

Intersection of Two Linked Lists

Two paths that merge into one — find where they meet

● Linked List

■ EASY

Intuition First

Two pointers walk their respective lists. When one reaches the end, redirect it to the head of the OTHER list. Both pointers travel the same total distance ($\text{lenA} + \text{lenB}$). So they meet exactly at the intersection point — or both reach null together if there's none. Pure elegance!

```
List A:    a1 → a2 ■
              c1 → c2 → c3
List B: b1 → b2 → b3 ■

pA travels: a1,a2,c1,c2,c3,null → b1,b2,b3,c1...
pB travels: b1,b2,b3,c1,c2,c3,null → a1,a2,c1...
Both reach c1 at same time ✓

Total steps each: len(A) + len(B)
Both reach intersection after same number of steps
```

Java Code

```
1  public class Solution {
2  public ListNode getIntersectionNode(ListNode headA, ListNode headB) {
3      ListNode a = headA, b = headB;
4
5      // They will meet at intersection or both reach null
6      while (a != b) {
7          // When a reaches end, redirect to headB
8          a = (a == null) ? headB : a.next;
9          // When b reaches end, redirect to headA
10         b = (b == null) ? headA : b.next;
11     }
12     return a; // null if no intersection
13 }
14 }
```

Complexity

Type	Value	Meaning
Time	$O(m+n)$	Both pointers travel at most $m+n$ steps

Space	$O(1)$	Only two pointers
-------	--------	-------------------

■ Why it works

Both pointers travel $\text{lenA} + \text{lenB}$ total steps. The intersection (if any) is at position $\text{lenA} + \text{lenB}$ -shared from the start for both paths. So they sync up perfectly!

■ Quick Revision

✓	Two pointers $a = \text{headA}$, $b = \text{headB}$
✓	When $a == \text{null}$, redirect to headB. When $b == \text{null}$, redirect to headA
✓	They meet at intersection after $\text{lenA} + \text{lenB}$ steps
✓	If no intersection, both become null simultaneously
✓	Time $O(m+n)$, Space $O(1)$