Algoritmos y estructuras de datos

Listas doblemente enlazadas

Francisco Javier Zaragoza Martínez

Universidad Autónoma Metropolitana Unidad Azcapotzalco
Departamento de Sistemas

Universidad Autónoma Metropolitana
Casa abierta al tiempo Azcapotzalco

Posgrado en Optimización

14 de mayo de 2021

Citas

Will Advise

Si tienes la mitad de nada, véndela por el doble de algo, revende la mitad al doble del precio y compra otro algo y medio. ¿Cuánta nada tendrás dentro de dos días? Como tres.

George Orwell

El doble pensar es el poder de mantener al mismo tiempo en la mente dos creencias contradictorias y aceptarlas ambas.

Thomas Hood

Un doble significado muestra un doble sentido; y si los proverbios dicen la verdad, un doble diente es la residencia adoptada por la sabiduría.

Lista doblemente enlazada

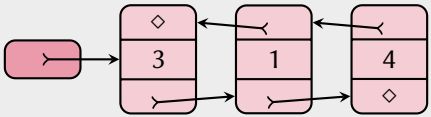
Desventaja de las listas enlazadas

En un arreglo es posible avanzar de una posición a la siguiente y a la anterior en una unidad de tiempo. En una lista enlazada se puede avanzar a la siguiente, pero para moverse a la anterior se requiere recorrer la lista, lo cual es demasiado lento.

Lista doblemente enlazada

Una lista doblemente enlazada puede estar vacía o consistir de una secuencia de nodos, donde cada nodo contiene un dato y sabe dónde están tanto el siguiente como el anterior nodo o, alternativamente, que no hay siguiente o anterior.

Ejemplo



Definiremos un tipo estructurado nodod para representar un nodo doble y un tipo listad para representar una lista doblemente enlazada. El tipo nodod consiste de un dato y dos apuntadores.

```
typedef struct nodod {
  int a;              // dato almacenado
  struct nodod *sig; // enlace al siguiente
  struct nodod *ant; // enlace al anterior
} nodod;
```

Por otro lado, el tipo listad es un apuntador a nodod.

```
typedef nodod *listad;
```

Note que los tipos nodod *, struct nodod * y listad son equivalentes.

Esta función pide la memoria para un nodo doble y llena sus campos.

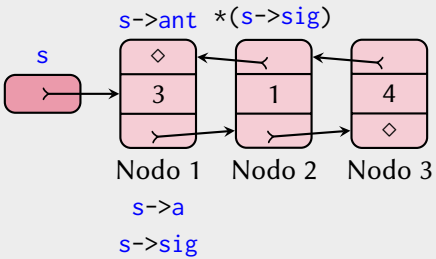
```
nodod *creaNodod(int x, nodod *p, nodod *q) {
  nodod *t = (nodod *) malloc(sizeof(nodod));
  t->a = x;
  t->sig = p;
  t->ant = q;
  return t;
}
```

Lista doblemente enlazada

De regreso al ejemplo

Declaramos una lista doblemente enlazada vacía `s` con `listad s = NULL;`

La siguiente es una lista doblemente enlazada que ya tiene tres nodos:



Complicación

La experiencia con colas en listas enlazadas nos dice que nos esperan varios casos especiales para insertar el primer nodo, borrar el único nodo, agregar un nodo antes del primero o después del último.

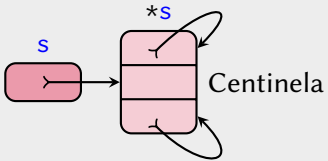
Solución

Una solución es la prevención. En lugar de que una lista doblemente enlazada vacía `s` conste sólo del apuntador nulo, haremos que conste también de un nodo especial `*s` al que llamaremos centinela. El dato `s->a` en este nodo será ignorado, mientras que sus dos enlaces `s->sig` y `s->ant` apuntarán al mismo centinela. El plan es que cada nodo tenga siguiente y anterior. Esto es una lista circular doblemente enlazada.

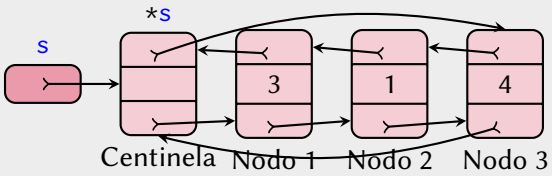
Lista doblemente enlazada

Otra vez de regreso al ejemplo

Una lista circular doblemente enlazada vacía:



Una lista circular doblemente enlazada que ya tiene tres nodos:



Crear una lista circular doblemente enlazada vacía

```
void creaListaC(listad *s) {
  *s = creaNodod(0, NULL, NULL);
  (*s)->sig = (*s)->ant = *s;
}
```

Saber si una lista circular doblemente enlazada está vacía

```
int esVaciaListaC(listad s) {
  return (s == s->sig);
}
```

Dos tipos de inserción

Si tenemos una referencia s a un nodo de una lista doblemente enlazada, estaremos interesados en dos tipos de inserción.

1. Insertar un nodo nuevo como siguiente a *s.
2. Insertar un nodo nuevo como anterior a *s.

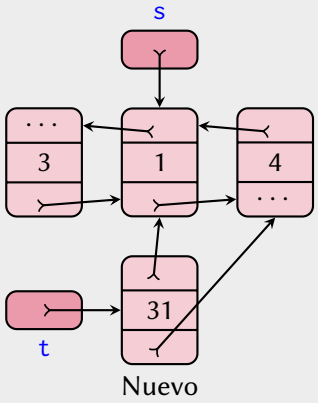
Efecto del nodo centinela

Aunque los enlaces desde el nodo centinela pueden cambiar, el nodo centinela *s y el apuntador s a él siempre son los mismos. Esto se traducirá en que no necesitamos pasar la lista s por referencia.

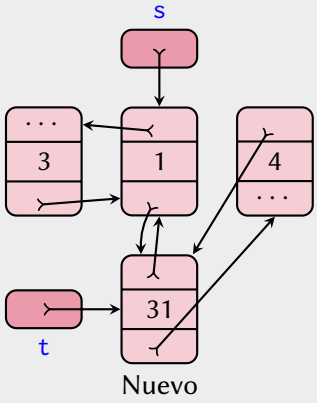
Lista doblemente enlazada

Ejemplo de inserción después de un nodo

Justo después de pedir el nodo



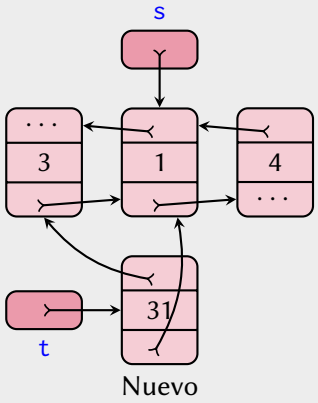
Justo antes de terminar



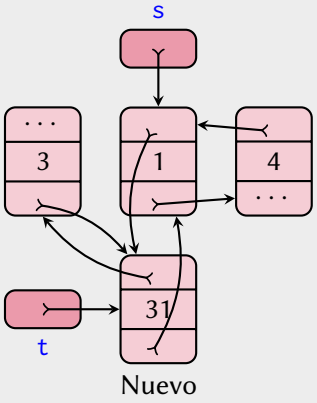
Lista doblemente enlazada

Ejemplo de inserción antes de un nodo

Justo después de pedir el nodo



Justo antes de terminar



Lista doblemente enlazada

Dos tipos de inserción

Insertar después

```
void despuesLC(nodod *s, int x) {
  nodod *t = creaNodod(x, s->sig, s);
  t->sig->ant = t->ant->sig = t;
}
```

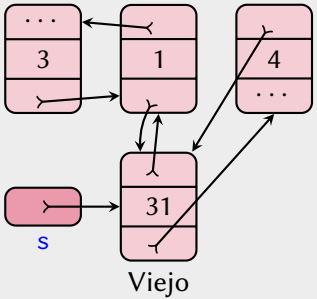
Insertar antes

```
void antesLC(nodod *s, int x) {
  nodod *t = creaNodod(x, s, s->ant);
  t->sig->ant = t->ant->sig = t;
}
```

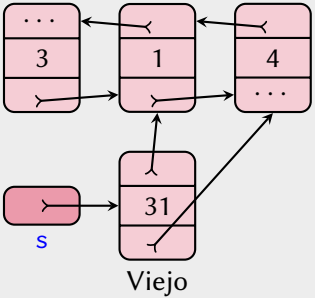
Lista doblemente enlazada

Ejemplo de eliminación de un nodo

Justo después de copiar el dato



Justo antes de liberar el nodo



Lista doblemente enlazada

Eliminación de un nodo

```
void eliminaLC(nodod *s, int *x) {
  *x = s->a;              // copiamos dato
  s->sig->ant = s->ant; // nuevo anterior
  s->ant->sig = s->sig; // nuevo siguiente
  free(s);                // liberamos nodo
}
```

Lista doblemente enlazada

Destrucción de una lista circular doblemente enlazada

```
void destruyeLC(listad *s) {
  int x;
  while (!esVaciaLC(*s))
    eliminaLC(s->sig, &x);
  free(*s);
  *s = NULL;
}
```

Cola doble en una lista circular doblemente enlazada

Si tenemos una lista circular doblemente enlazada `s` con centinela `*s`, entonces las cuatro operaciones de las colas dobles se pueden llevar a cabo de la siguiente forma:

1. adelante(s, x) se hace `despuesLC(s, x)`
2. atrás(s, x) se hace `antesLC(s, x)`
3. delante(s) se hace `eliminaLC(s->sig, &x)`
4. detrás(s) se hace `eliminaLC(s->ant, &x)`

Observa que el enlace `s->sig` del centinela juega el papel del lado ante y el enlance `s->ant` del centinela juega el papel del lado tras.