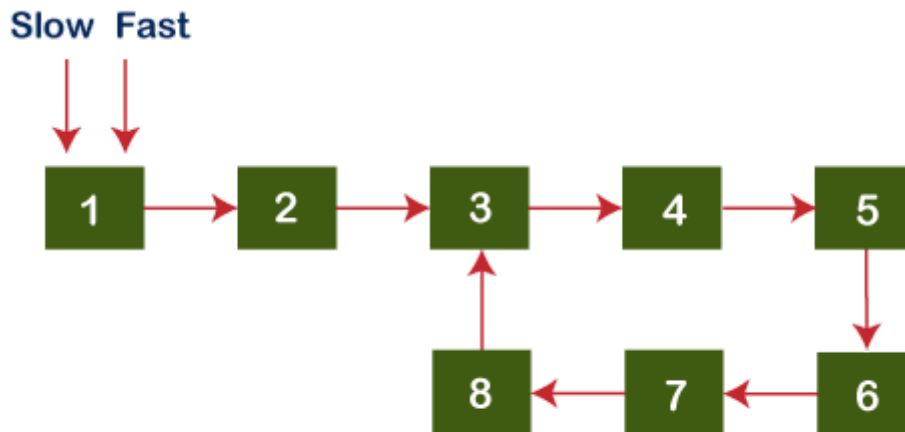


[Home](#)[Data Structure](#)[C](#)[C++](#)[C#](#)[Java](#)[SQL](#)[HTML](#)[CSS](#)[JavaScript](#)[Ajax](#)[↑ SCROLL TO TOP](#)

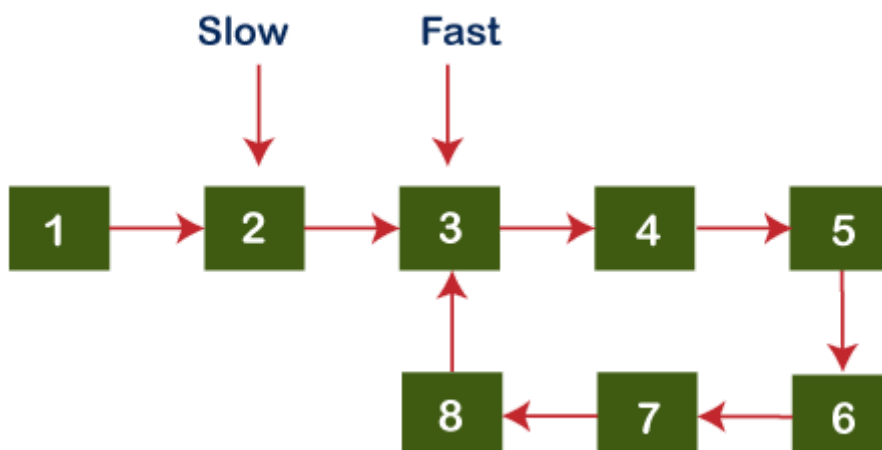
Remove the loop in a Linked List

In this topic, we will learn how to remove the loop from the linked list. Till now, we have learnt how to detect and start of the loop by using **Floyd's** algorithm. The **Floyd's** algorithm will also be used to remove the loop from the linked list.

Let's understand through an example.

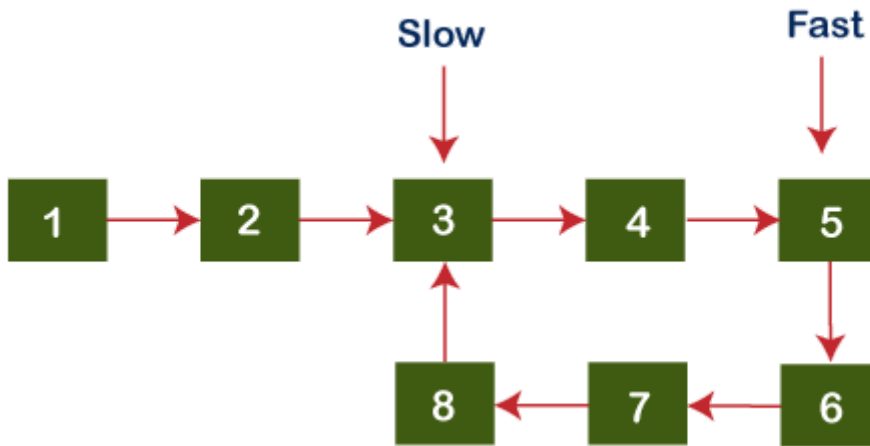


As we know that slow pointer increments by one and fast pointer increments by two. In the above example, initially, both slow and fast pointer point to the first node, i.e., node 1. The slow pointer gets incremented by one, and fast pointer gets incremented by two, and slow and fast pointers point to nodes 2 and 3, respectively, as shown as below:

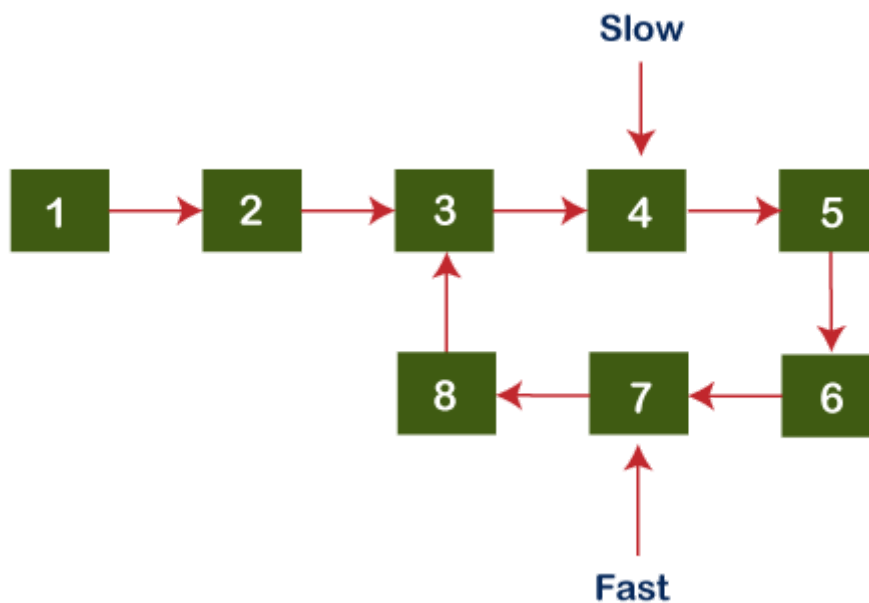


Since both the pointers do not point to the same node, we will increment both fast and slow pointers again. Now, slow pointer points to the node 3 while the fast pointer points to node 5 shown as below:

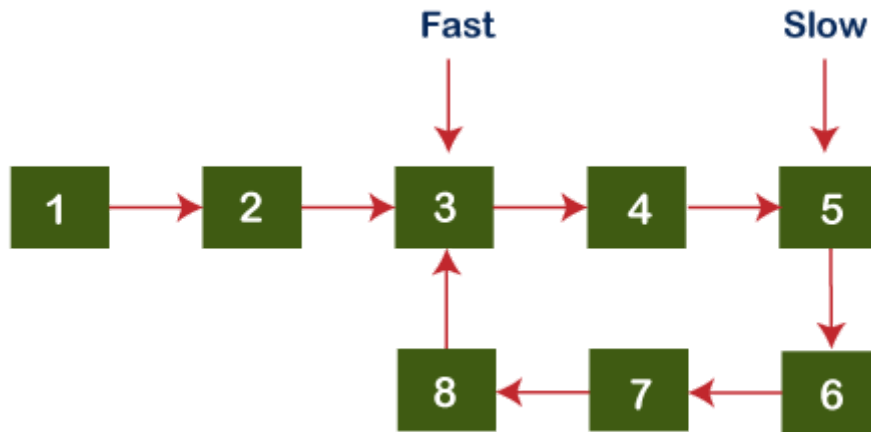
↑ SCROLL TO TOP



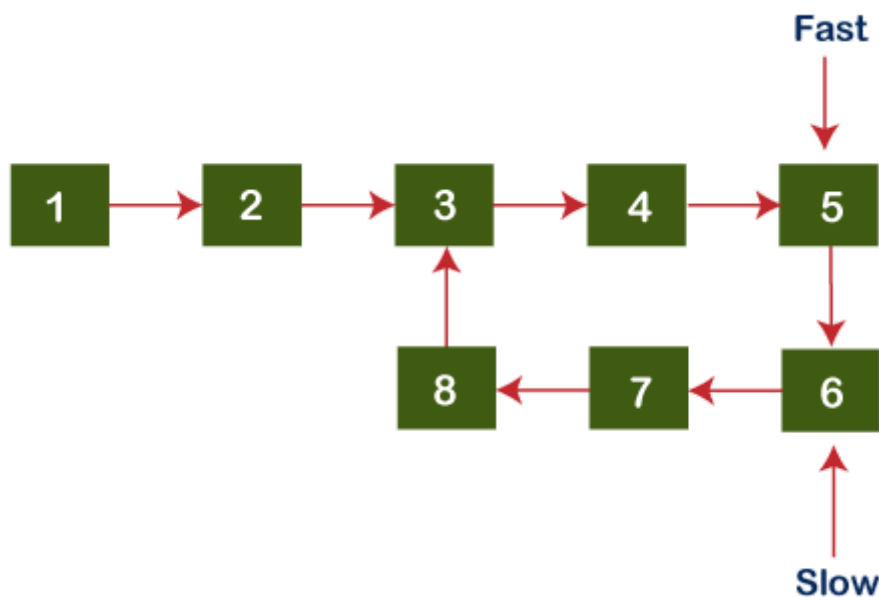
Since both the pointers do not point to the same node, we will increment both fast and slow pointers again. Now, the slow pointer points to node 4 while the fast pointer points to node 7 shown as below:



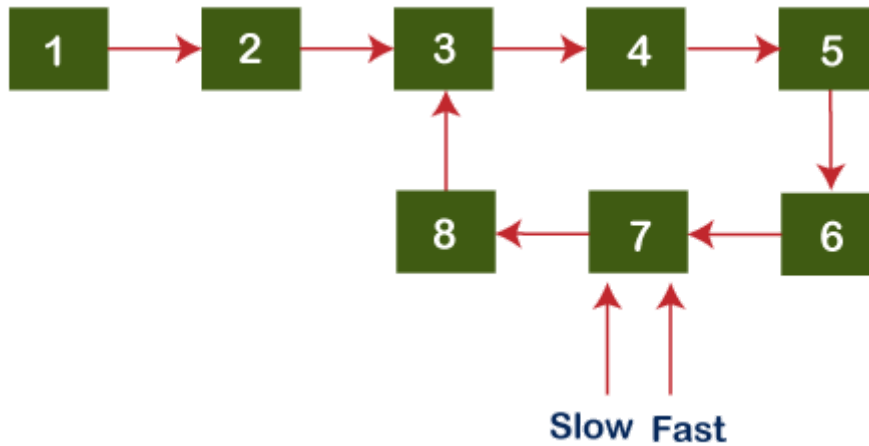
Since both the pointers do not point to the same node, we will increment both fast and slow pointers again. Now, the slow pointer points to node 5, and the fast pointer points to node 3 shown as below:



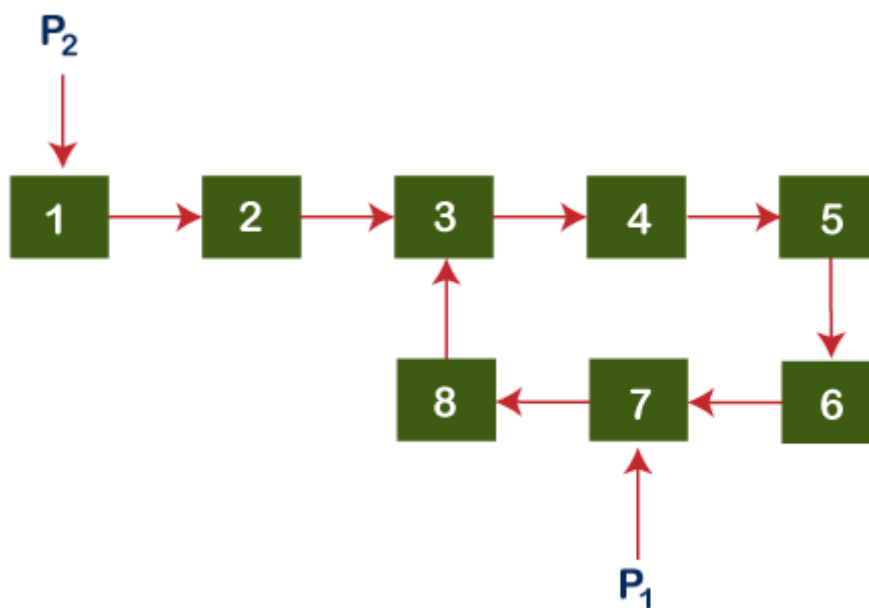
Since both the pointers do not point to the same node, we will increment both fast and slow pointers again. Now, slow pointer points to node 6 while the fast pointer points to the node 5 shown as below:



Again, both the pointers (slow and fast) are not pointing to the same node, so we will increment both fast and slow pointers again. Now, the slow pointer points to node 7, and the fast pointer also points to node 7 shown as below:



As we can observe in the above example both slow and fast pointers meet at node 7. We create one pointer named p_1 that points to the node 7 where both the pointers meet, and we also create one more pointer named p_2 that points to the first node as shown in the below figure:



To remove the loop, we will define the following logic:

```
while(p1.next!= p2.next)
{
    P1 = P1.next;
    P2 = P2.next;
}
P1.next = NULL;
```

↑ SCROLL TO TOP

The above logic is defined to remove a loop from the linked list. The while loop will execute till the p1.next is not equal to p2.next. When p1.next is equal to p2.next, the control will come out of the while loop, and we set the p1.next equal to Null. This statement breaks the link, which is creating the loop in the linked list.

Implementation of removing a loop in C

```
// C program to remove a loop from the linked list
#include <stdio.h>
#include <stdlib.h>
// creating a structure of a node
struct node
{
    int data;
    struct node *next;
};
struct node *head;
// creating a node in a linked list..
struct node* create_node(int value)
{
    // creating a temp variable of type node..
    struct node *temp = (struct node*)malloc(sizeof(struct node));
    temp->data = value;
    temp->next = NULL;
    return temp;
}
// detecting a node that creates a loop in the linked list..
int detect_loop(struct node *ptr)
{
    struct node *fast; // declaration of fast pointer of type node..
    struct node *slow; // declaration of slow pointer of type node..
    fast = slow = ptr;
    while (slow && fast && fast->next)
```

↑ SCROLL TO TOP

```
    slow = slow->next;
    fast = fast->next->next;
    // checking whether slow is equal to fast or not.
    if(slow==fast)
    {
        removeloop(slow, head); // calling removeloop() function.
        return 1;
    }
}

// Removing a loop from the linked list..
void removeloop(struct node *slow, struct node *head)
{
    struct node *p1;
    struct node *p2;
    p1 = slow;
    p2 = head;

    // while loop will execute till the p1.next is not equal to p2.next..
    while(p1->next!=p2->next)
    {
        p1 = p1->next;
        p2 = p2->next;
    }
    p1->next = NULL;
}

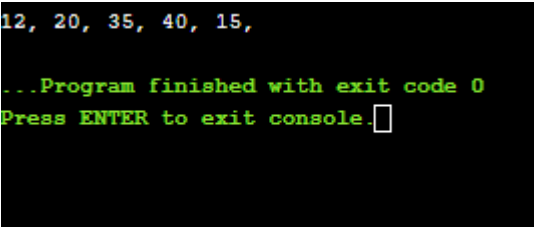
// display the linked list..
void display(struct node *temp)
{
    while (temp !=NULL)
    {
        printf("%d, ", temp->data);
```

>next;

↑ SCROLL TO TOP

```
}  
  
int main()  
{  
    head = create_node(12);  
    head->next = create_node(20);  
    head->next->next = create_node(35);  
    head->next->next->next = create_node(40);  
    head->next->next->next->next = create_node(15);  
    /* Create a loop for testing */  
    head->next->next->next->next->next = head->next->next;  
    detect_loop(head);  
    display(head);  
    return 0;  
}
```

Output



```
12, 20, 35, 40, 15,  
...Program finished with exit code 0  
Press ENTER to exit console.█
```

[< Prev](#)[Next >](#)

 [For Videos Join Our Youtube Channel: Join Now](#)












Feedback

- Send your Feedback to feedback@javatpoint.com






Help Others, Please Share

[↑ SCROLL TO TOP](#)


Learn Latest Tutorials

 Splunk tutorial Splunk	 SPSS tutorial SPSS	 Swagger tutorial Swagger	 T-SQL tutorial Transact-SQL
 Tumblr tutorial Tumblr	 React tutorial ReactJS	 Regex tutorial Regex	 Reinforcement learning tutorial Reinforcement Learning
 R Programming tutorial R Programming	 RxJS tutorial RxJS	 React Native tutorial React Native	 Python Design Patterns Python Design Patterns
 Python Pillow tutorial Python Pillow	 Python Turtle tutorial Python Turtle	 Keras tutorial Keras	


Preparation

 Aptitude Aptitude	 Logical Reasoning Reasoning	 Verbal Ability Verbal Ability	 Interview Questions Interview Questions
 Company Interview Questions Company Questions			


Trending Technologies




Artificial Intelligence Tutorial
Artificial Intelligence




AWS Tutorial
AWS




Selenium tutorial
Selenium




Cloud Computing tutorial
Cloud Computing




Hadoop tutorial
Hadoop




ReactJS Tutorial
ReactJS




Data Science Tutorial
Data Science




Angular 7 Tutorial
Angular 7




Blockchain Tutorial
Blockchain



Git Tutorial
Git



Machine Learning Tutorial
Machine Learning



DevOps Tutorial
DevOps

B.Tech / MCA



DBMS tutorial
DBMS



Data Structures tutorial
Data Structures



DAA tutorial
DAA



Operating System tutorial
Operating System



Computer Network tutorial
Computer Network



Compiler Design tutorial
Compiler Design



Computer Organization and Architecture
Computer Organization



Discrete Mathematics Tutorial
Discrete Mathematics



Ethical Hacking Tutorial
Ethical Hacking



Computer Graphics Tutorial
Computer Graphics



Software Engineering Tutorial
Software Engineering



html tutorial
Web Technology



C++ tutorial
C++



Automata Tutorial










C Language tutorial



C++ tutorial
C++

↑ SCROLL TO TOP

Cyber Security	Automata	C Programming	
 Java tutorial Java	 .Net Framework tutorial .Net	 Python tutorial Python	 List of Programs Programs
 Control Systems tutorial Control System	 Data Mining Tutorial Data Mining	 Data Warehouse Tutorial Data Warehouse	