

# Método de ordenamiento por pilas

## ¿En qué consiste?

Mediante dos pilas auxiliares, se va ordenando cada elemento de la lista al ir comparando los topes de ambas con el primer elemento de la lista. Durante el proceso de ordenamiento, la lista se comporta como una cola.

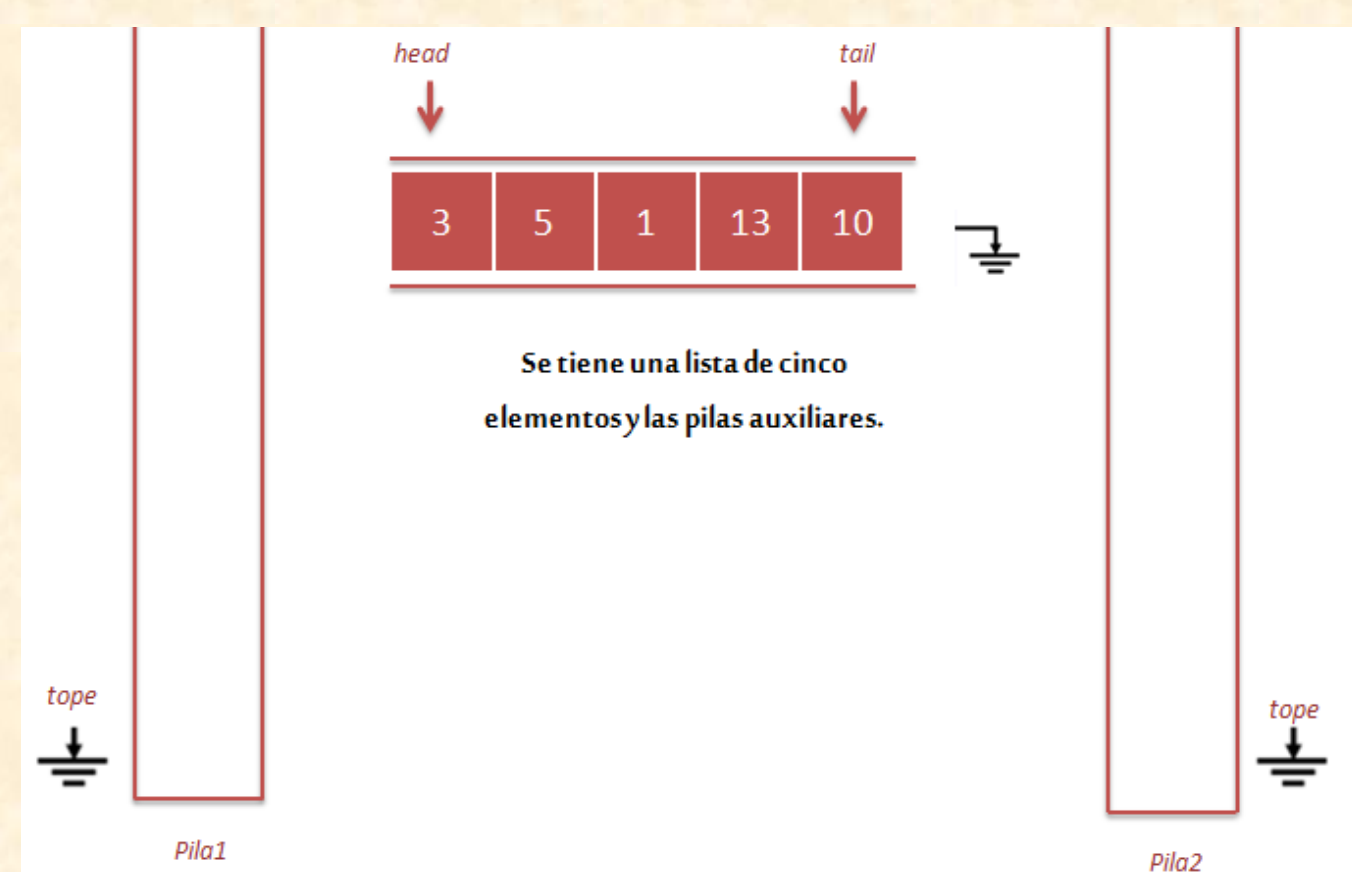


## Funciones necesarias

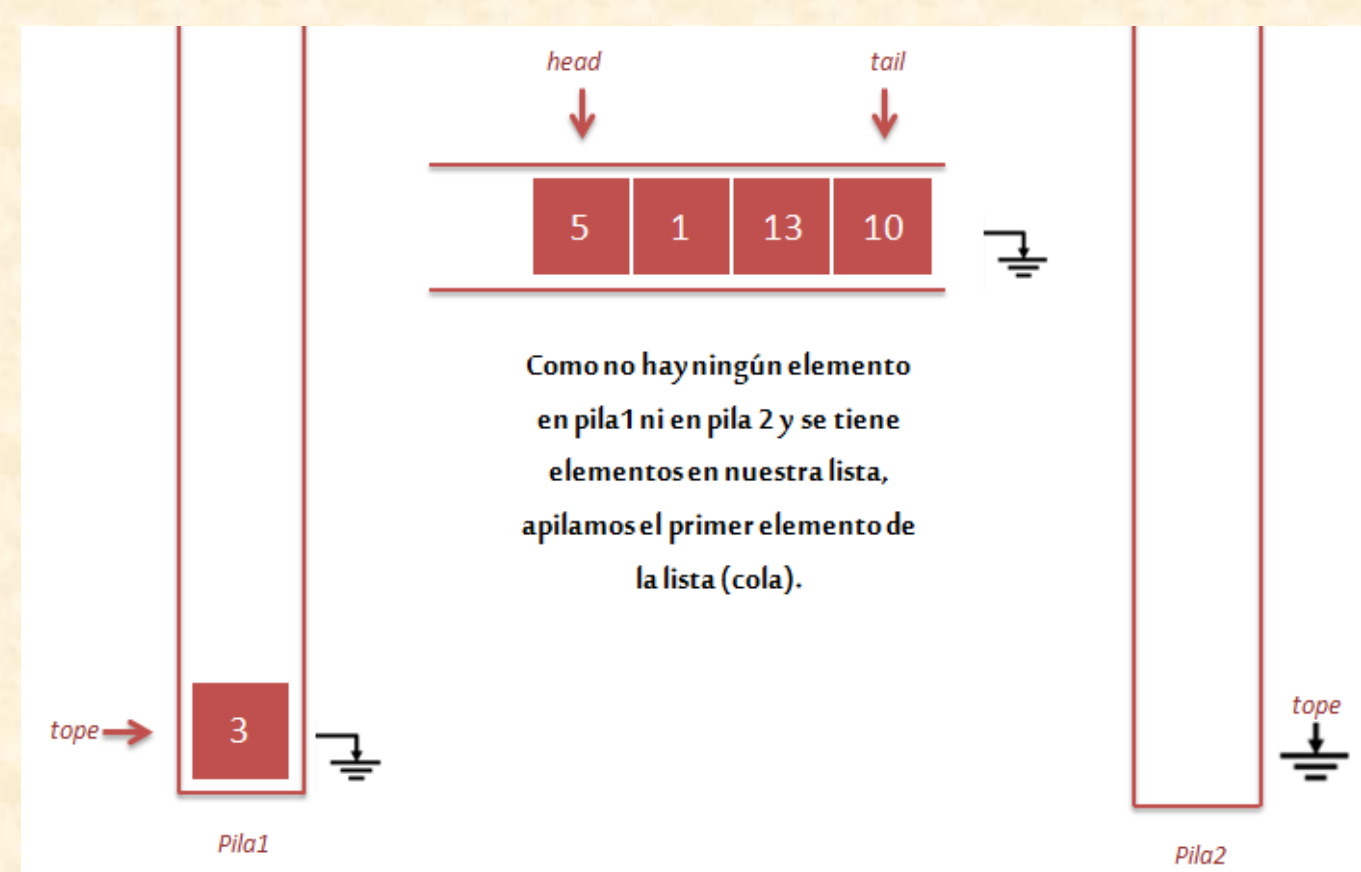
- ♣ SacarP (desencolar)
- ♣ Push
- ♣ Pop
- ♣ leerUltimo (top)
- ♣ leerPrimero(peek)

## Pseudocódigo

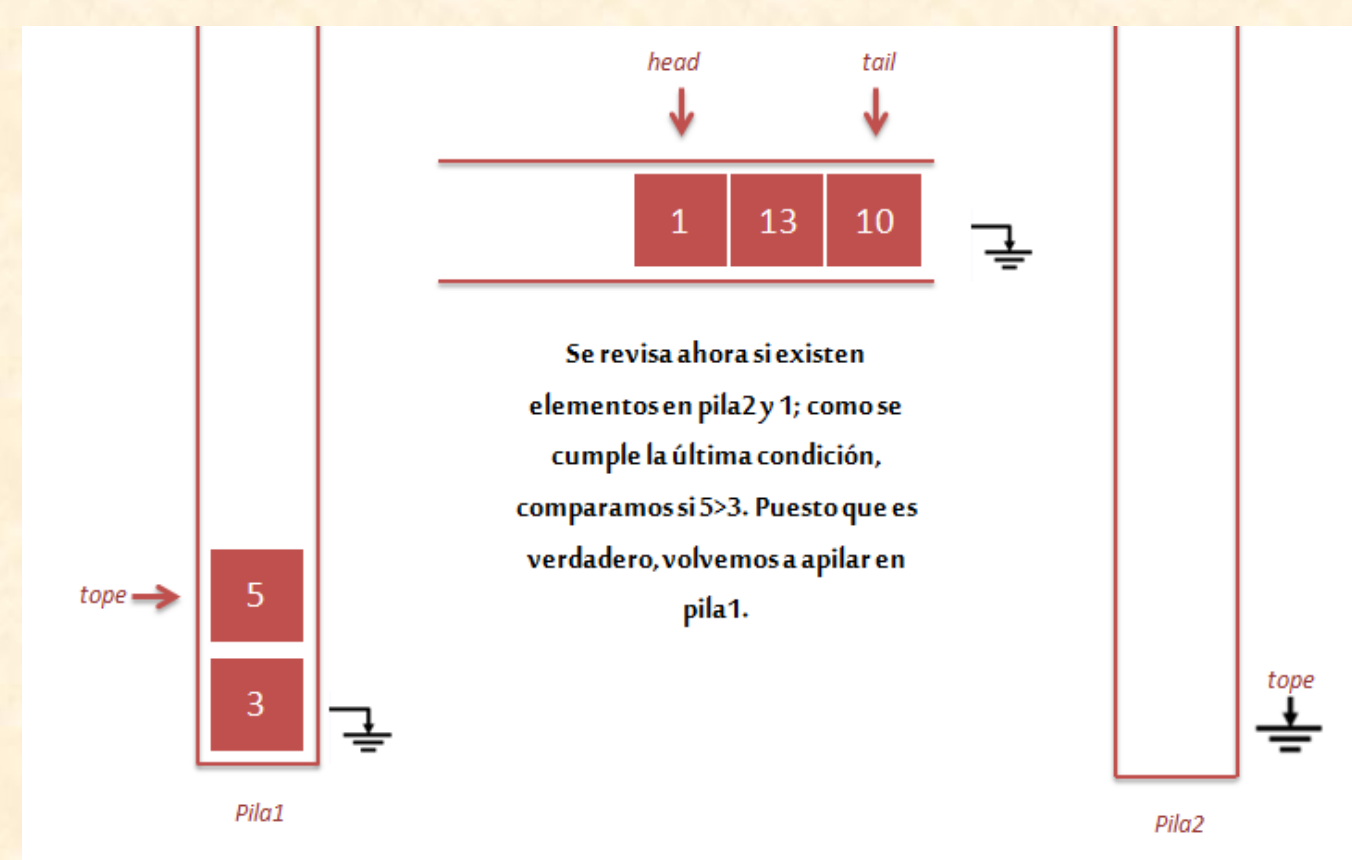
```
OrdenamientoInterno(A[]: Entero): DEV L0[]:ENTERO
Pila1 = ∅
Pila2 = ∅
SI A = ∅:
    DEV A
FIN SI
Apilar(pila1,desencolar(A))
MIENTRAS A ≠ ∅:
    SI pila1 ≠ ∅:
        Apilar(pila1,desencolar(A))
    FIN SI
    SI A = ∅:
        RomperCiclo
    FIN SI
    MIENTRAS pila2 ≠ ∅:
        SI A[0] > pila2[length(pila2)-1]:
            Apilar(pila1,pop(pila2))
        FIN SI
        EN CASO CONTRARIO:
            RomperCiclo
        FIN EN CASO CONTRARIO
    FIN MIENTRAS
    SI A[0] > pila1[length(pila1)-1]:
        Apilar(pila1,desencolar(A))
    FIN SI
    EN CASO CONTRARIO:
        Apilar(pila2,pop(pila1))
    FIN EN CASO CONTRARIO
    FIN MIENTRAS
    SI pila1 ≠ ∅:
        MIENTRAS pila1 ≠ ∅:
            Apilar(pila2,pop(pila1))
        FIN MIENTRAS
    FIN SI
    DEV pila2
```



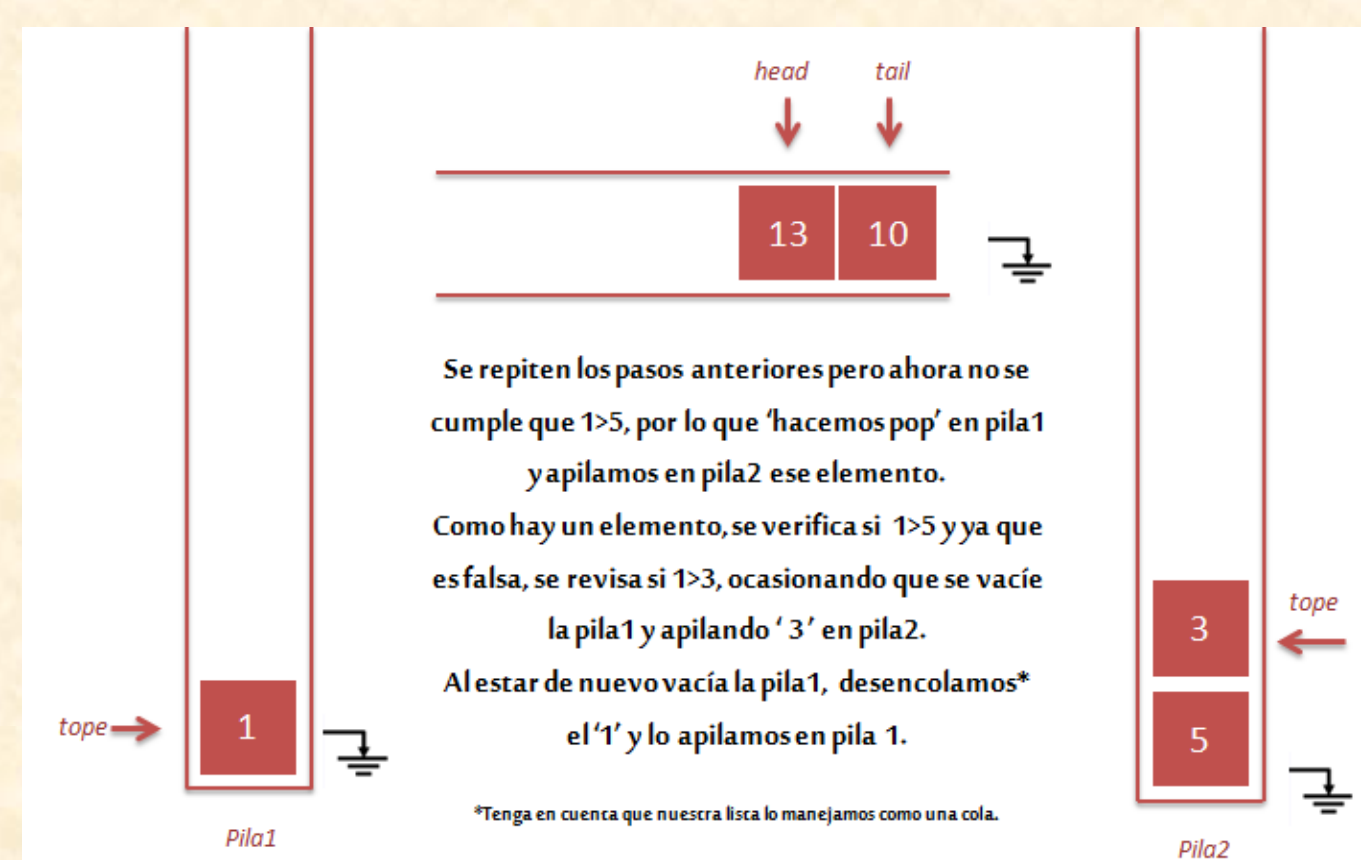
Se tiene una lista de cinco elementos y las pilas auxiliares.



Como no hay ningún elemento en pila1 ni en pila2 y se tiene elementos en nuestra lista, apilamos el primer elemento de la lista (cola).

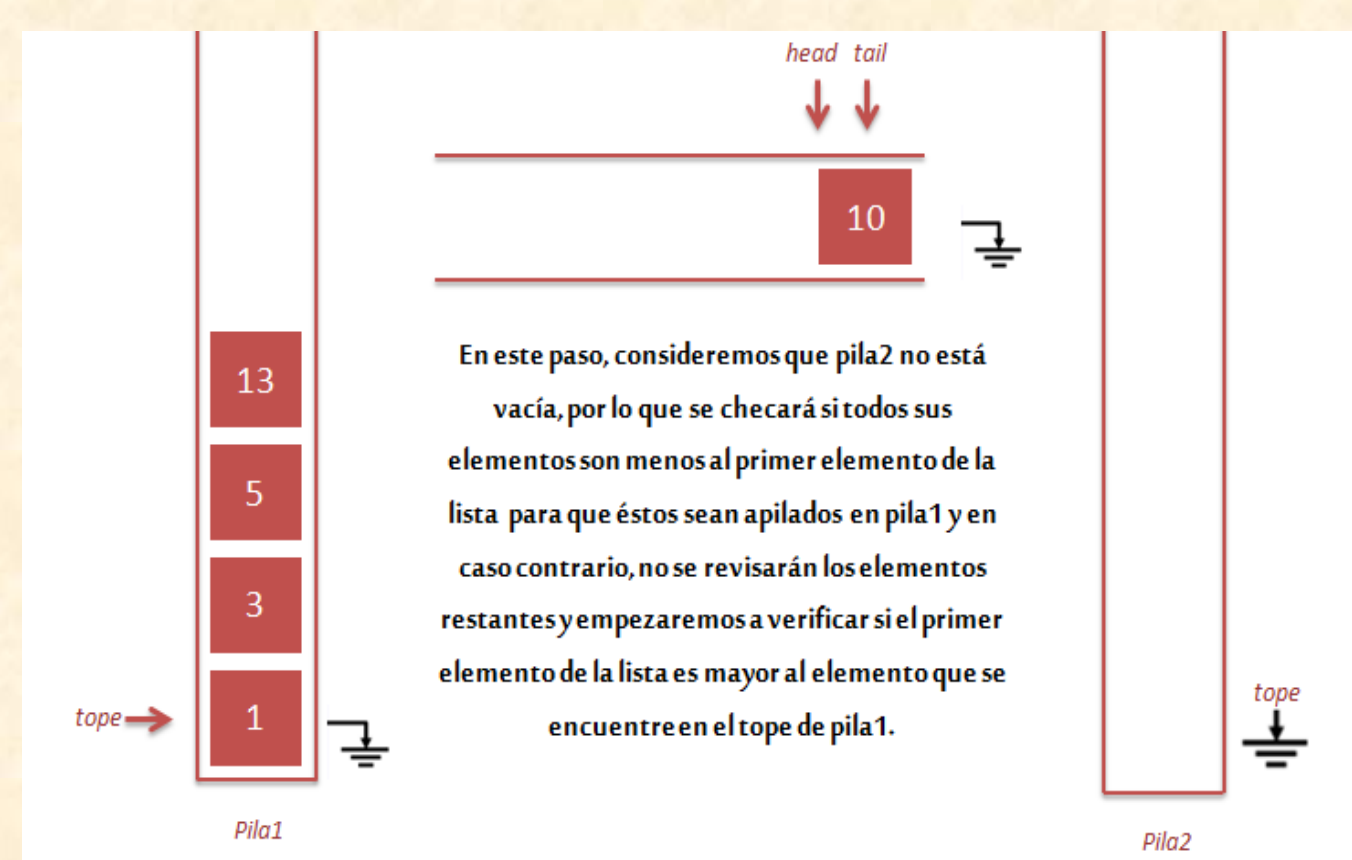


Se revisa ahora si existen elementos en pila2 y 1; como se cumple la última condición, comparamos si 5 > 3. Puesto que es verdadero, volvemos a apilar en pila1.

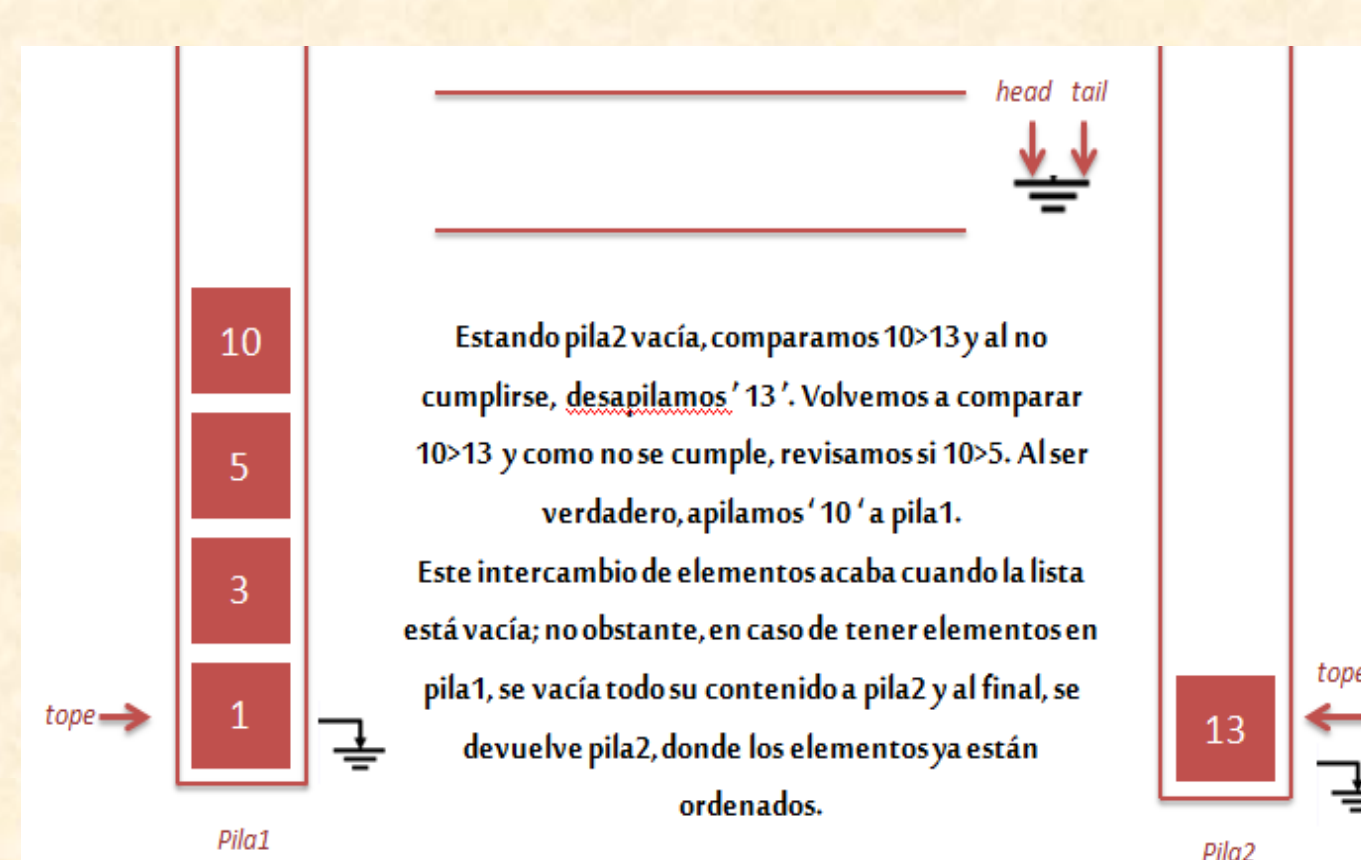


Se repiten los pasos anteriores pero ahora no se cumple que 1 > 5, por lo que 'hacemos pop' en pila1 y apilamos en pila2 ese elemento. Como hay un elemento, se verifica si 1 > 5 y ya que es falsa, se revisa si 1 > 3, ocasionando que se vacíe la pila1 y apilando '3' en pila2. Al estar de nuevo vacía la pila1, desencolamos\* el '1' y lo apilamos en pila 1.

\*Tenga en cuenta que nuestra lista lo manejamos como una cola.



En este paso, consideremos que pila2 no está vacía, por lo que se checará si todos sus elementos son menores al primer elemento de la lista para que éstos sean apilados en pila1 y en caso contrario, no se revisarán los elementos restantes y empezaremos a verificar si el primer elemento de la lista es mayor al elemento que se encuentre en el tope de pila1.



Estando pila2 vacía, comparamos 10 > 13 y al no cumplirse, desapilamos '13'. Volvemos a comparar 10 > 13 y como no se cumple, revisamos si 10 > 5. Al ser verdadero, apilamos '10' a pila1. Este intercambio de elementos acaba cuando la lista está vacía; no obstante, en caso de tener elementos en pila1, se vacía todo su contenido a pila2 y al final, se devuelve pila2, donde los elementos ya están ordenados.



Estando pila2 vacía, comparamos 10 > 13 y al no cumplirse, desapilamos '13'. Volvemos a comparar 10 > 13 y como no se cumple, revisamos si 10 > 5. Al ser verdadero, apilamos '10' a pila1. Este intercambio de elementos acaba cuando la lista está vacía; no obstante, en caso de tener elementos en pila1, se vacía todo su contenido a pila2 y al final, se devuelve pila2, donde los elementos ya están ordenados.

Como podemos notar, lo que se regresa es una pila e inicialmente lo que se mandó es una lista por lo que causa una **incompatibilidad**. Esto no es problema si todo lo trabajamos desde un **inicio con listas**. Sin embargo, deben utilizarse como **pilas** y la lista a ordenar como **cola**.

Al realizar esa modificación, se debe mandar al final todos los elementos a la pila1 (en caso de querer una lista ordenada de menor a mayor).

Algoritmo de ordenamiento creado por Suexo Pacheco Elsa Guadalupe· Facultad de Ingeniería· Semestre 2019-1