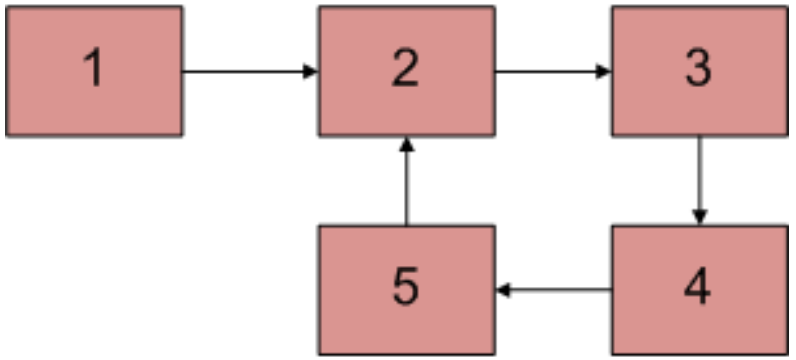
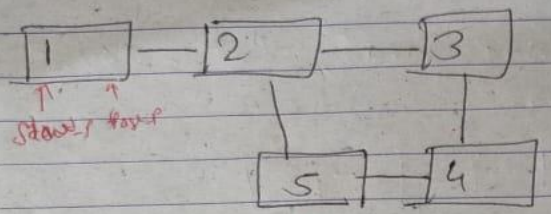
Find First Node In A loop

Step-1 Check Loop Exist or not
Step 2-if not exist then return null
Step-3 if exist then ,initialize a slow pointer to head ,fast ba at its position
Step -4 Move both pointer alow and fast one node at a time
Step 5- meeting point of both pointer is the start of the loop (i.e.first node in a loop)
Step 6:let’s enjoy

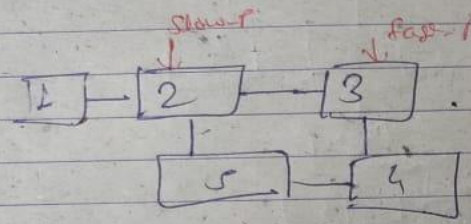


Loop checking

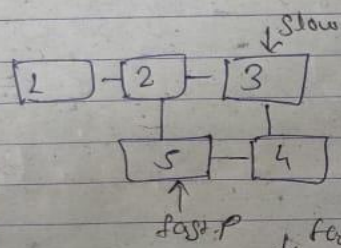
Step-1



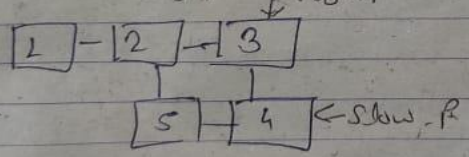
Step-2



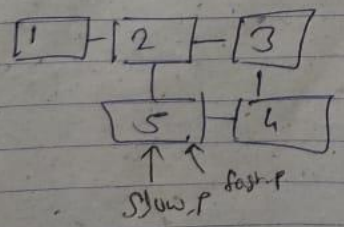
Step-3



Step-4



Step-5

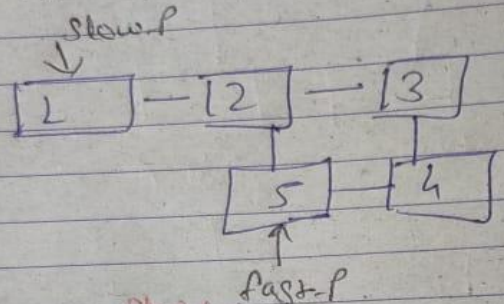


both Pointer ~~meet~~ point same node So loop exist.

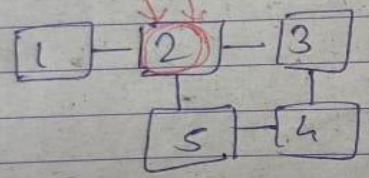
Step-2

now, slow_p = head, fast_p = fast_p

Step-1



Step-2



So, 2 is a first node in a linked list

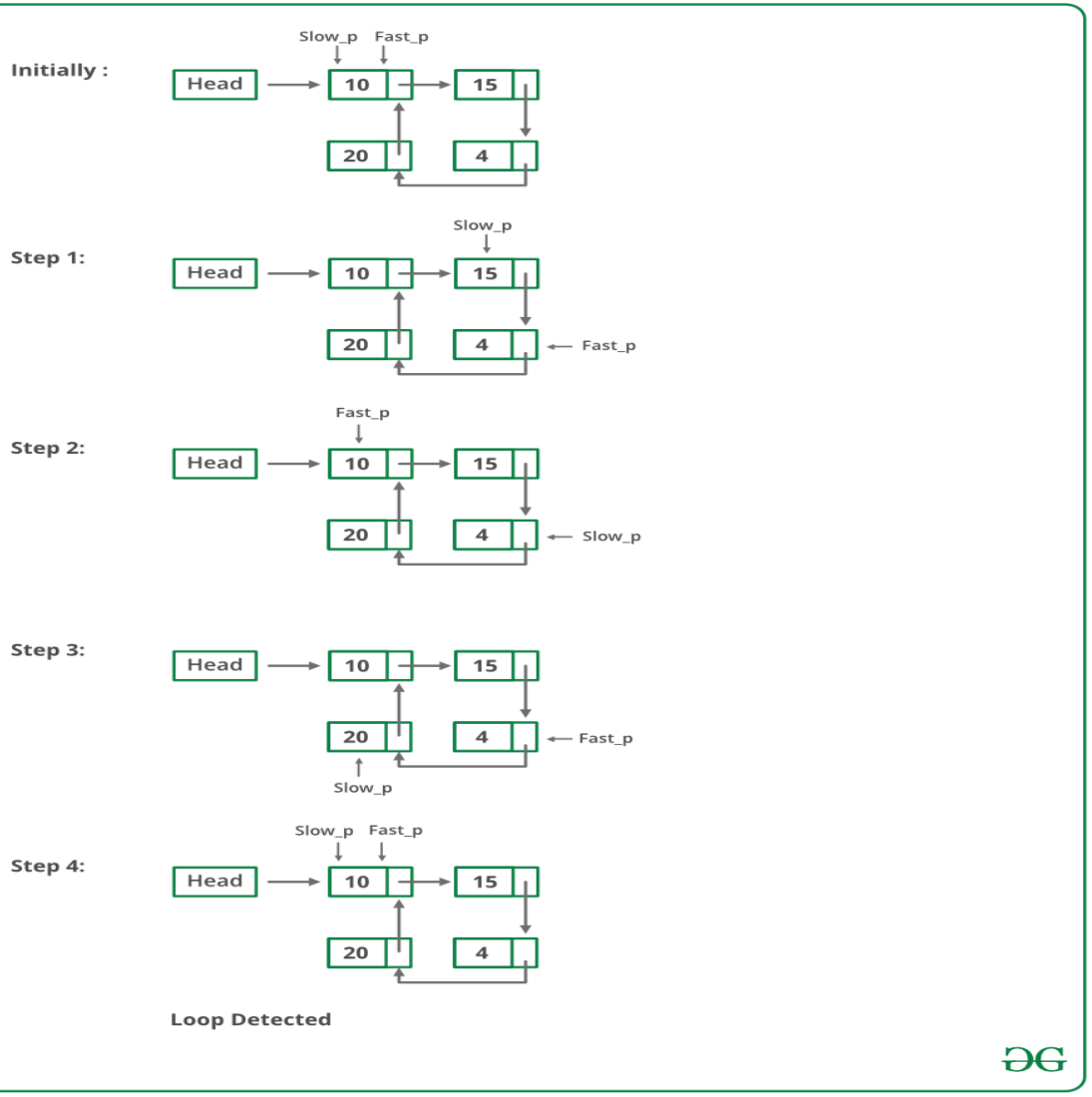
Check Loop Exist or not:

Floy’d Cycle-Finding Alogrithm-Fatsest Method

Step -1 : Traverse linked list using Two Pointers

Step-2: Move Fast pointer by two node at a time & slow pointer by one pointer at a time

Step 3: if they meet at the same node then loop exist otherwise loop not exist



Implementation :-

```
Node *findFirstNodeInLoop(Node *head){
      if(head==NULL||head->Next==NULL)
                return NULL;
       Node *slow_p=*fast_p=head;
       while(fast_p&&fast_p->next){
            slow_p=slow_p->next;
            fast_p=fast_p->next;
            if(slow_p==fast_p)
                  break;//LOOP EXIST
       }
      if(slow_p!=fast_p)return NULL;//LOOP NOT EXIST
          //if LOOP EXIST
       slow_p=head;
       while(slow_p!=fast_p){
            slow_p=slow_p->next;
             fast_p=fast_p->next;
        }
    return slow;//retrun fast;
}
```