

Fundamentos de análisis y diseño de algoritmos

Estructuras de datos

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)

Estructuras de datos

Pila

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

	1	2	3	4	5
S	10	4	5		

Esto supone varios aspectos:

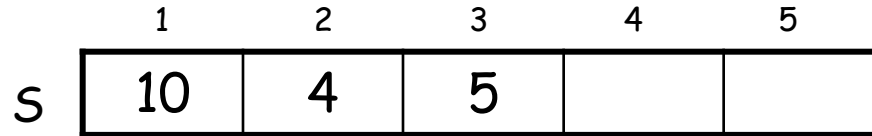
- La pila tiene una capacidad limitada
- Se cuenta con un atributo adicional, llamado $\text{top}[S]$, que almacena el índice en el arreglo que guarda el último valor, esto es, el tope de la pila
- Cuando la pila esté vacía, $\text{top}[S]=0$

Estructuras de datos

	1	2	3	4	5
S	10	4	5		

$\text{top}[S]=3$

Estructuras de datos



top[S]=3

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

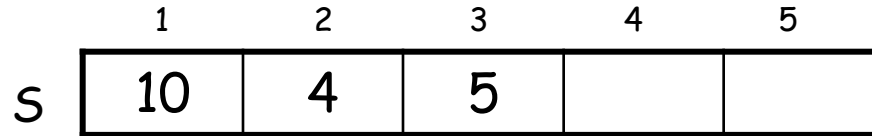
STACK-EMPTY(*S*)

PUSH(*S*,4)

PUSH(*S*,12)

PUSH(*S*,7)

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)

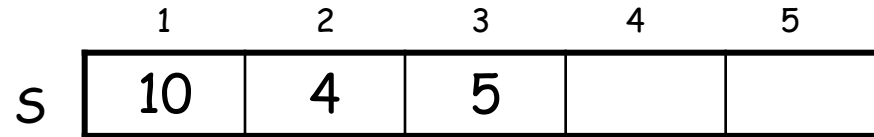
PUSH(S,12)

PUSH(S,7)



Overflow - desbordamiento en su capacidad máxima

Estructuras de datos



$\text{top}[S]=3$

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

STACK-EMPTY(S)

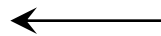
POP(S)

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:

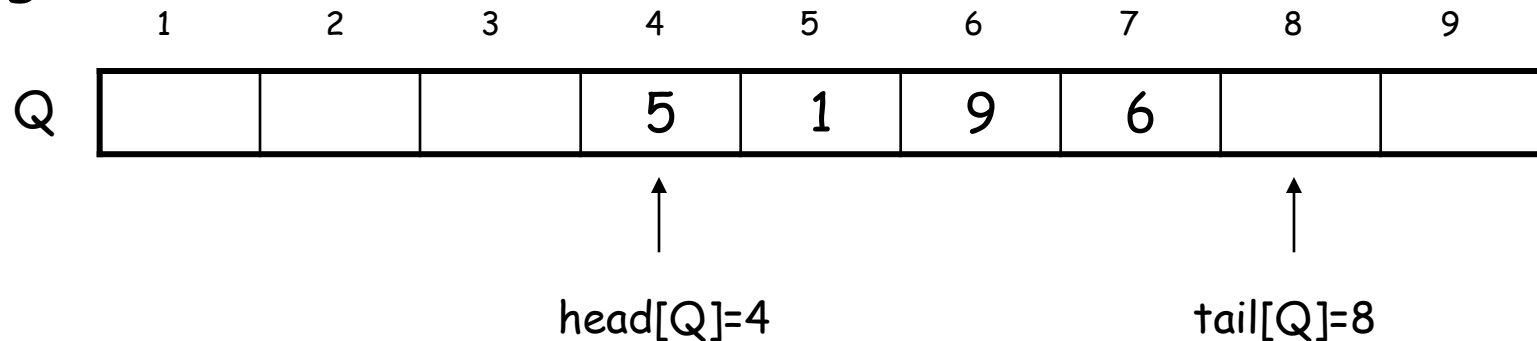
ENQUEUE(Q,x)

DEQUEUE(Q)

Estructuras de datos

Cola

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



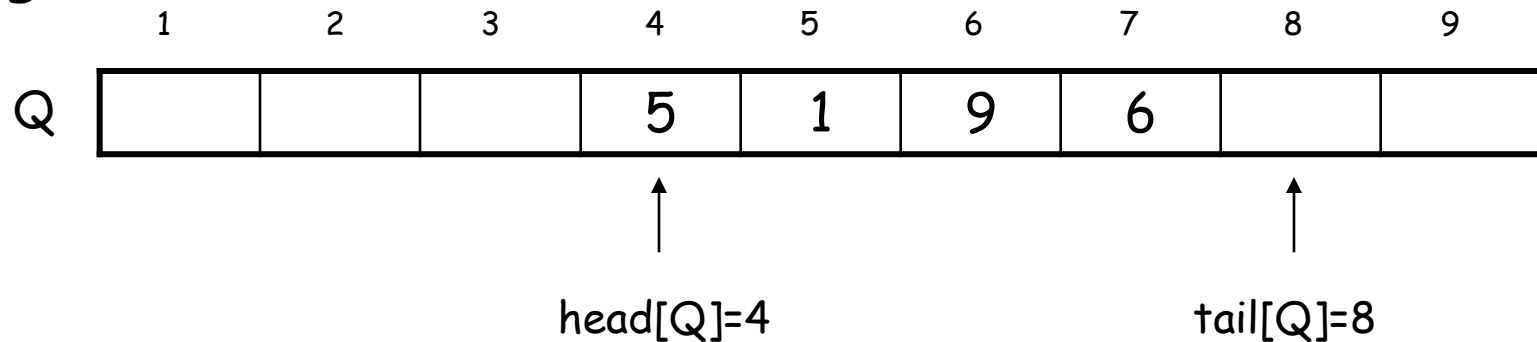
Esto supone varios aspectos:

- La cola tiene una capacidad limitada
- Se cuenta con dos atributos adicionales, $head[Q]$ que guarda el índice 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)

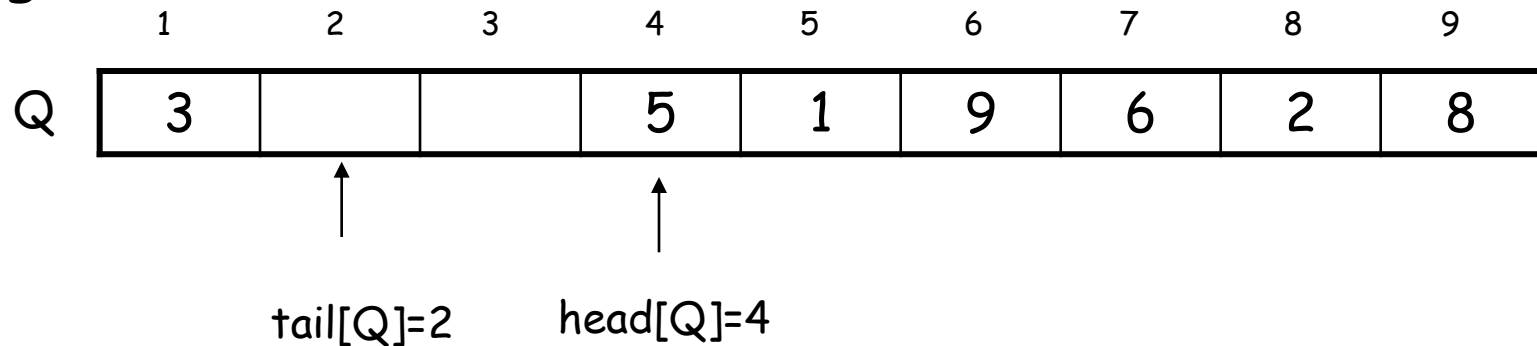
ENQUEUE(Q,8)

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)

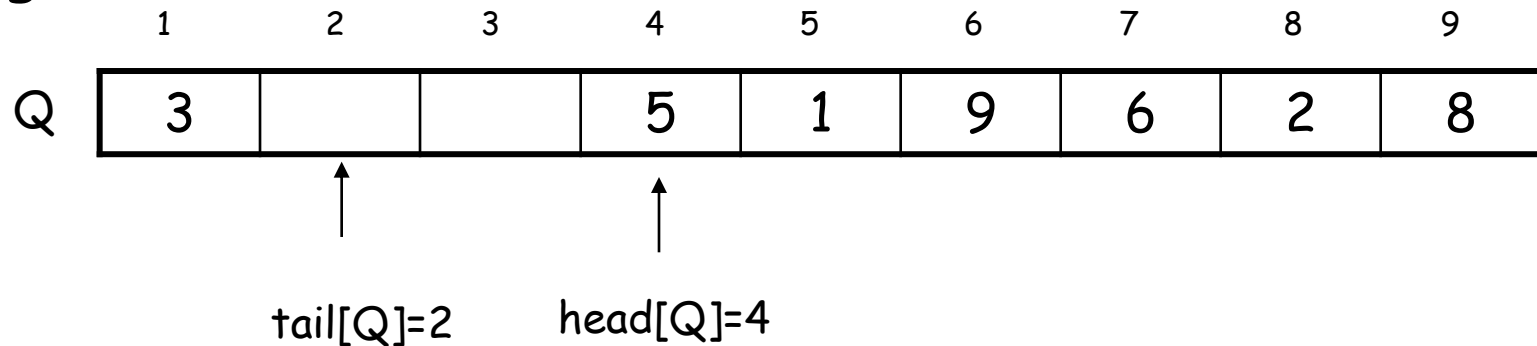
ENQUEUE(Q,3)

← Si se llega al final del arreglo, se intenta insertar en la posición 1

Estructuras de datos

Cola

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

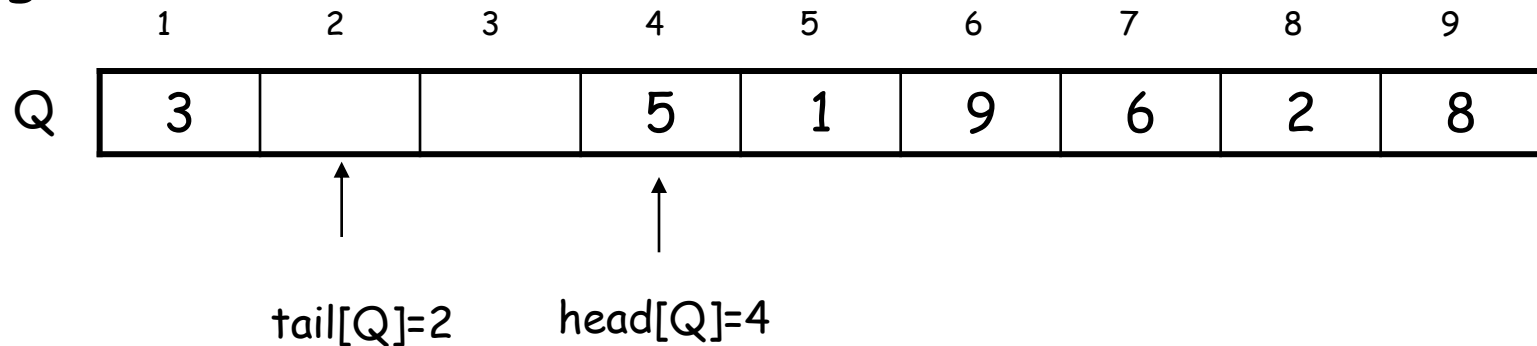


Cómo sabe que la cola está llena?

Estructuras de datos

Cola

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



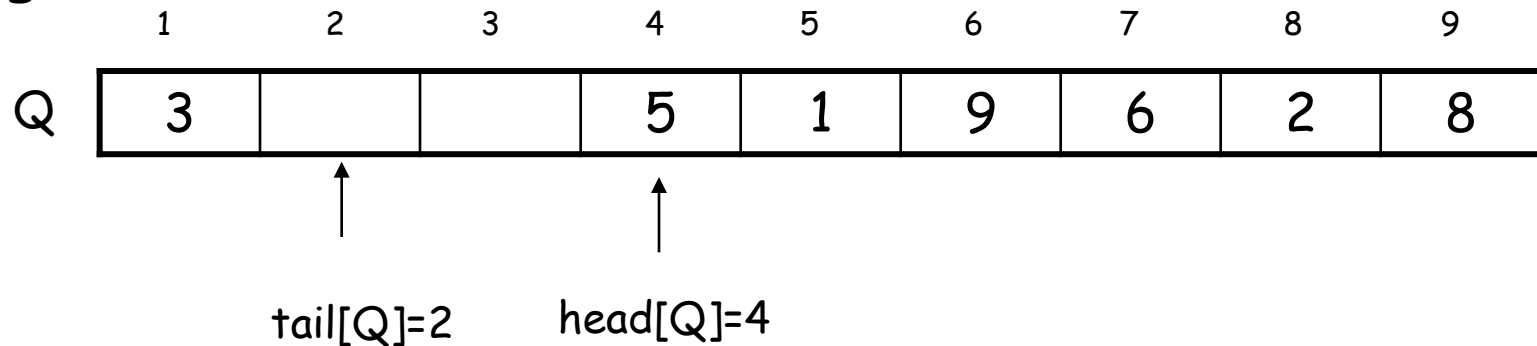
Cómo sabe que la cola está llena?

$tail[Q]=head[Q]$

Estructuras de datos

Cola

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



Inicialmente $\text{tail}[Q] = \text{head}[Q] = 1$

Estructuras de datos

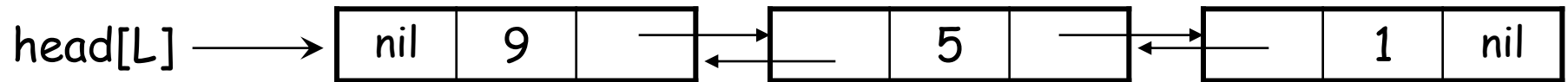
Indique un algoritmo para cada operación básica, acompañado de su respectiva complejidad usando la notación O

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

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

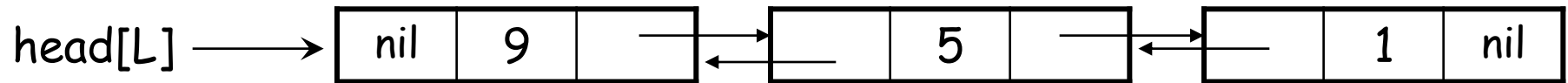


- Cada nodo en una lista doblemente enlazada tiene 3 campos: prev, key y next
- Se tiene además un puntero al primer nodo

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

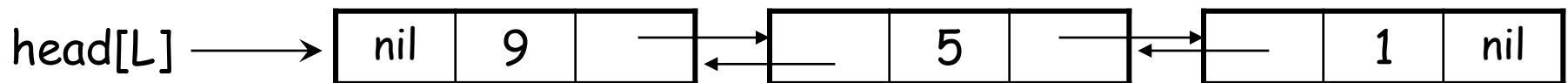
Estructuras de datos

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

LIST-SEARCH(L, k)

1. $x \leftarrow \text{head}[L]$
2. while $x \neq \text{nil}$ and $\text{key}[x] \neq k$
3. $x \leftarrow \text{next}[x]$
4. return x

¿Cuál es la complejidad en el peor caso?



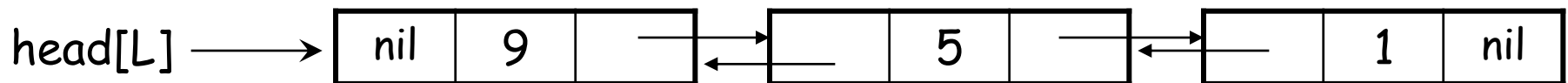
Estructuras de datos

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

LIST-SEARCH(L, k)

1. $x \leftarrow \text{head}[L]$
2. while $x \neq \text{nil}$ and $\text{key}[x] \neq k$
3. $x \leftarrow \text{next}[x]$
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

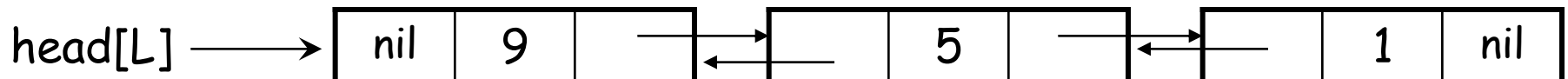
next[w]=nil

key[w]=8

LIST-INSERT(L,w)

x=LIST-SEARCH(L,10)

LIST-DELETE(L,x)



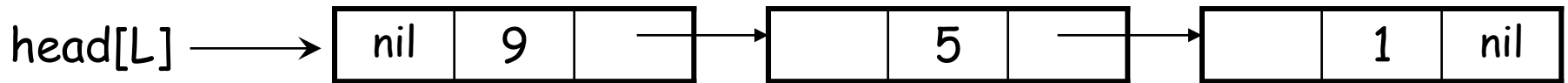
Estructuras de datos

Indique el algoritmo para las siguientes 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 $\text{key}[x]=k$, y $\text{prev}=\text{next}=\text{nil}$
- `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

Estructuras de datos

Árboles

Cada nodo tiene los campos p , $left$ y $right$ para almacenar los punteros al padre, hijo izquierdo e hijo derecho. Además, se tiene el campo key .

Si $p[x]=nil$ entonces x es la raíz

Si el nodo x no tiene hijo izquierdo entonces $left[x]=nil$

Si $left[x]=right[x]=nil$ entonces x es una hoja

Estructuras de datos

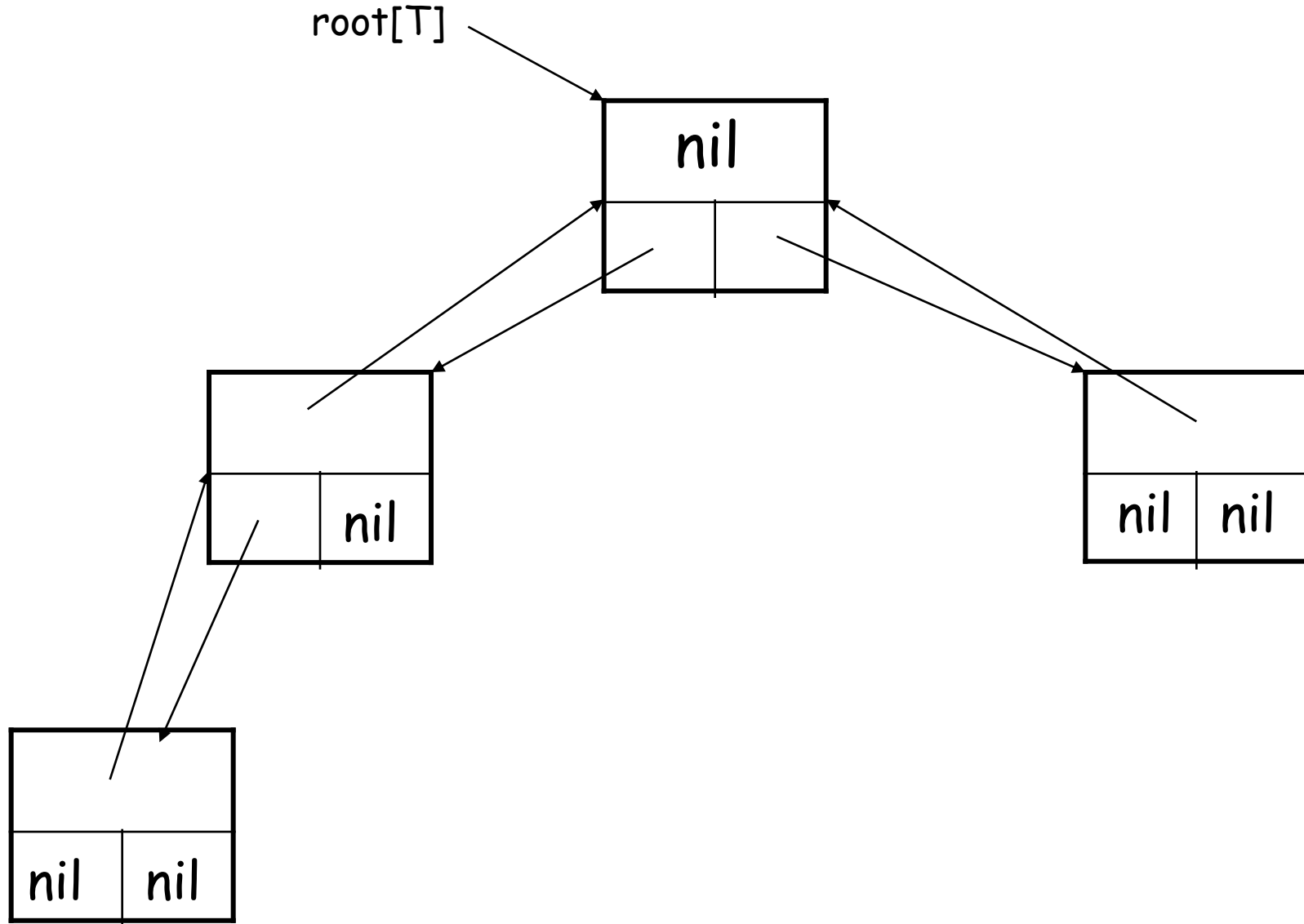
Árboles

Cada nodo tiene los campos `p`, `left` y `right` para almacenar los punteros al padre, hijo izquierdo e hijo derecho. Además, se tiene el campo `key`.

`root[T]` es el apuntador a la raíz del árbol

Si `root[T]=nil` entonces el árbol está vacío

Estructuras de datos



Estructuras de datos

Desarrolle un algoritmo que permita imprimir las llaves de un árbol binario, tal que en el peor caso sea $O(n)$