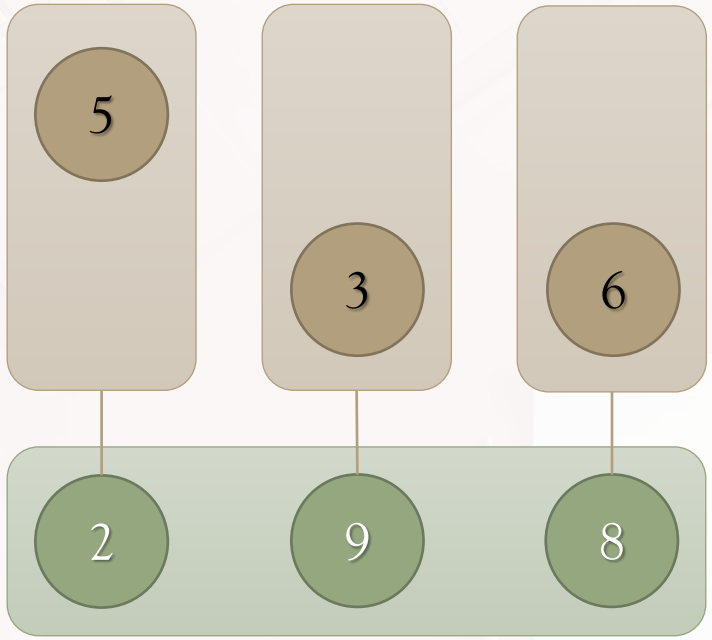
Diccionario múltiple - implementación dinámica

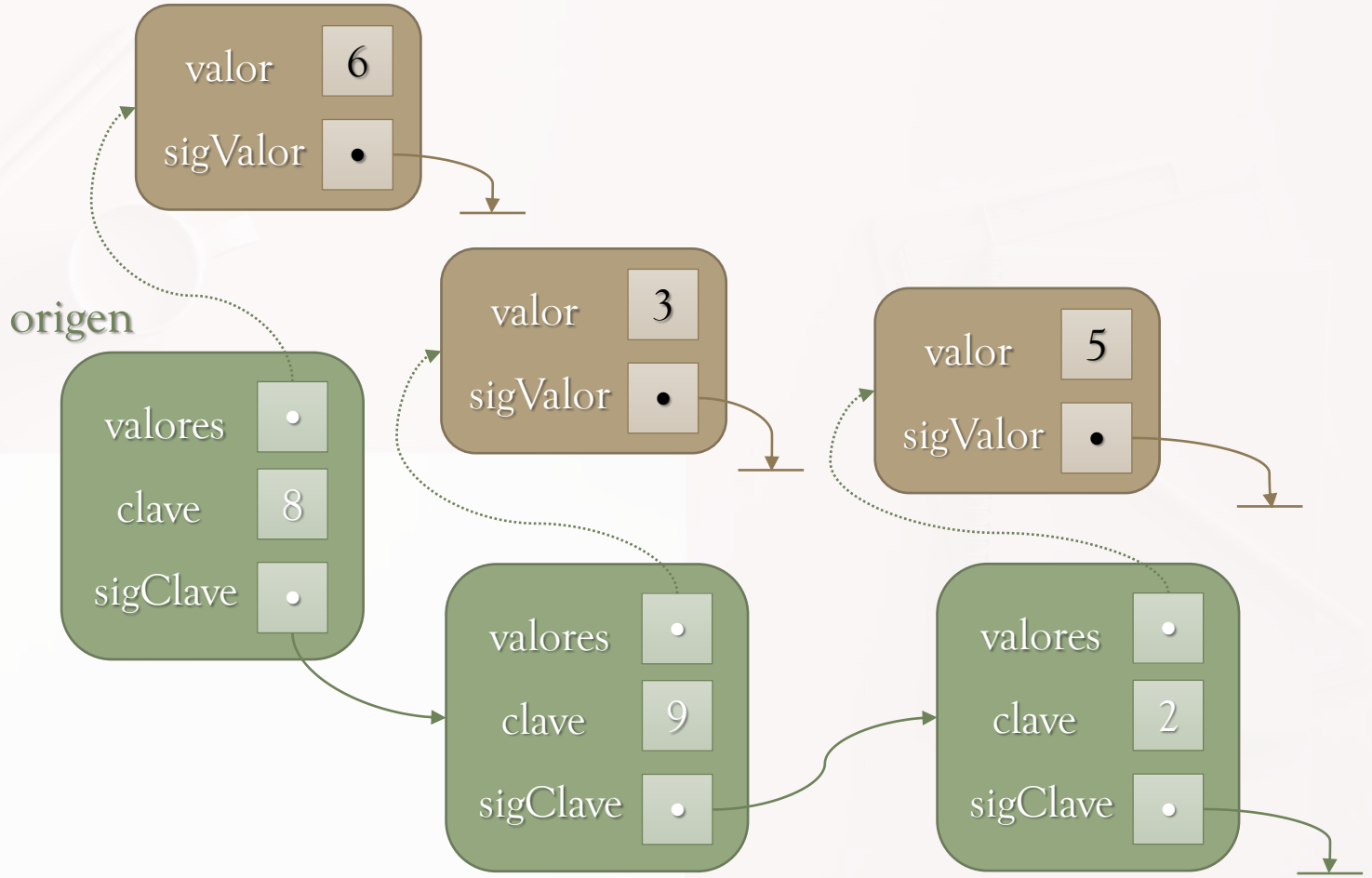




eliminarValor(2, 1)

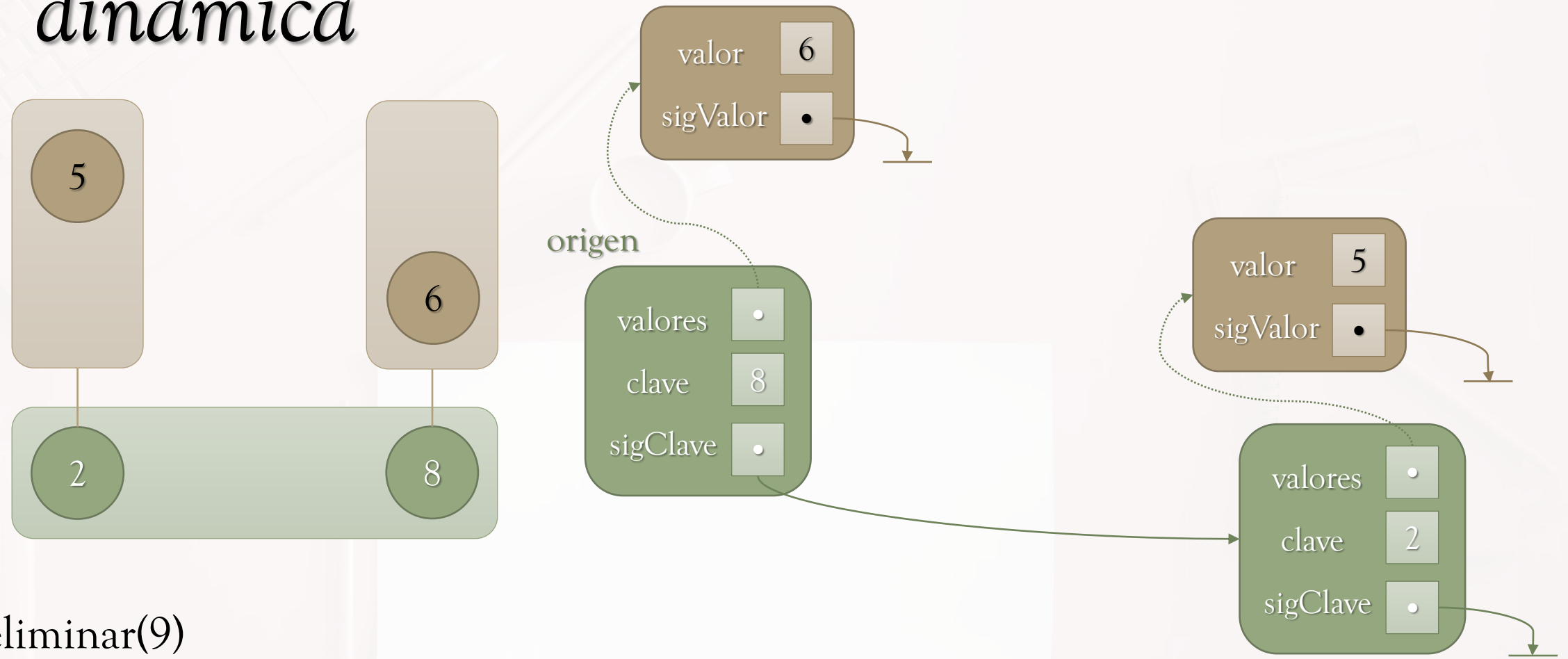
Diccionario múltiple - implementación dinámica





eliminar(9)

Diccionario múltiple - implementación dinámica



eliminar(9)


¡Muchas Gracias!

Bibliografía

- *Programación II – Apuntes de Cátedra – V1.3 – Cuadrado Trutner – UADE*
- *Programación II – Apuntes de Cátedra – Wehbe – UADE*