Linked list intersect

Ah, you're talking about skipA and skipB from the other approach (the length-difference method).
Let me explain super simply.

✅ What is skipA and skipB?

When you have two linked lists of different lengths,
you can calculate how many extra nodes are in one list before the intersection.

- skipA = how many nodes List A has before the intersection
- skipB = how many nodes List B has before the intersection

✅ Example

```less
List A: 4  1  8  4  5
List B:  5  6  1  8  4  5
```

Let's count:

- List A length = 5
- List B length = 6
- Common part (after intersection at 8) = 3 nodes (`8  4  5`)

So:

- skipA = 5 - 3 = 2 nodes (`4  1`)
- skipB = 6 - 3 = 3 nodes (`5  6  1`)

So before the intersection,

- List A has 2 extra nodes
- List B has 3 extra nodes

✅ How do we use skipA and skipB?

1. Find lengths `lenA` and `lenB`
2. Calculate difference `d = |lenA - lenB|`
3. Move the longer list pointer ahead by `d` steps

- If List B is longer by 1, move pB 1 step forward first

4. Then move both pointers together until they meet

✅ Applying to this example

- lenA = 5
- lenB = 6
- difference = 1

So move pB 1 step ahead first (skip `5` in List B).

Now both have equal remaining length →

```nginx
pA  4
pB  6
```

Move step by step → they will meet at 8.

✅ TL;DR

- skipA = number of extra nodes in List A before intersection
- skipB = number of extra nodes in List B before intersection
- Helps us align both lists to the same remaining length before walking together.

Would you like me to:

✅ Draw a simple diagram showing skipA/skipB visually?
✅ Or explain again using a short story?