Notes on Assignment 4

CSCI 2270: Data Structures
Section: 202
TA: Sanskar Katiyar

In this assignment, you have 5 functions to complete:

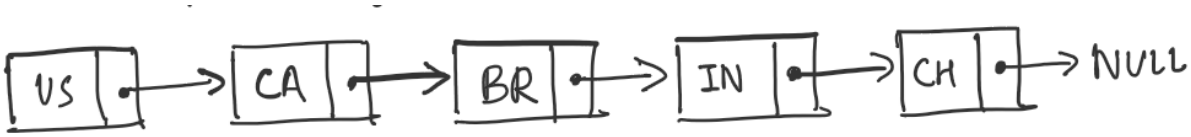
`void deleteEntireNetwork();`
`void deleteCountry(string countryName);`
`bool detectLoop();`
`Country* createLoop(string countryName);`
`void readjustNetwork(int start, int end);`

For the first two functions you should refer the slides, code examples from Recitation 4, specifically the "Deletion in a linked list" section. They are straightforward.

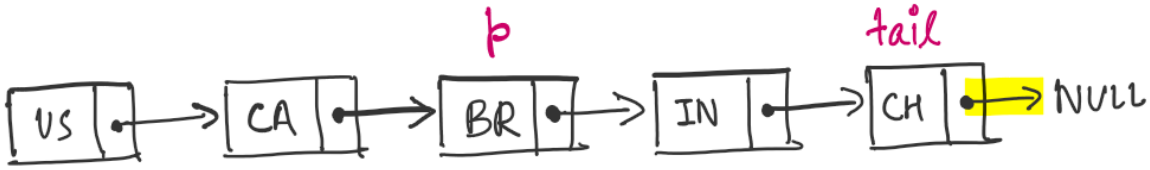
For the other 3 functions, I will try to give you some guiding questions so you can understand the problem better and approach the solution yourself.

createLoop

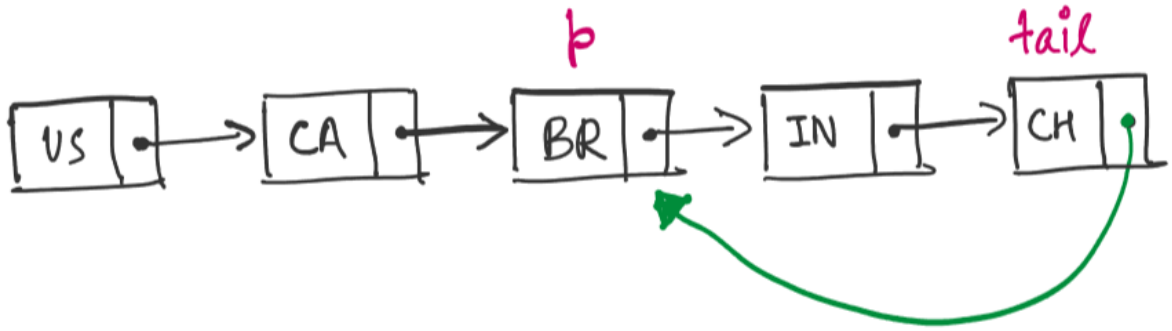
Let's say we call `createLoop("Br");`



Now, first you need to find the pointer to the node to "Br". You already have a function implemented in the class which does this – ***which one?***



You will also need to find the tail pointer of the list. ***How will you find this?***



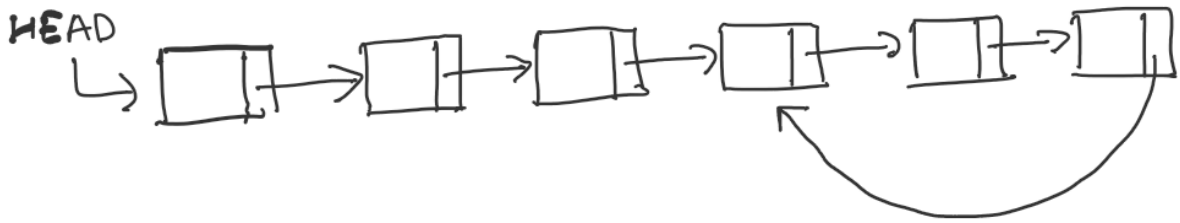
If you have found both the pointers p and tail, then you just need to link tail to p in order to create that cycle. It's one statement.

Remember to return the tail pointer!

detectLoop

The two videos mentioned in the writeup should be enough. This is a very popular algorithm and often asked in programming interviews. This approach is not obvious.

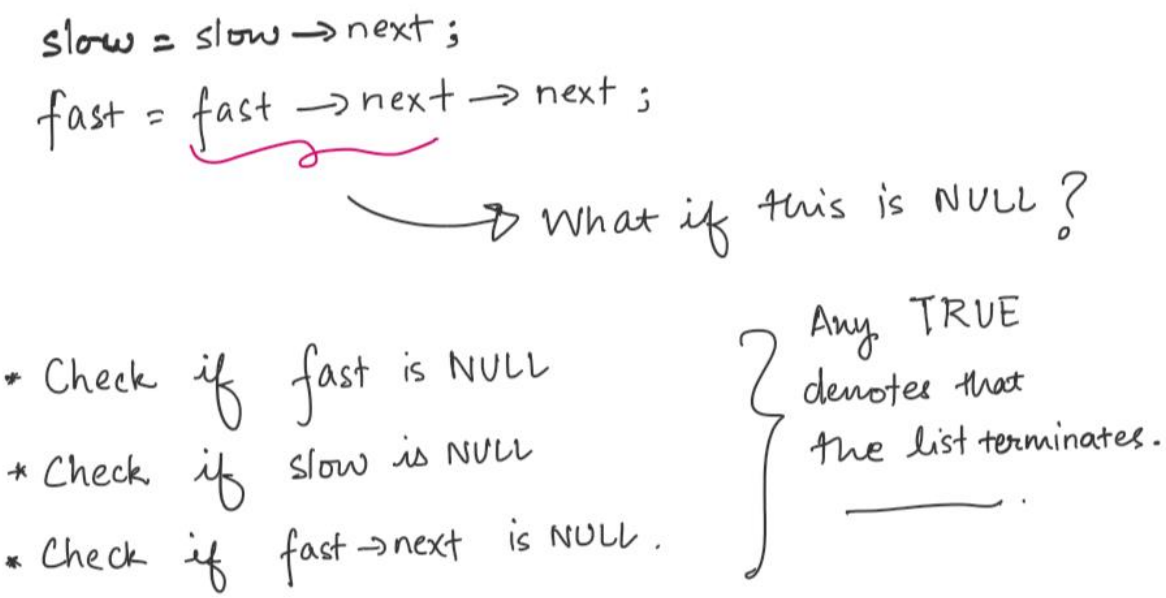
You are given a Linked list, which may or may not have a loop. You are required to implement a method which tells us whether there exists a loop in the given LL.



We maintain two pointers (fast, slow) and traverse the linked list until the `fast` and `slow` pointer are not NULL, as well as `fast->next` is not NULL.

If any of those three pointers are NULL, then that means we have encountered the end of the list, which is only possible if the list doesn't contain a loop. Thus, return False.

However, the condition for the cycle to exist is if fast == slow. Because this condition will never be true, unless there is a cycle in the list. If you are not convinced, think of it this way: ***in a list <u>without</u> a loop, there is no way slow == fast since you can never return to a previous node in the list***.

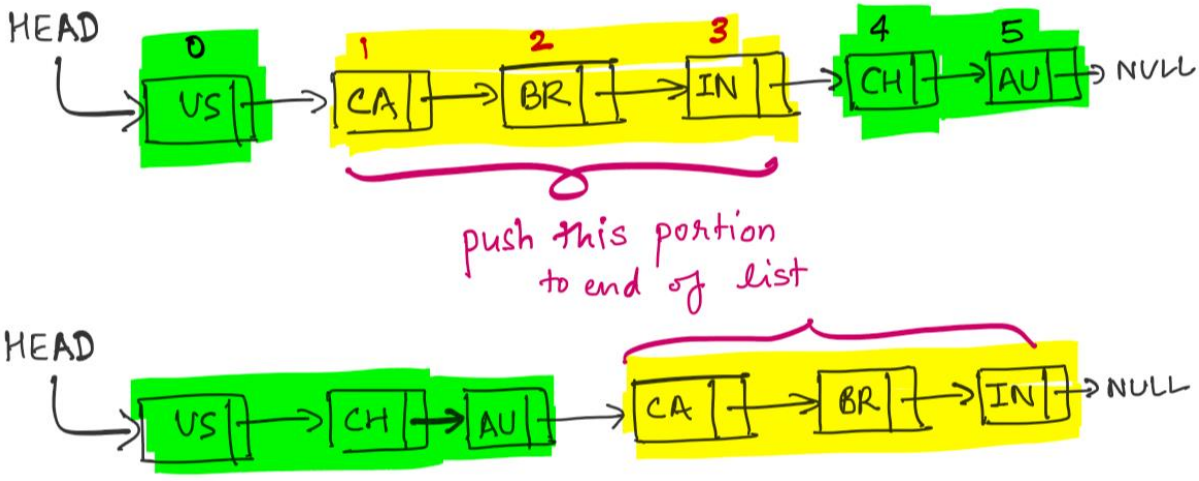


readjustNetwork

This is not an easy problem. I'd suggest breaking the problem down into pieces.

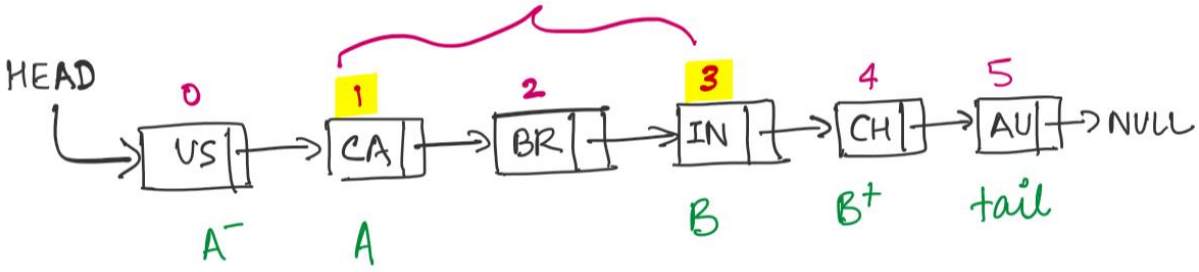
First, see if you understand what the problem is asking you to do.

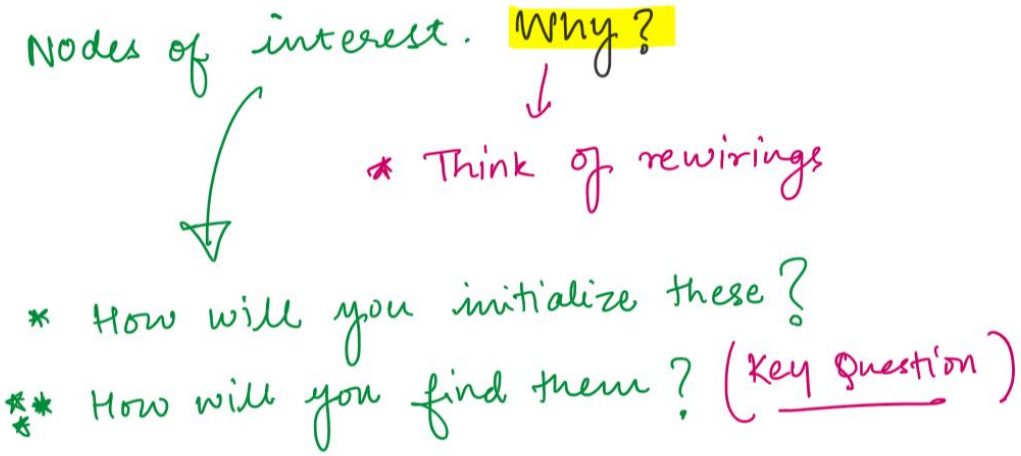
startIndex = 1 , endIndex = 3



Here are the **guiding questions** to solve this problem:

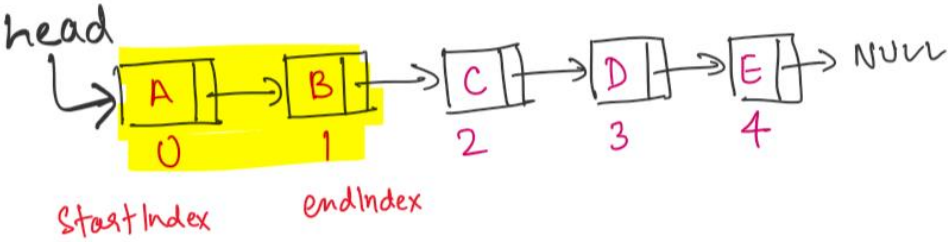
- How to find number of nodes in the list?
- How to find the tail node of the list?
- What are the nodes of interest in the list? (Check rewirings)
- What are the minimum number of nodes in a valid list, here?
- Think about these edge cases:
    - If start_index == 0
    - If start_index == end_index





* Check the minimum #nodes in a valid list. (i.e. if all index conditions are satisfied)

* Edge case: startIndex == 0



* Head got modified!!