Ejercicios del profe: Colas 2025-1

Ludwig Alvarado Becerra

9 de junio de 2025 — Universidad Jorge Tadeo Lozano - Semillero de Programación Competitiva

Problema 2

Problema 2

Dada una **cola** y un número **k**, invierte el orden de los primeros **k** elementos de la **cola**. Si **k** es mejor que **0**, si **k** excede el tamaño de la **cola** o si la **cola** está vacía, se devuelve **NULL**. De lo contrario, se devuelve la **cola** modificada.

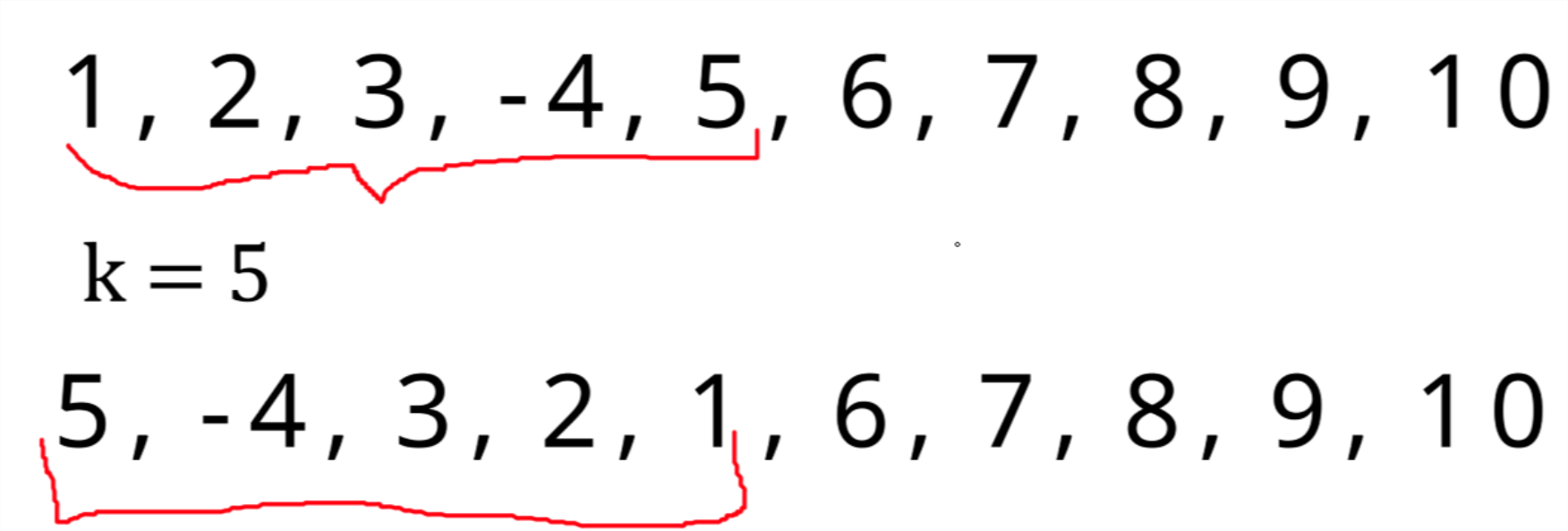
Restricciones

- $0 \leq cola.length \leq 10^3$
- $10^{-3} \leq cola[i] \leq 10^3$
- $10^{-3} \leq k \leq 10^3$

Problema 2 | Tests

Entrada	Salida
$cola = [1, 2, 3, -4, 5, 6, 7, 8, 9, 10] \quad k = 5$	$[5, -4, 3, 2, 1, 6, 7, 8, 9, 10]$
$cola = [10, -20, 30, 40, -50, 60, 70, 80] \quad k = 3$	$[30, -20, 10, 40, -50, 60, 70, 80]$
$cola = [5, 6, 12, 5, 2, 7, 3] \quad k = 10$	NULL
$cola = [43, 5, 12, 4, 9, 6, 5] \quad k = -5$	NULL

Test 1

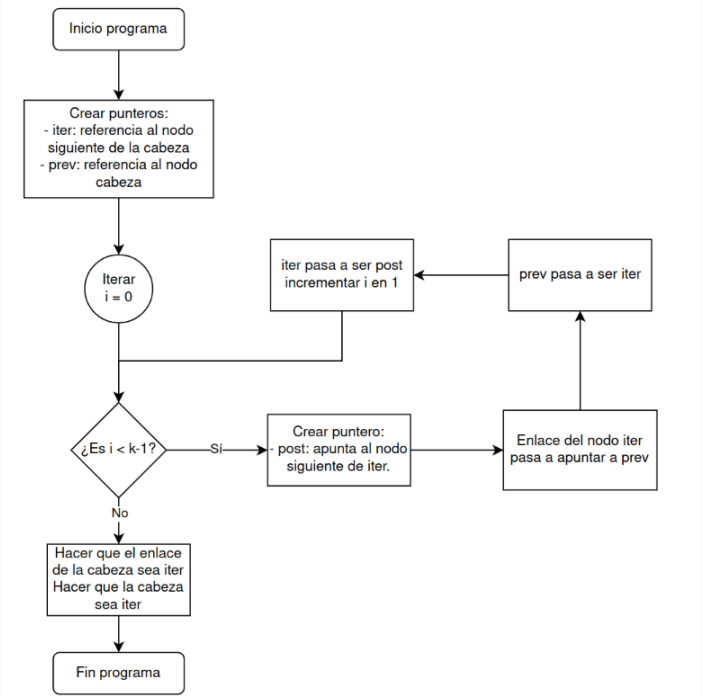


Solución

Prerrequisitos

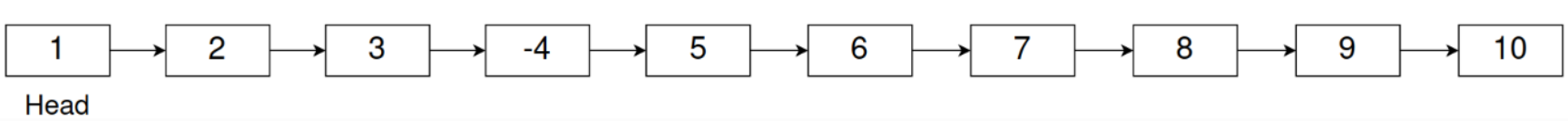
- Se utiliza un algoritmo para invertir una lista enlazada.
- Ejercicio recomendado: https://leetcode.com/problems/reverse-linked-list/description/
- Solución explicada del ejercicio recomendado en: https://github.com/ProgramacionCompetitivaUTADEO/Estructuras-De-Datos/tree/main/C%2B%2B/LinkedList/Ejercicios/LC206ReverseLinkedList

Diagrama de flujo

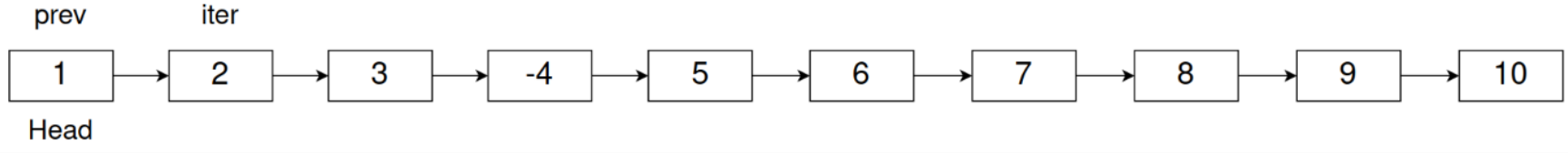


Funcionamiento del algoritmo

Cola inicial

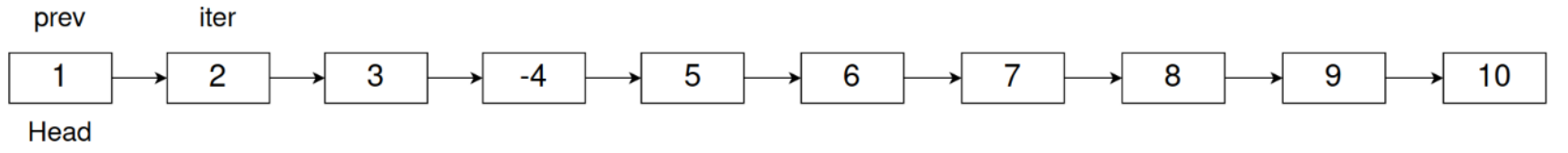


Inicializar los punteros `iter` y `prev`

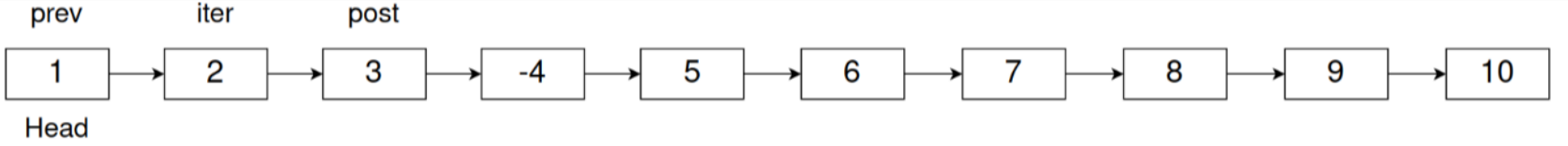


Funcionamiento del algoritmo

Inicia el bucle $i = 0$, condición $i < k - 1$. $k = 5$.

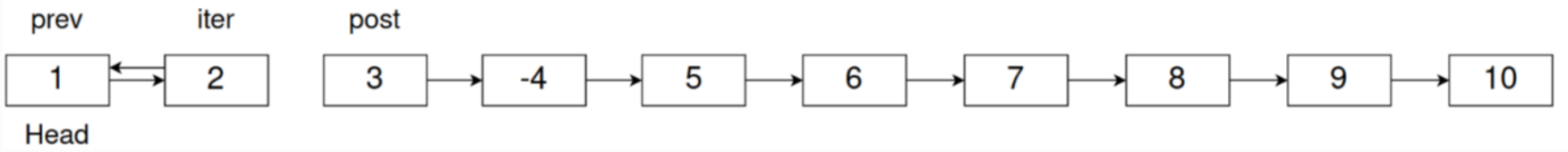


Crear `post` que sea el nodo al que apunta `iter`

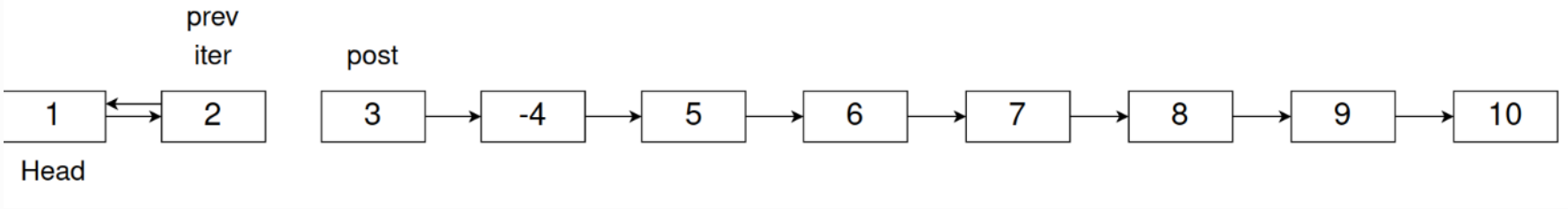


Funcionamiento del algoritmo

Hacer que el nodo `iter` apunte al nodo `prev`

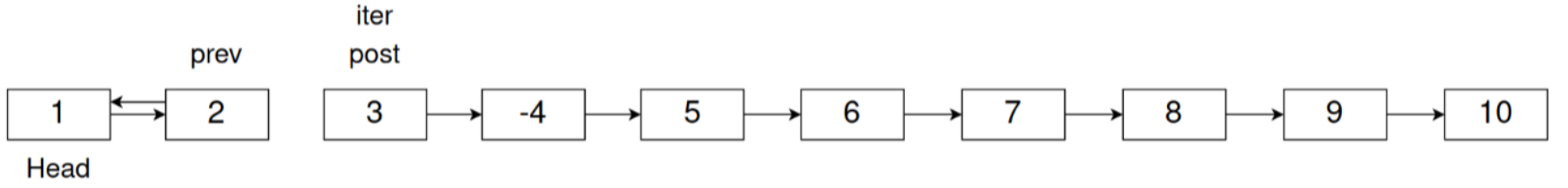


Mover `prev` a `iter`

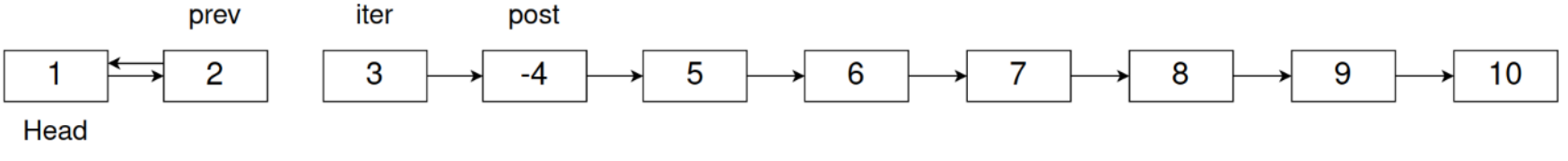


Funcionamiento del algoritmo

Mover `iter` a `post`.

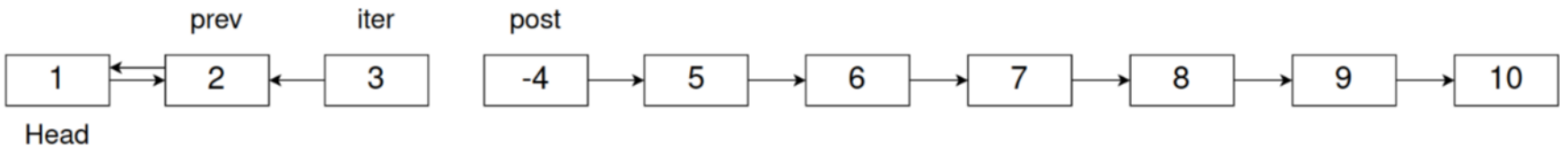


$i = 1$, condición $i < k - 1$. $k = 5$. Crear `post` que sea el nodo al que apunta `iter`

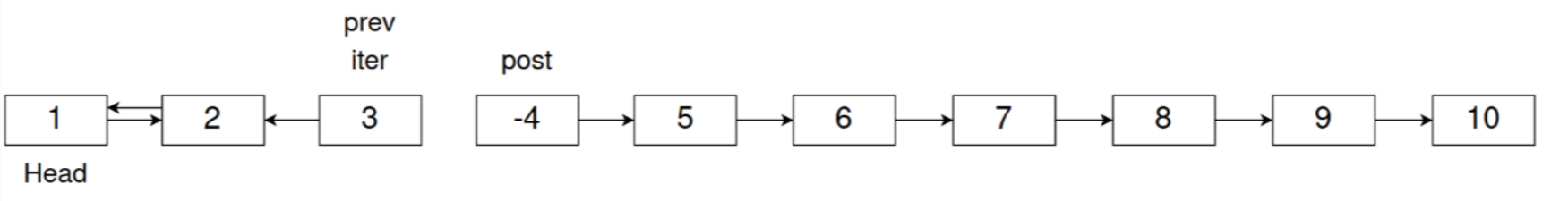


Funcionamiento del algoritmo

Hacer que el nodo `iter` apunte al nodo `prev`

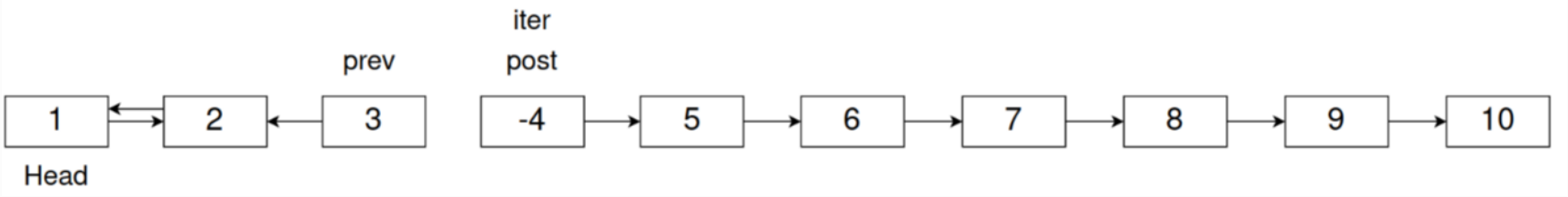


Mover `prev` a `iter`

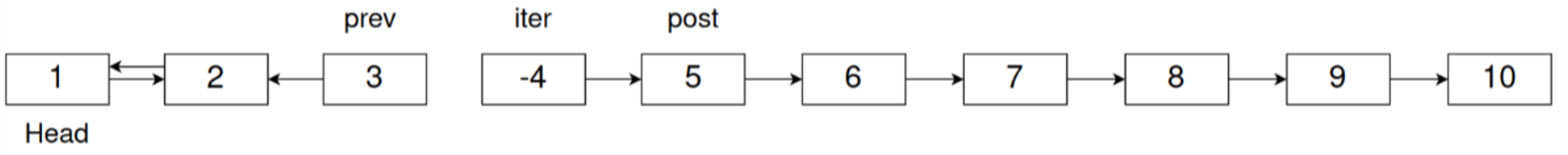


Funcionamiento del algoritmo

Mover `iter` a `post`

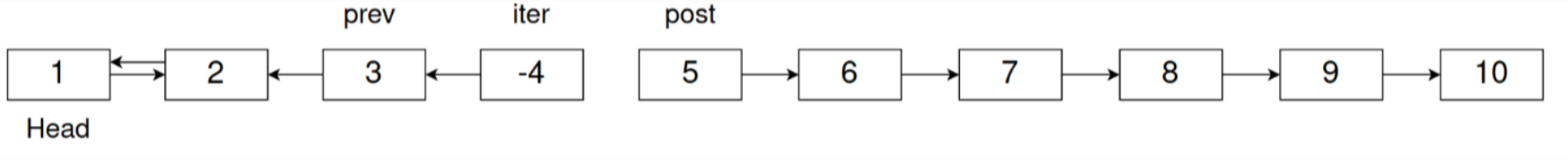


$i = 2$, condición $i < k - 1$. $k = 5$. Crear `post` que sea el nodo al que apunta `iter`

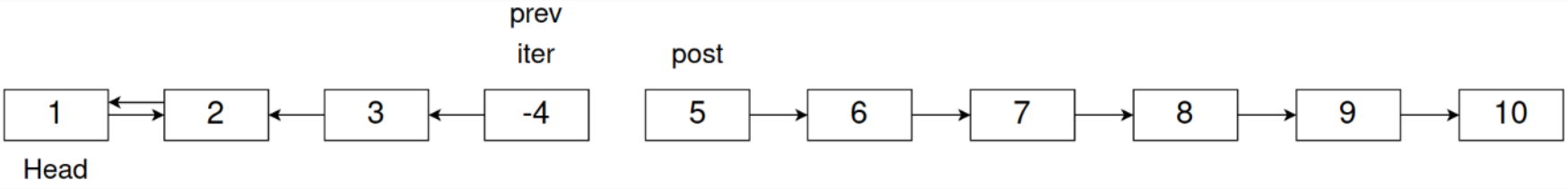


Funcionamiento del algoritmo

Hacer que el nodo `iter` apunte al nodo `prev`

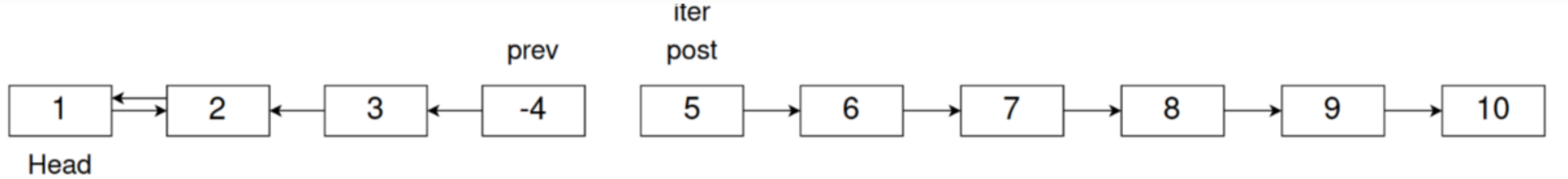


Mover `prev` a `iter`

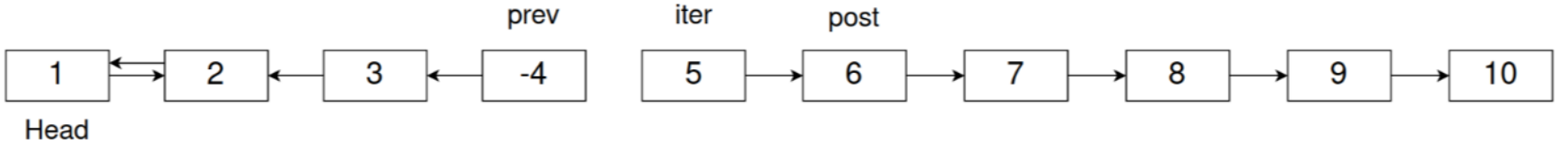


Funcionamiento del algoritmo

Mover `iter` a `post`

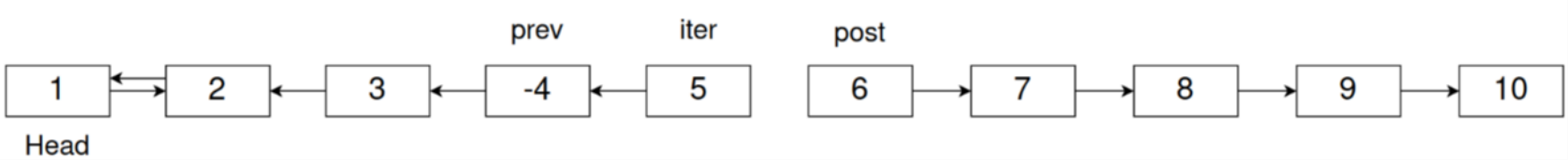


$i = 3$, condición $i < k - 1$. $k = 5$. Crear `post` que sea el nodo al que apunta `iter`

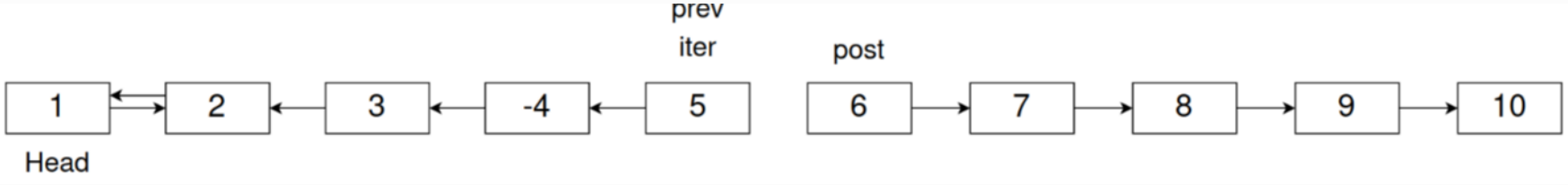


Funcionamiento del algoritmo

Hacer que el nodo `iter` apunte al nodo `prev`

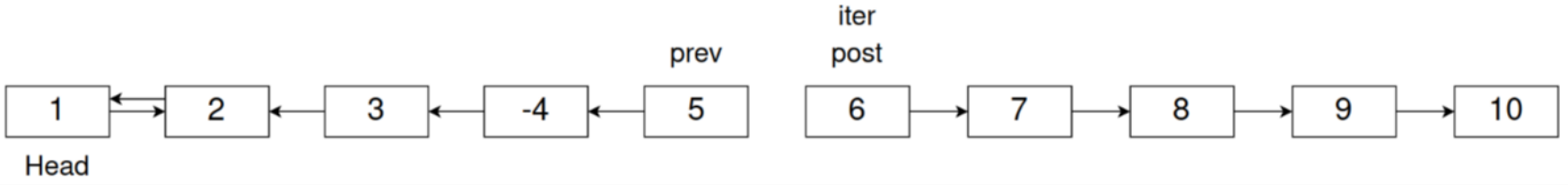


Mover `prev` a `iter`

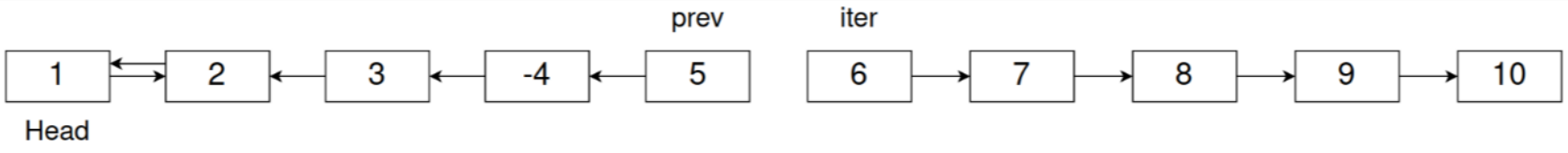


Funcionamiento del algoritmo

Mover `iter` a `post`

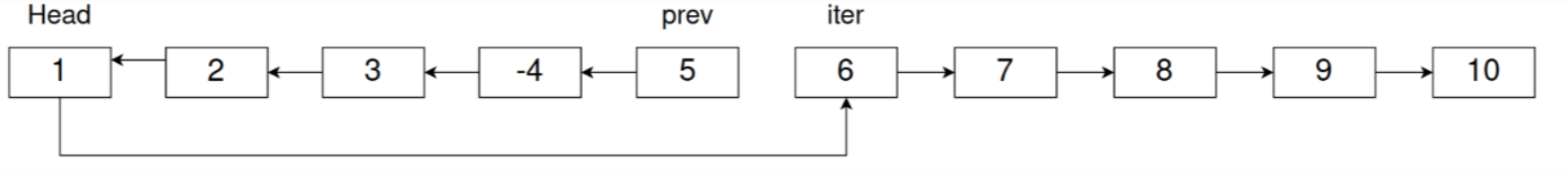


$i = 4$, condición $i < k - 1$. $k = 5$. **Finaliza el ciclo**.

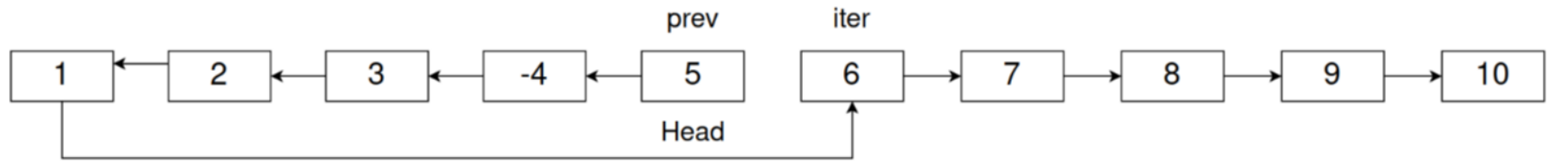


Funcionamiento del algoritmo

Hacer que `head` apunte a `iter`.

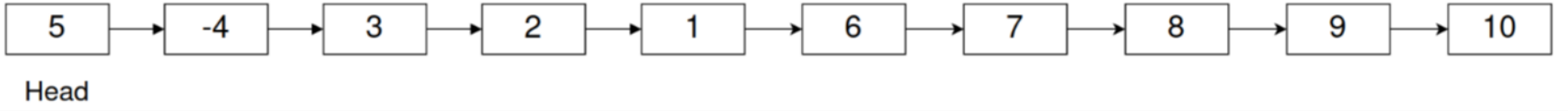


Hacer que `head` sea `prev`.

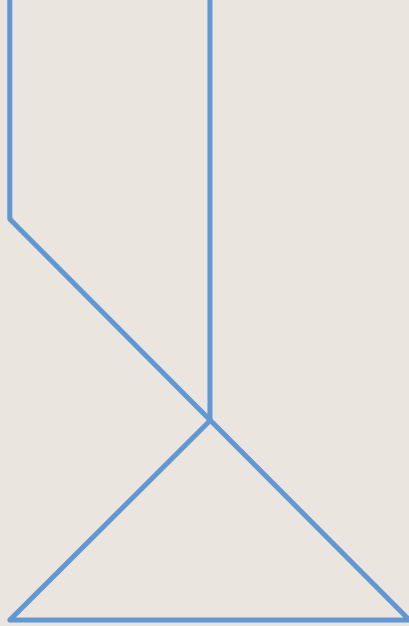


Funcionamiento del algoritmo

La lista entonces se ve:



UTADEO



Implementación En C++

Implementación en C++

```cpp
#include "Queue.h"
#define endl "\n"

void reverse_at_k_2(Queue<int> cola, int k){

    if (k < 0 | k > cola.get_size() | cola.is_empty()) { cout << "NULL" << endl;
       return;}

    Node<int>* iter = cola.front() -> next;
    Node<int>* prev = cola.front();
    for (int i =0; i < k-1; i++){
        Node<int>* post = iter -> next;
        iter -> next = prev;
        prev = iter;
        iter = post;
    }
    cola.head -> next = iter;
    cola.head = prev;
}
```

UTADEO

Complejidad temporal y espacial

Complejidad temporal y espacial

- Mayoría de casos $O(k)$. Peor de los casos $k = n$

$$O(n)$$

- Espacial, no se crean nuevas estructuras.

$$O(1)$$