Data Structures and Algorithms

Lecture 12: Linked List - Circular Header linked list

Header Linked List

- A Header linked list is one more variant of linked list.
- In Header linked list, we have a special node present at the beginning of the linked list.
- This special node is used to store number of nodes present in the linked list.
- In other linked list variant, if we want to know the size of the linked list we use traversal method.
- But in Header linked list, the size of the linked list is stored in its header itself.

Types of Header Linked List

Types of Header linked list

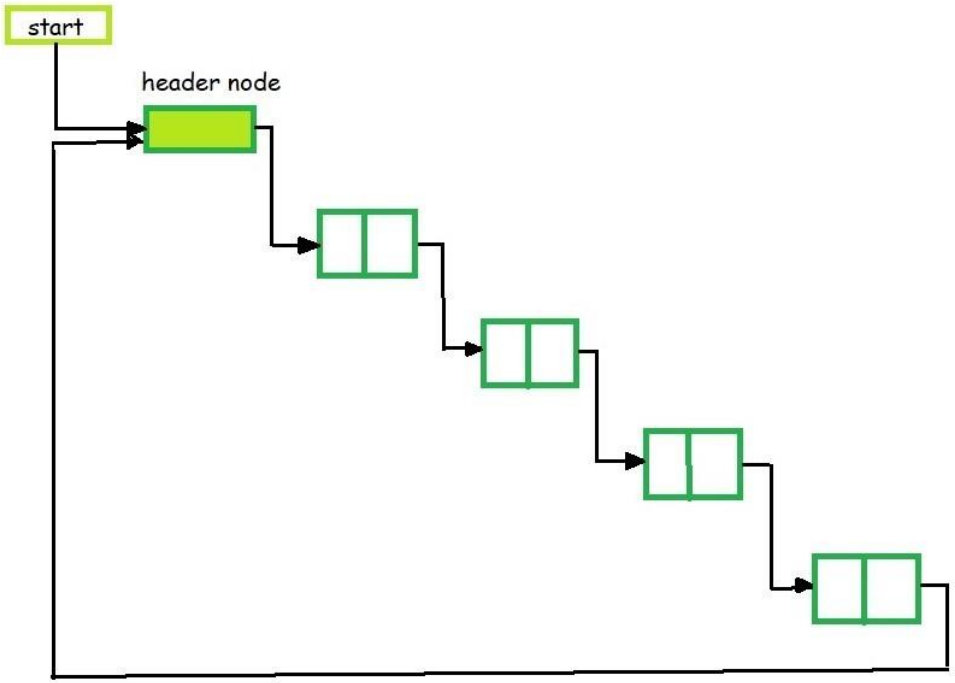
- Grounded header linked list.
- Circular header linked list.

Structure – Header Linked list

```
struct node
{
        int data;
        struct node *next;
};
```

Circular Header Linked List

- A list in which last node points back to the header node is called circular linked list.
- The chains do not indicate first or last nodes. In this case, external pointers provide a frame of reference because last node of a circular linked list does not contain the NULL pointer.
- The possible operations on this type of linked list are Insertion, Deletion and Traversing.



Circular Header Linked List

- Circular Structure: Unlike traditional linked lists, where the last node points to nullptr, circular header linked lists form a loop by having the last node point back to the header node. This circular structure enables seamless traversal from the last node to the first node without requiring any special handling or extra checks.

- Header Node: The header node in a circular header linked list contains information about the list, such as the total number of nodes or any other relevant metadata. The header node is typically not used to store actual data, but rather to maintain the structure and provide an entry point for accessing the list.

Circular Header Linked List

Advantages:

- Efficient Traversal: The circular structure of the linked list allows for efficient traversal in both forward and backward directions. Starting from the header node, you can traverse the entire list by following the next pointers until you reach the header node again. This property is particularly useful in scenarios where you need to repeatedly iterate over the list in different directions.

- Simplified Operations: Circular header linked lists simplify certain operations compared to traditional linked lists. For example, appending a new node to the end of the list involves updating the next pointer of the current last node to point to the new node and updating the next pointer of the new node to point back to the header node. No special case handling is required for an empty list or for reaching the end of the list.

Recall:

Headed linked list – Circular header linked list