Fundamentos de análisis y diseño de algoritmos

Estructuras de datos

Universidad del Valle
Facultad de Ingeniería
Escuela de Ingeniería de sistemas y computación
Septiembre 2017

Pilas

Colas

Listas enlazadas

Listas doblemente enlazadas

Árboles

Estructuras de datos

Pila

Una pila es una estructura de datos tipo LIFO (Last In First Out), por lo que el último elemento insertado será el primero en ser borrado

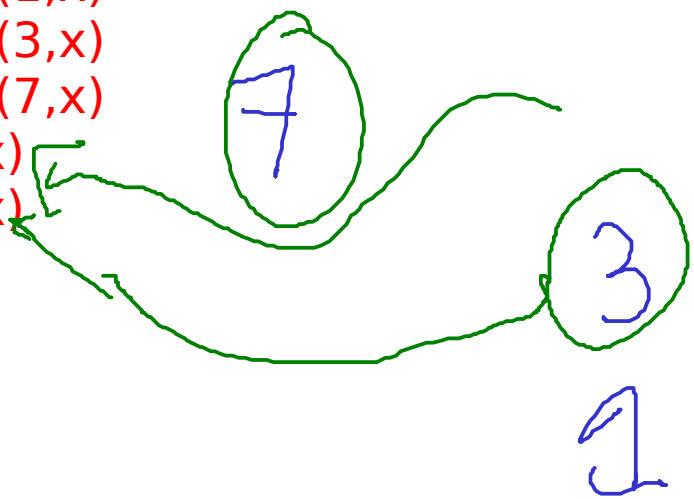
Operaciones básicas:

STACK-EMPTY(S)

PUSH(S,x)

POP(S)

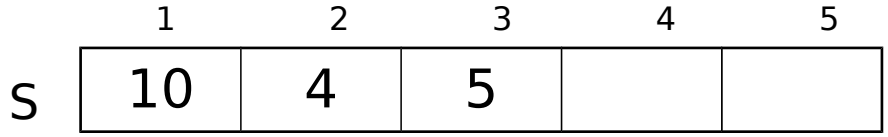
PUSH(1,x)
PUSH(3,x)
PUSH(7,x)
POP(x)
POP(x)



Estructuras de datos

Pila

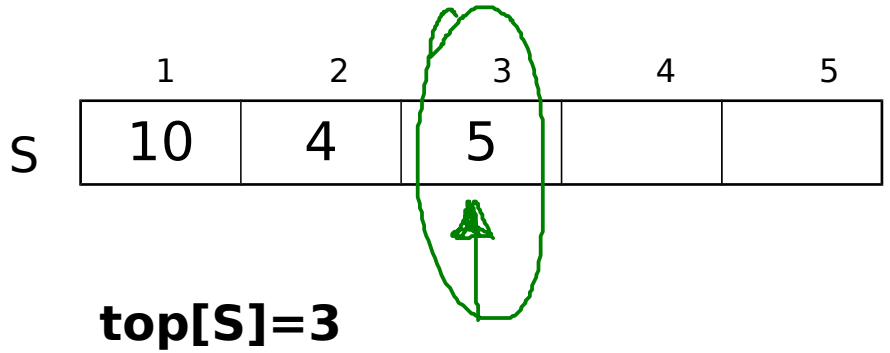
Una forma de implementar la pila es por medio de un arreglo unidimensional



Esto supone varios aspectos:

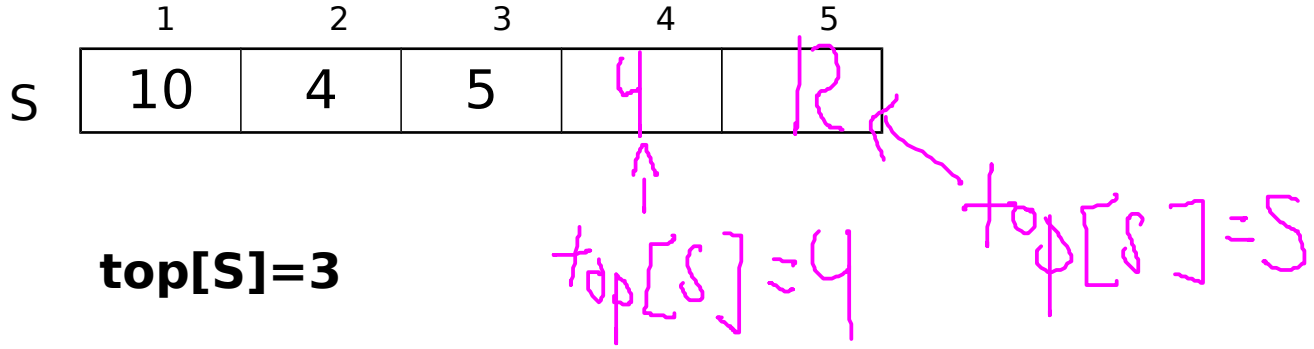
- La pila tiene una capacidad limitada
- Se cuenta con un atributo adicional, llamado top[S], que almacena el índice en el arreglo que guarda el último valor, esto es, el tope de la pila
- Cuando la pila esté vacía, top[S]=0

Estructuras de datos



top[S]=3

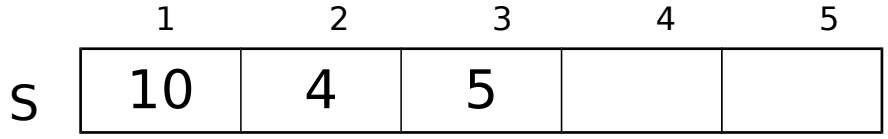
Estructuras de datos



Indique lo que sucede después de cada instrucción, siendo S la pila que se muestre arriba:

STACK-EMPTY(S)
PUSH(S,4)
PUSH(S,12)
PUSH(S,7)

Estructuras de datos

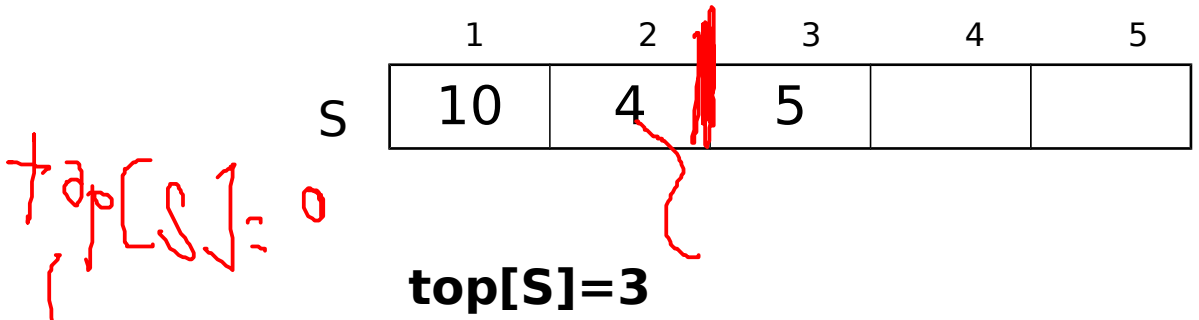


top[S]=3

Indique lo que sucede después de cada instrucción, siendo S la pila que se muestre arriba:

STACK-EMPTY(S)
PUSH(S,4)
PUSH(S,12)
PUSH(S,7) ← Overflow - desbordamiento en su capacidad máxima

Estructuras de datos



Indique lo que sucede después de cada instrucción, siendo S la pila que se muestre arriba:

STACK-EMPTY(S)
POP(S)
POP(S)
POP(S)
STACK-EMPTY(S)
POP(S)

Underflow

Estructuras de datos

Indique un algoritmo para cada operación básica, acompañado de su respectiva complejidad usando la notación O

- STACK-EMPTY(S), retorna true o false
- PUSH(S,x), adiciona x al tope, no devuelve ningún valor
- POP(S), borra el elemento que esté en el tope y devuelve ese valor

Estructuras de datos

STACK-EMPTY(S)

```
1   if (top[S]==0)
2    then return true
3    else return false
```

Estructuras de datos

STACK-EMPTY(S)

```
1   if (top[S]==0)
2    then return true
3    else return false
```

T(n)=O(1), tiempo constante

Estructuras de datos

Cola

Una cola es una estructura de datos tipo FIFO (First In First Out), por lo que el primer elemento que es insertado, es el primero en ser borrado

Operaciones básicas:

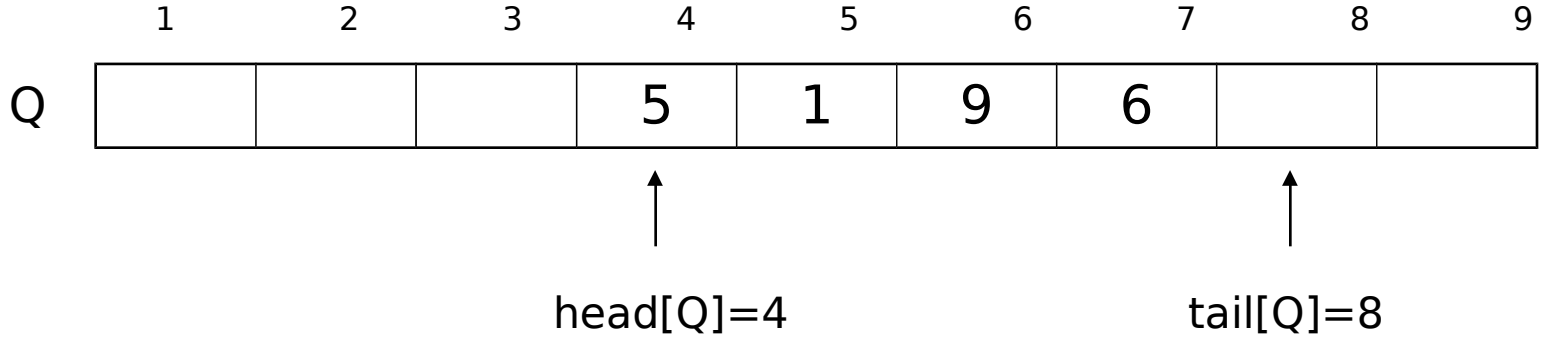
ENQUEUE(Q,x)

DEQUEUE(Q)

Estructuras de datos

Cola

Una forma de implementar la cola es por medio de un arreglo unidimensional



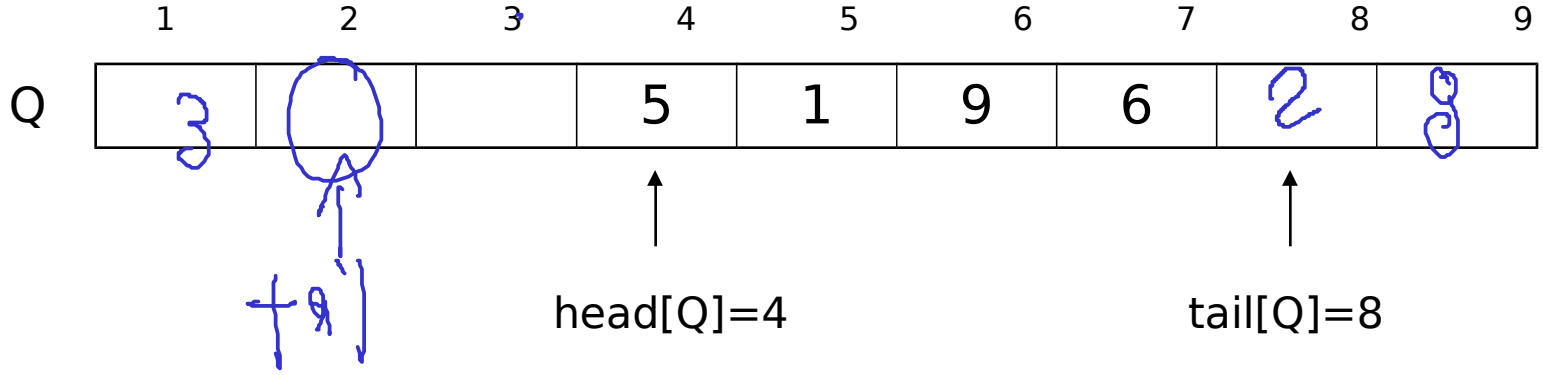
Esto supone varios aspectos:

- La cola tiene una capacidad limitada
- Se cuenta con dos atributos adicionales, head[Q] que guarda el indice de la cabeza y tail[Q] que apunta al siguiente lugar en el cual será insertado un elemento

Estructuras de datos

Cola

Una forma de implementar la cola es por medio de un arreglo unidimensional



ENQUEUE(Q,2)

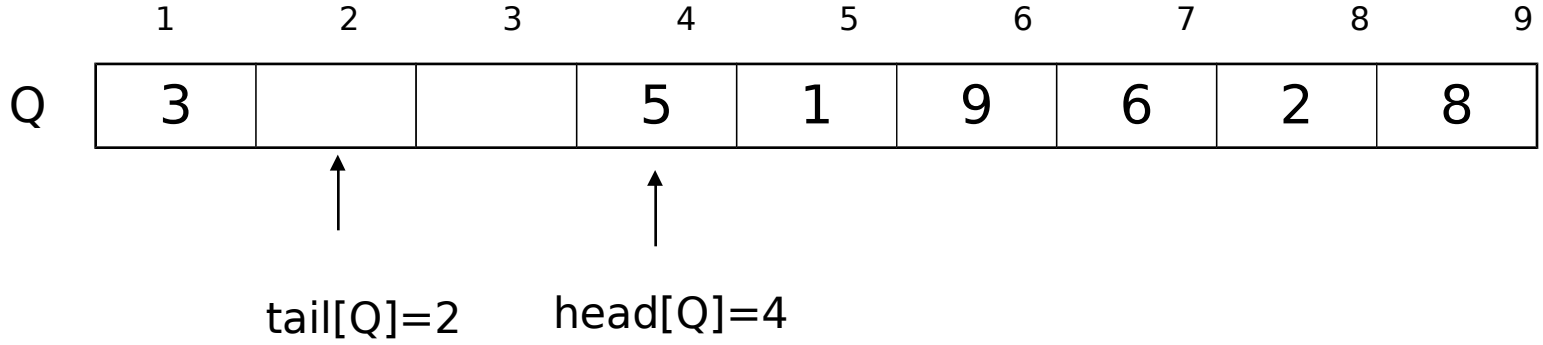
ENQUEUE(Q,8)

ENQUEUE(Q,3)

Estructuras de datos

Cola

Una forma de implementar la cola es por medio de un arreglo unidimensional



ENQUEUE(Q,2)

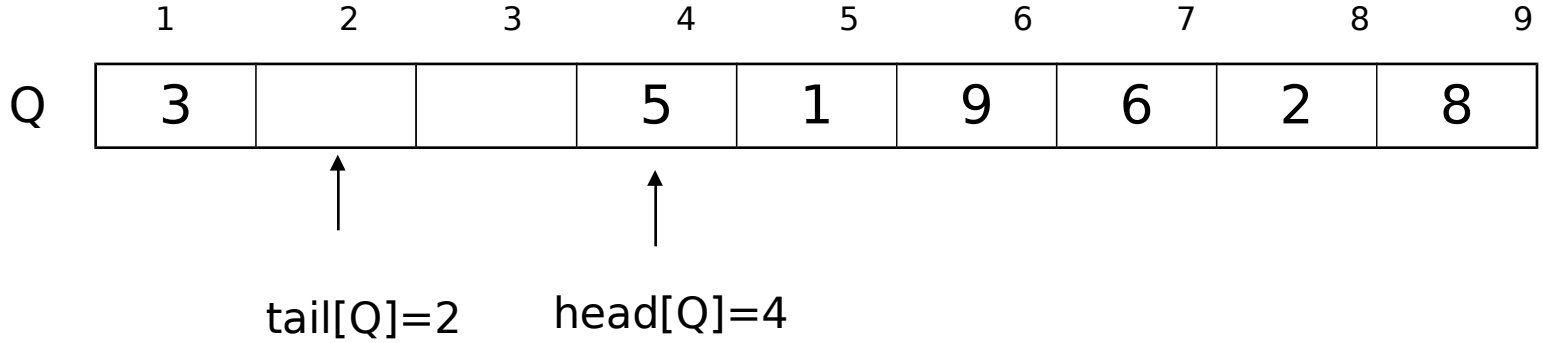
ENQUEUE(Q,8) ← Si se llega al final del arreglo, se intenta insertar en la posición 1

ENQUEUE(Q,3)

Estructuras de datos

Cola

Una forma de implementar la cola es por medio de un arreglo unidimensional

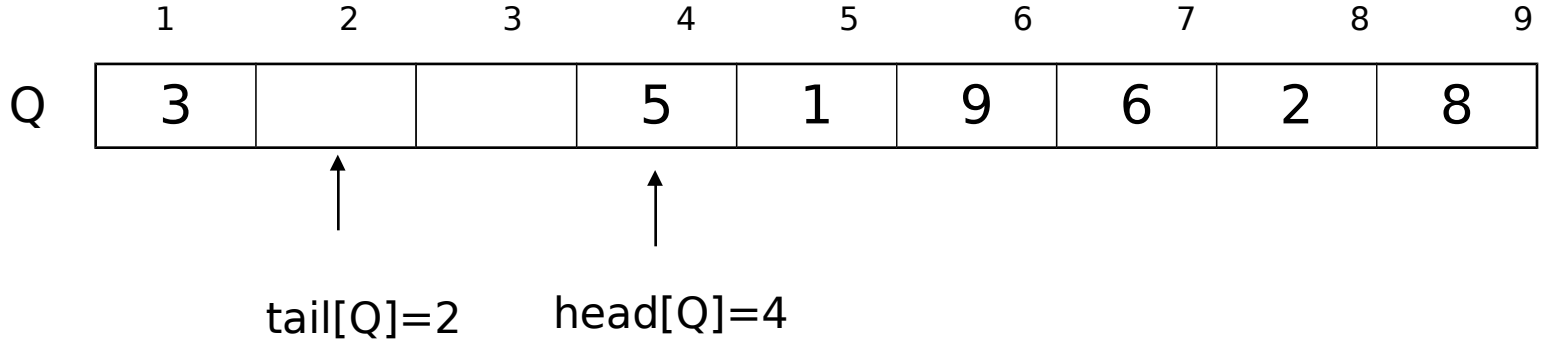


Cómo sabe que la cola está llena?

Estructuras de datos

Cola

Una forma de implementar la cola es por medio de un arreglo unidimensional



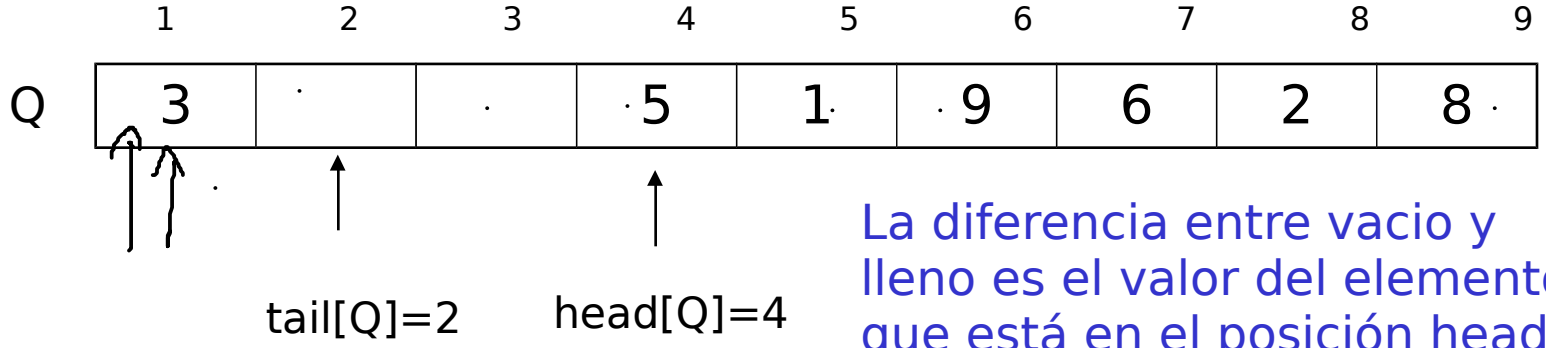
Cómo sabe que la cola está llena?

tail[Q]=head[Q]

Estructuras de datos

Cola

Una forma de implementar la cola es por medio de un arreglo unidimensional



Inicialmente tail[Q]=head[Q]=1

La diferencia entre vacio y lleno es el valor del elemento que está en el posición head

Cuando la cola se vacía debe colocarse un valor que indique vacio.

Tambien puede manejarse una variable que indique si esta vacio o no. En el procedimiento de encolar

Estructuras de datos

Indique un algoritmo para cada operación básica, acompañado de su respectiva complejidad usando la notación O

- ENQUEUE(Q,x)
- DEQUEUE(Q)

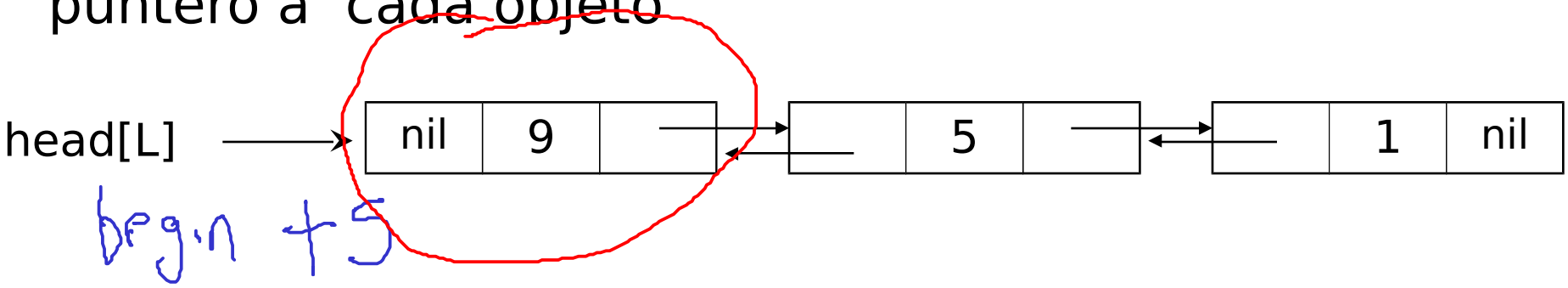
1) Ingresa en la posición tail el elemento
2) tail+=1 (Examinando no desbordarse)
3) Si head == tail y vacia = false, entonces debo retornar un mensaje de "Overflow"

DEQUEUE
1) head+=1 (Examinando no desbordarse)
2) Condiciones para vacio (Sea con valor nulo o con una variable para indicar que esta vacia)
3) Condicion si esta vacia, cambiar el valor de la variable.
4) Si head==tail y vacia = true, entonces retornar "Underflow"

Estructuras de datos

Listas doblemente enlazadas

Es una estructura de datos en la cual los objetos son organizados en un orden lineal. A diferencia de los arreglos, el orden en las listas está dado por un puntero a cada objeto

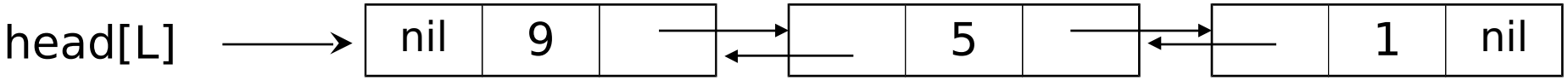


- Cada nodo en una lista doblemente enlazada tiene 3 campos: prev, key y next
- Se tiene además un puntero al primer nodo

Estructuras de datos

Listas doblemente enlazadas

Es una estructura de datos en la cual los objetos son organizados en un orden lineal. A diferencia de los arreglos, el orden en las listas está dado por un puntero a cada objeto



Operaciones

- LIST-INSERT(L,x): inserta x en la cabeza de la lista. x es un nodo tal que key[x]=k, y prev=next=nil
- LIST-DELETE(L,x): donde x es el nodo que se desea borrar
- LIST-SEARCH(L,k): busca el primer nodo que tiene llave k y retorna un puntero a ese nodo

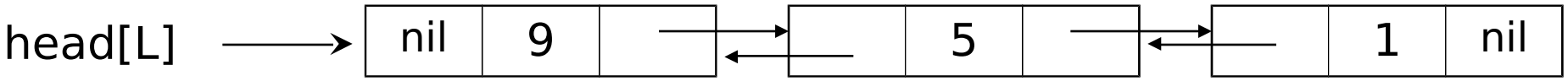
Estructuras de datos

LIST-SEARCH busca el primer nodo que tiene llave k y retorna un puntero a ese nodo

LIST-SEARCH(L,k)

1. x←head[L]
2. while x≠nil and key[x]≠k
3. x←next[x]
4. return x

¿Cuál es la complejidad en el peor caso?



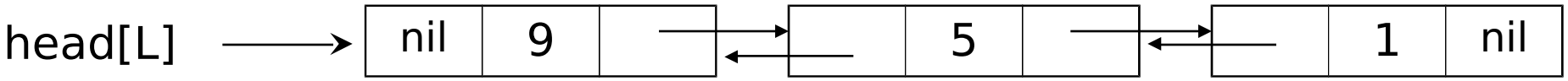
Estructuras de datos

LIST-SEARCH busca el primer nodo que tiene llave k y retorna un puntero a ese nodo

LIST-SEARCH(L,k)

1. x←head[L]
2. while x≠nil and key[x]≠k
3. x←next[x]
4. return x

En el peor caso será O(n)



Estructuras de datos

Indique el resultado de realizar las siguientes operaciones:

prev[z]=nil

next[z]=nil

key[z]=10

LIST-INSERT(L,z)

prev[w]=nil

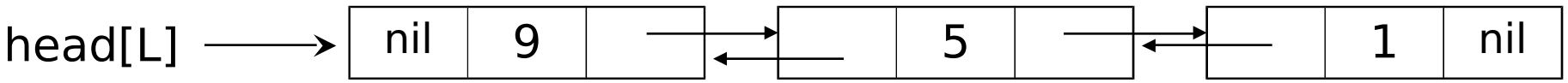
next[w]=nil

key[w]=8

LIST-INSERT(L,w)

x=LIST-SEARCH(L,10)

LIST-DELETE(L,x)



Estructuras de datos

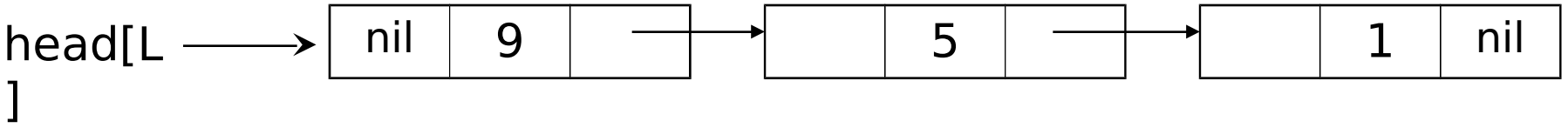
Indique el algoritmo para las siguiente operaciones y muestre su complejidad en el peor caso:

- LIST-INSERT(L,x): inserta x en la cabeza de la lista. x es un nodo tal que key[x]=k, y prev=next=nil

- LIST-DELETE(L,x): donde x es el nodo que se desea borrar

Estructuras de datos

Listas simplemente enlazada



Operaciones

- LIST-INSERT(L,x): inserta x al final de la lista. x es un nodo tal que key[x]=k, y prev=next=nil
- LIST-DELETE(L): donde x es el nodo al final de la lista
- LIST-SEARCH(L,k): busca el primer nodo que tiene llave k y retorna un puntero a ese nodo

Referencias

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. Introduction to Algorithms, Third Edition (3rd ed.). The MIT Press. Chapter 10

Gracias

Próximo tema:

Estructuras de datos: Tablas Hash