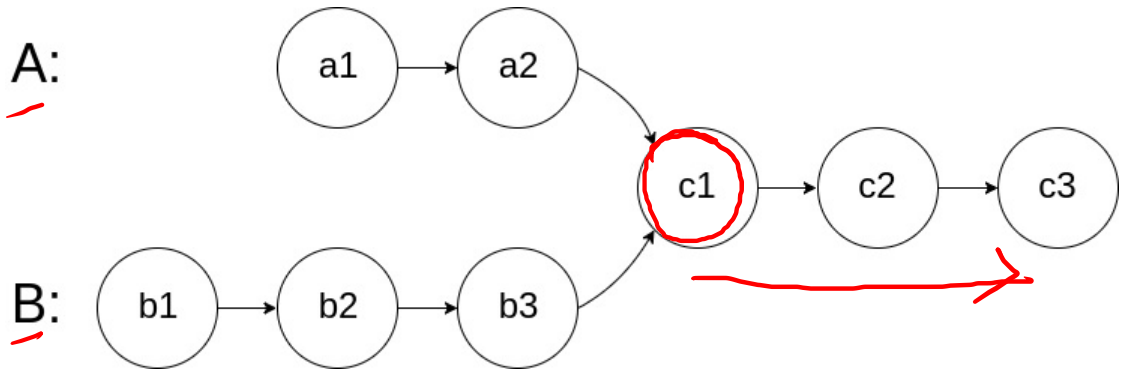
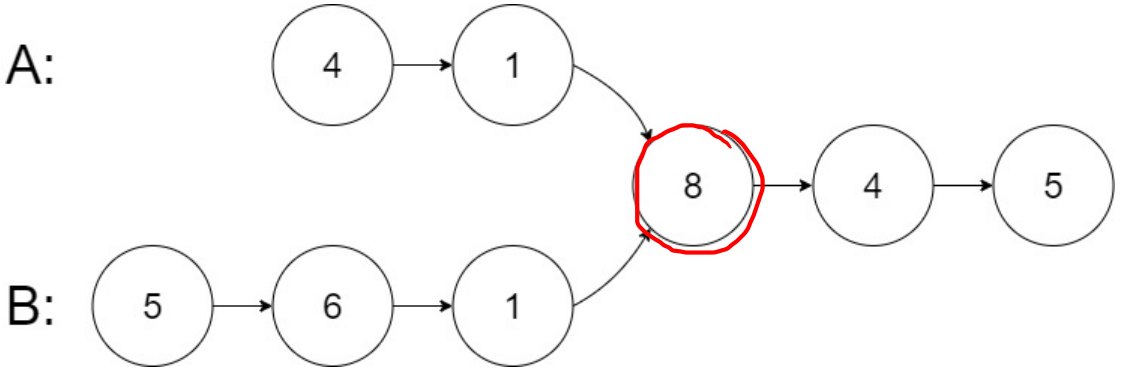
"Intersection of two linked list"

I/p:



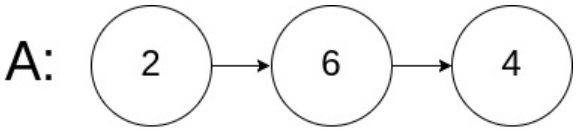
O/p: c1

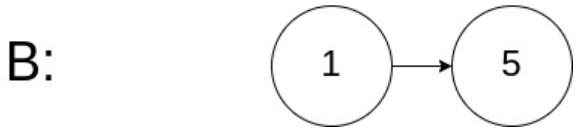
I/p:



o/p: 8

I/p:

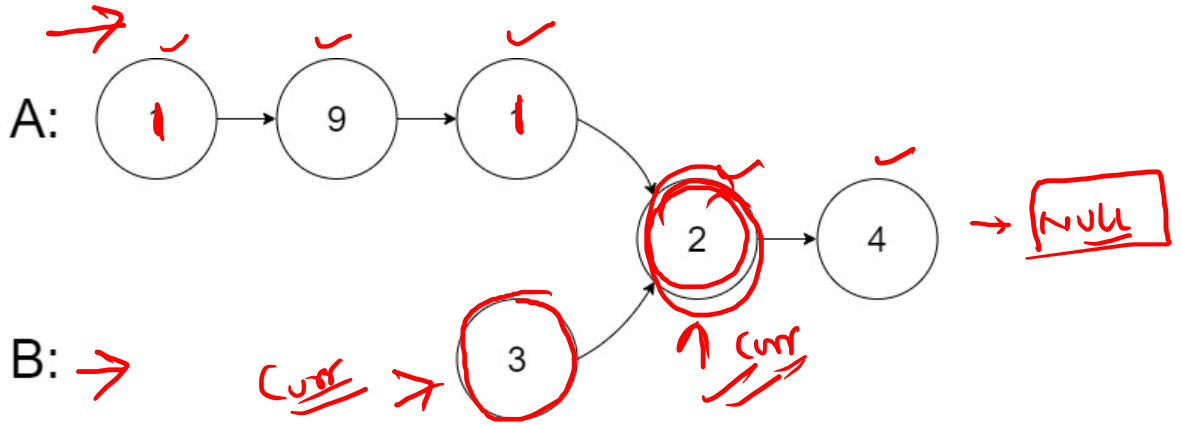




O/p:

NULL

NAIVE APPROACH



Val ✗

Node*

Soln :

1) create unordered_set <Node*> set ; // Hashing
2) traverse (A) → and store Node in set
3) traverse (B) → and Find the Curr Node is present in set or not?
4) if we found (set.find(curr) != set.end()) → return Curr;

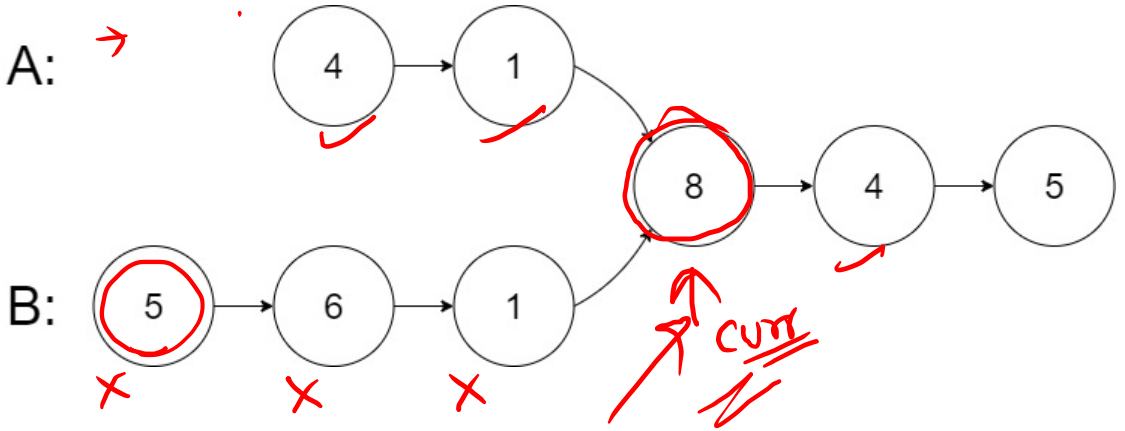
else

At the end Return NULL

O(1) { Space: O(M)

Time: O(N+M) }

(NAIVE)

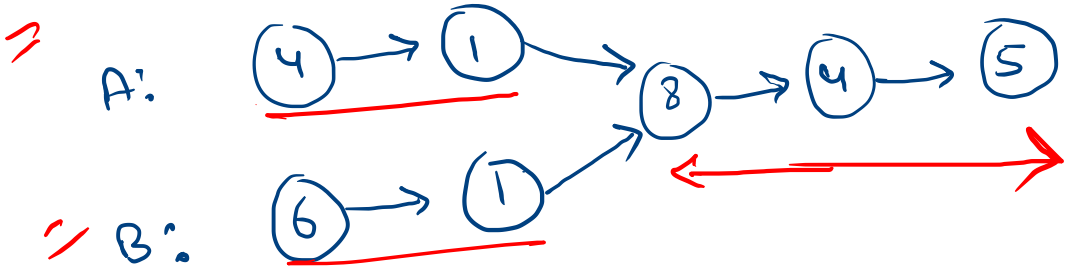


Soln:

set = { 4, 1, 8, 4, 5 } X

X 0 1 0 5 6 4 5

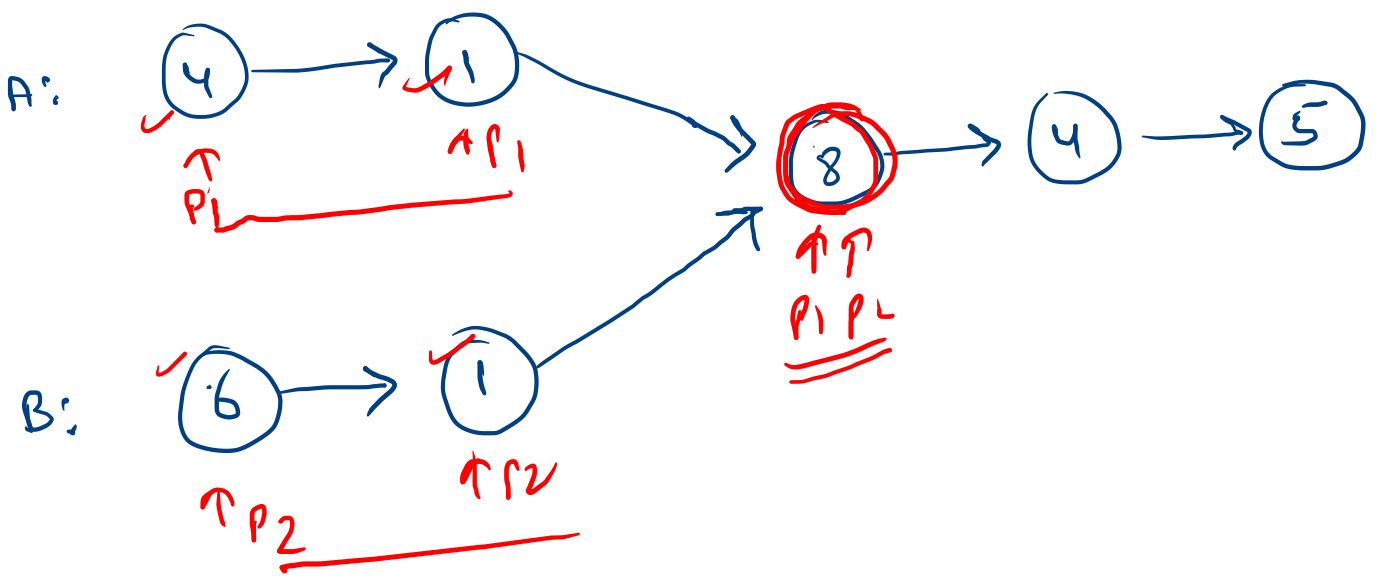
Better Approach :



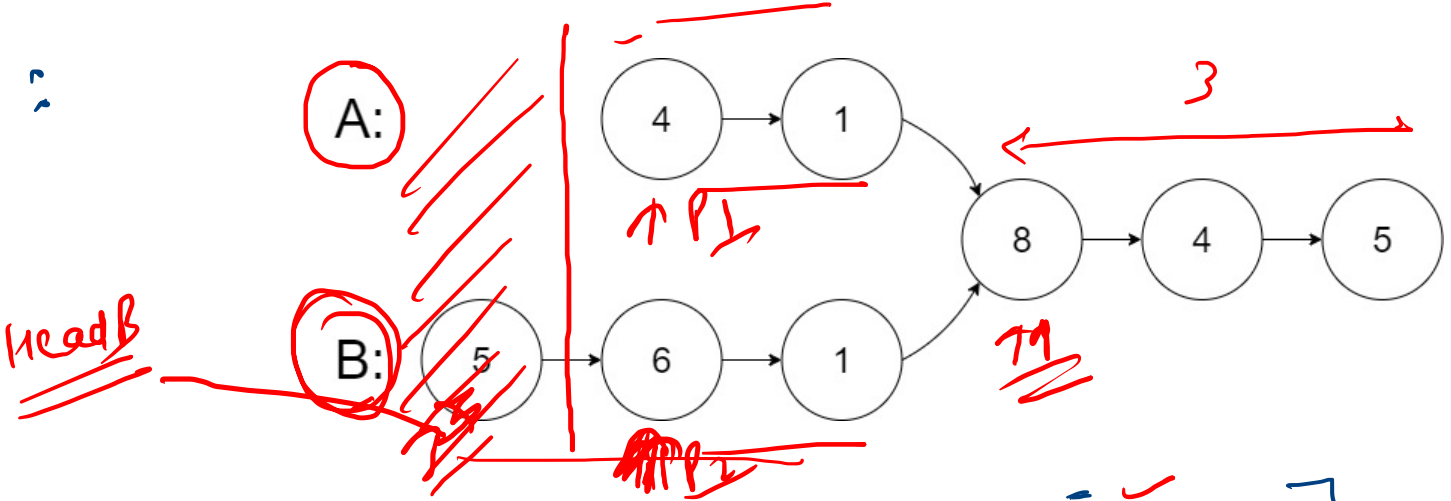
Soln:

P1 → Head A
P2 → Head B

→ Both Run at same speed



Better Approach :



Sol' :

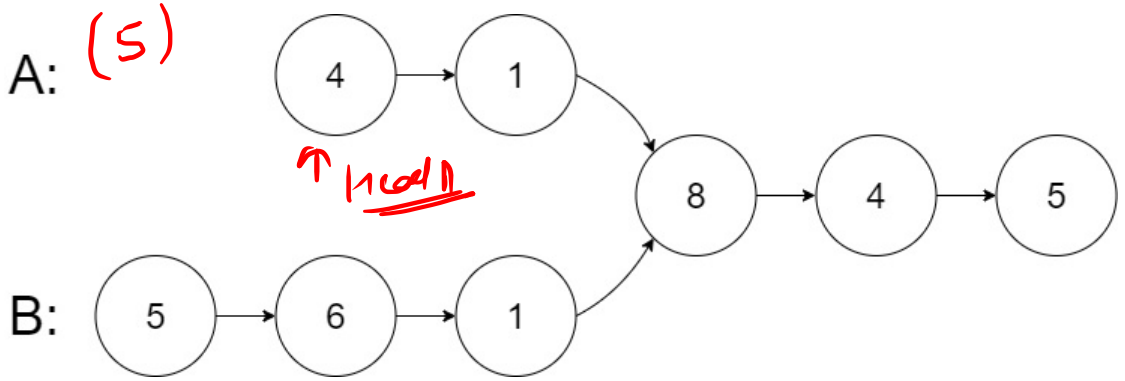
(1) Find count HeadA, a = 5

(2) Find count HeadB b = 6

→ (diff = 1)

⇒ Now, we can traverse HeadA and HeadB (Both at same speed)

```
while ( diff - - )
{
    HeadB = HeadB → next;
}
```



(5)

(6)

Head A

Head B

Sol^{n} :-

~~Head B~~

Head B

(Head A
Head B)