USA  Tutor

The Bovine Shuffle

Analysis by David Yang

Statement Summary

We are given N<=100,000 cows

We are also given where each cow “goes to”

Find the number of positions where a cow will always be

Random Ideas

Simulate: N^2

When do we stop our simulation?

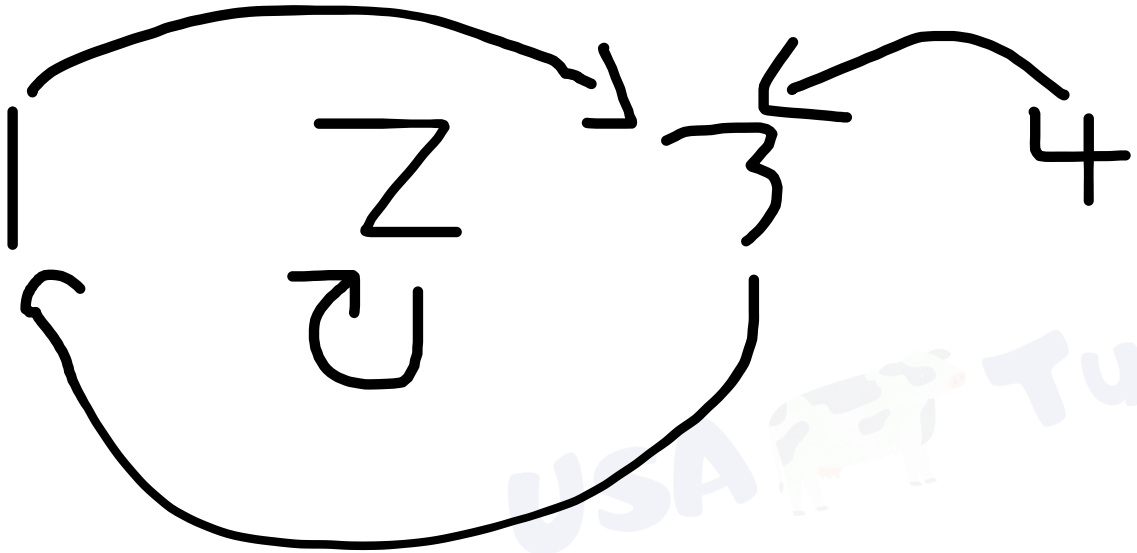
Do we stop at a cow? Stop at a time? How do you know?

Full credit: N?

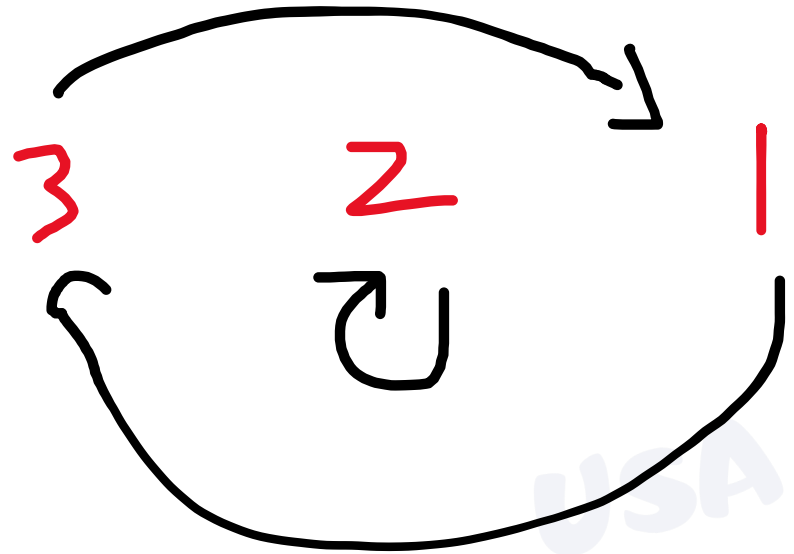
We can realize that *maybe* some spots will always have a cow?

Where do we find those spots?

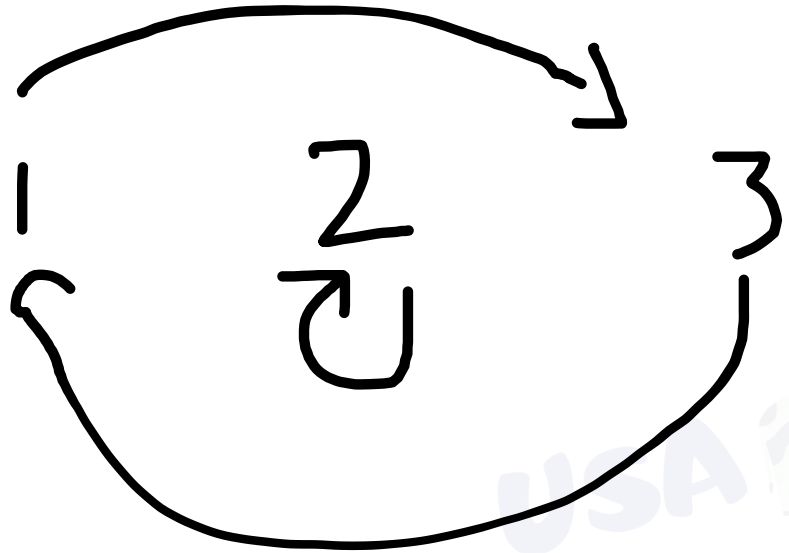
Sample



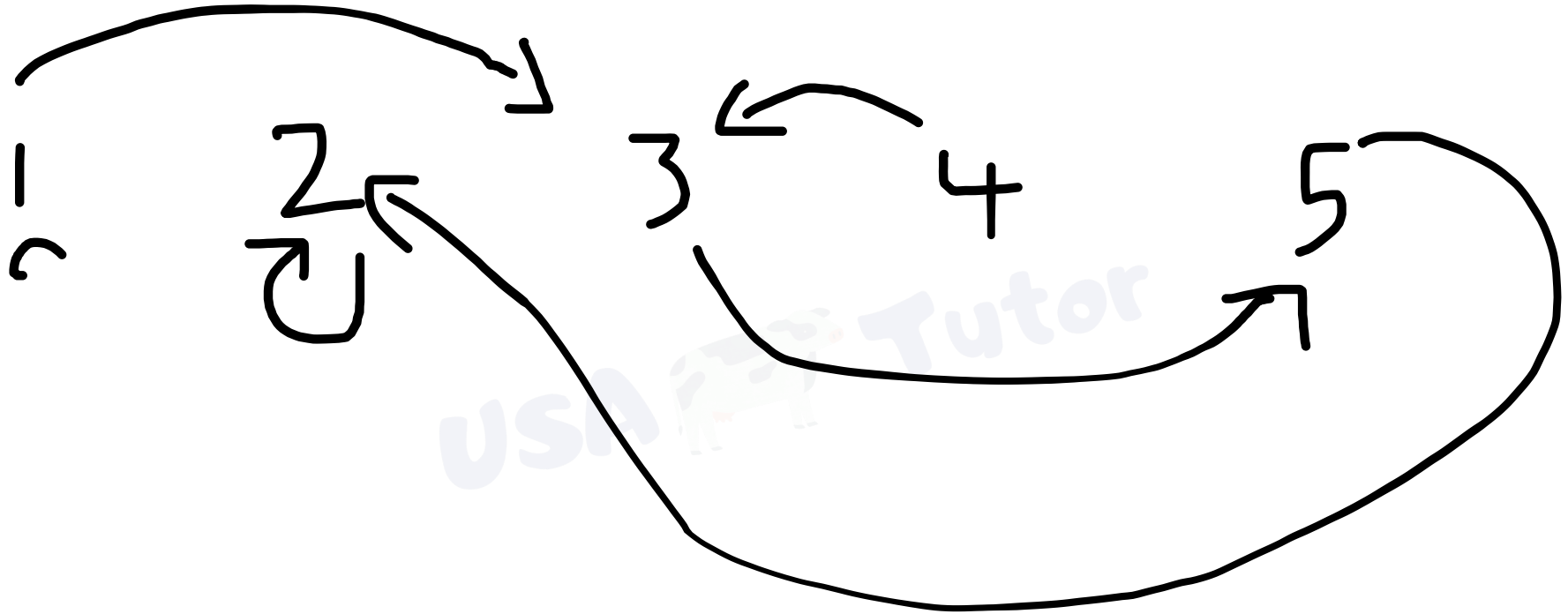
Sample



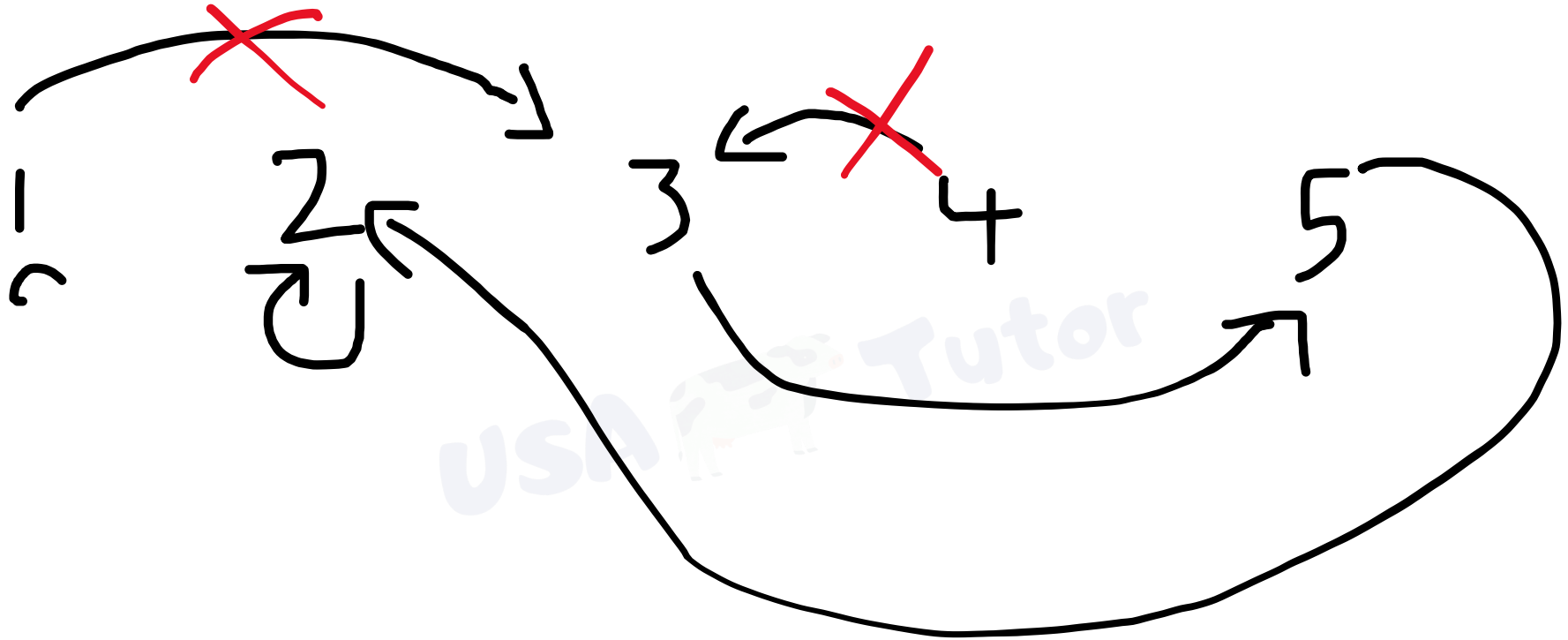
Sample



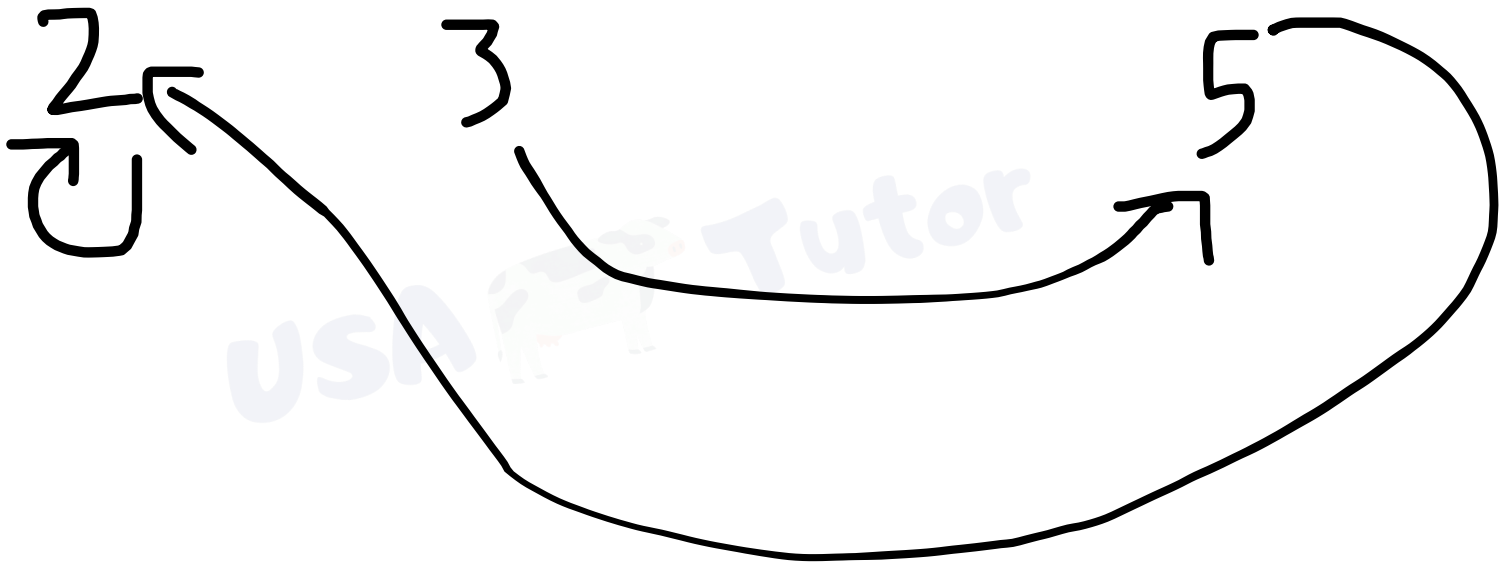
Maybe a better sample



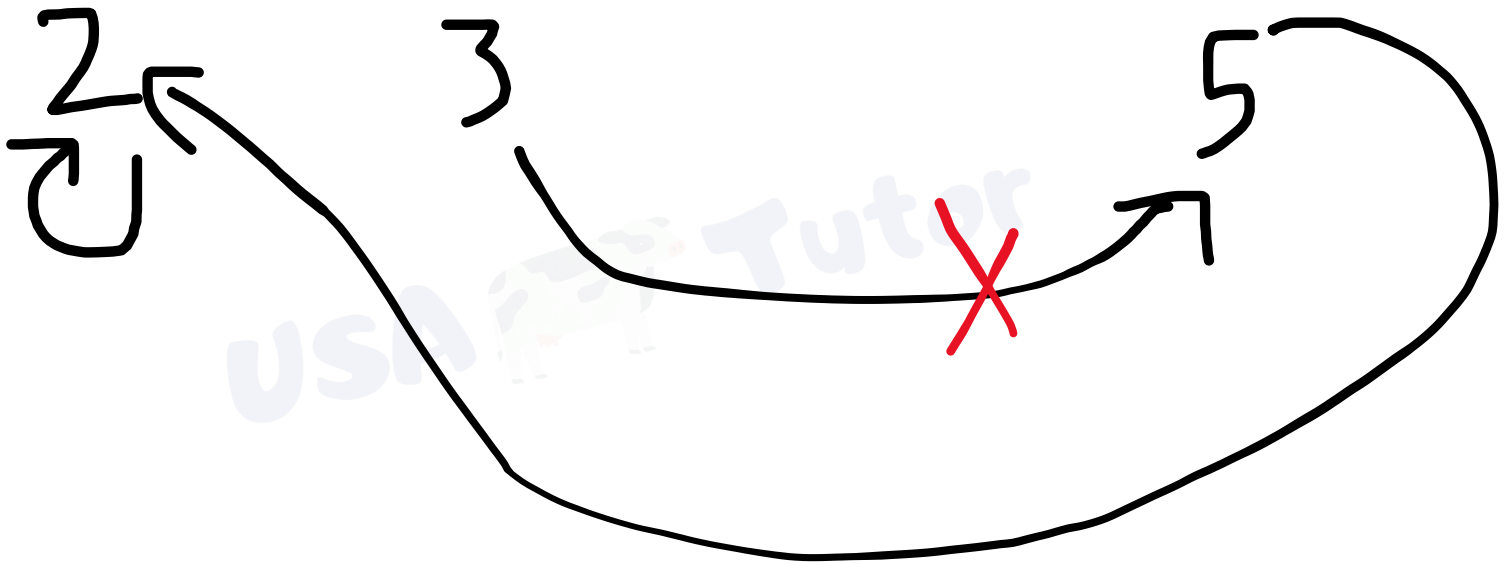
Maybe a better sample



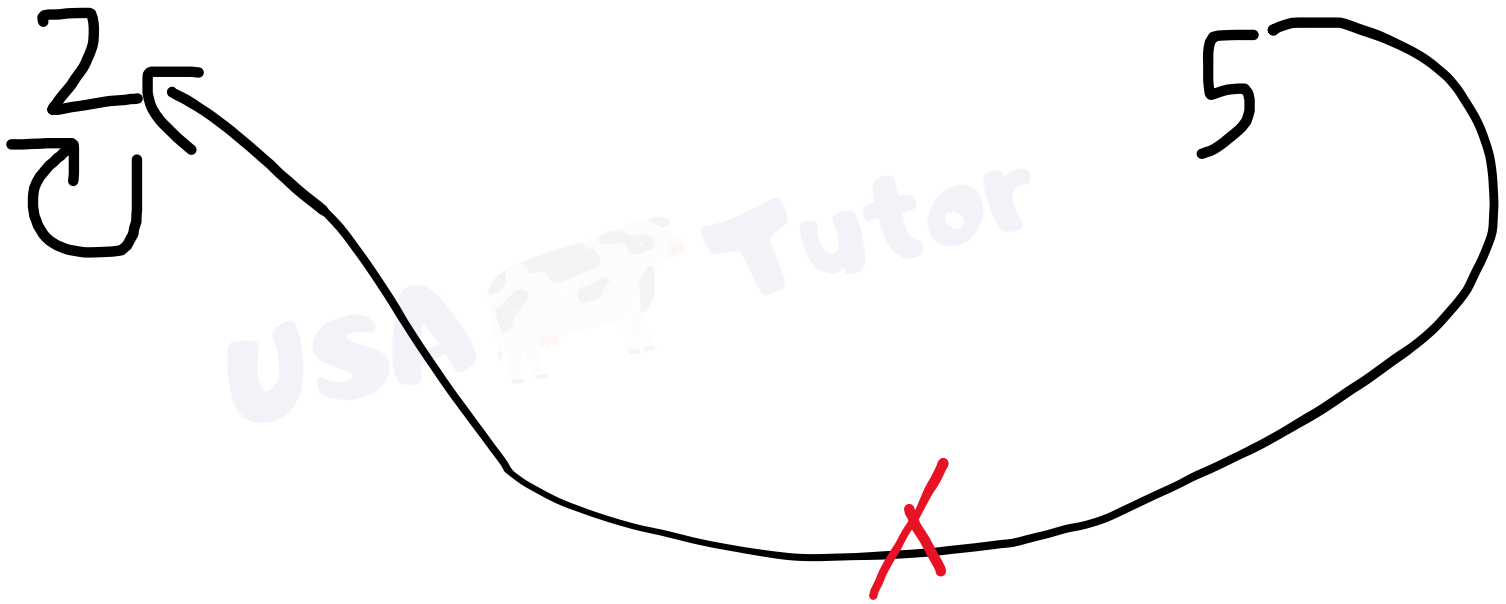
Maybe a better sample



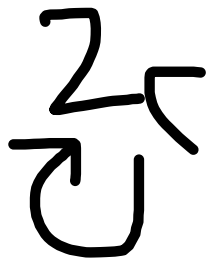
Maybe a better sample



Maybe a better sample



Maybe a better sample



Realizations?

Certain cows get “sucked in”

There are *cycles*

A cycle guarantees there will be cows in that cycle, and there will always be a cow in that cycle.

Self-cycles are allowed.

Realizations

If you are in a cycle, you will be a cemented position

If not you will not be one

How to code?

Coding

Let’s assume we’re NOT in a cycle:

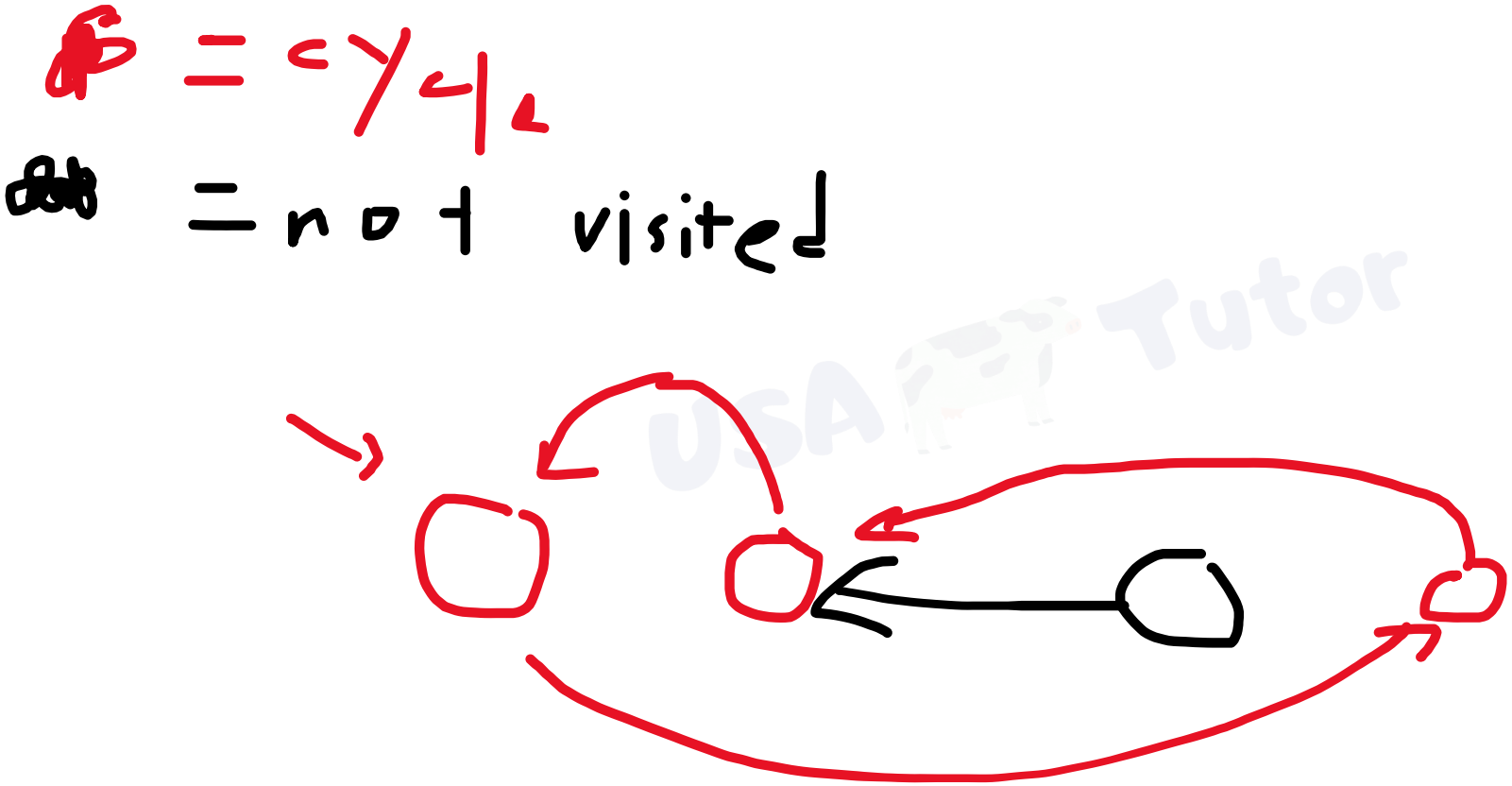
Mark as not cycle

This node does not contribute to answer

The Cycle Case:

How do you even find you are in a cycle?

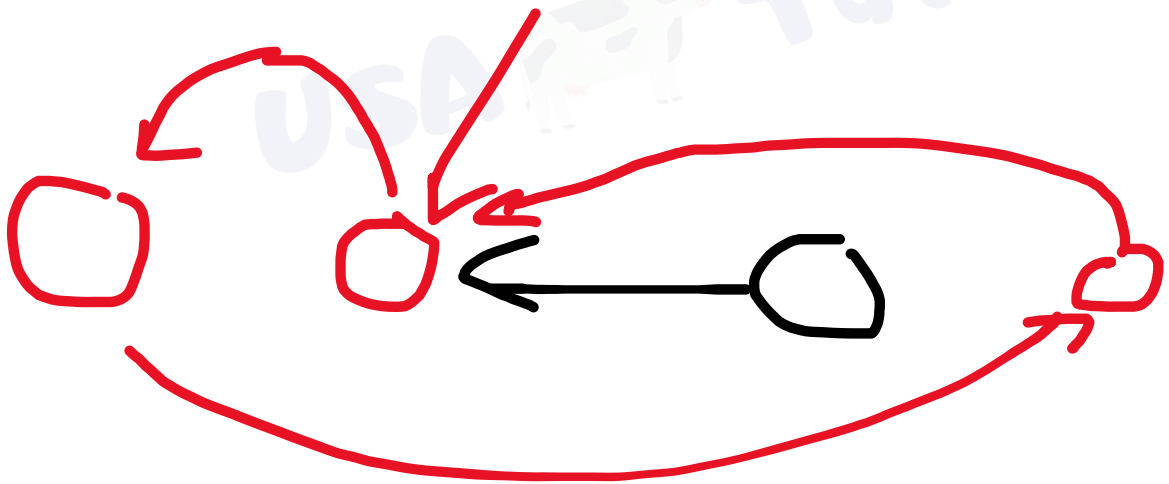
How not in cycle



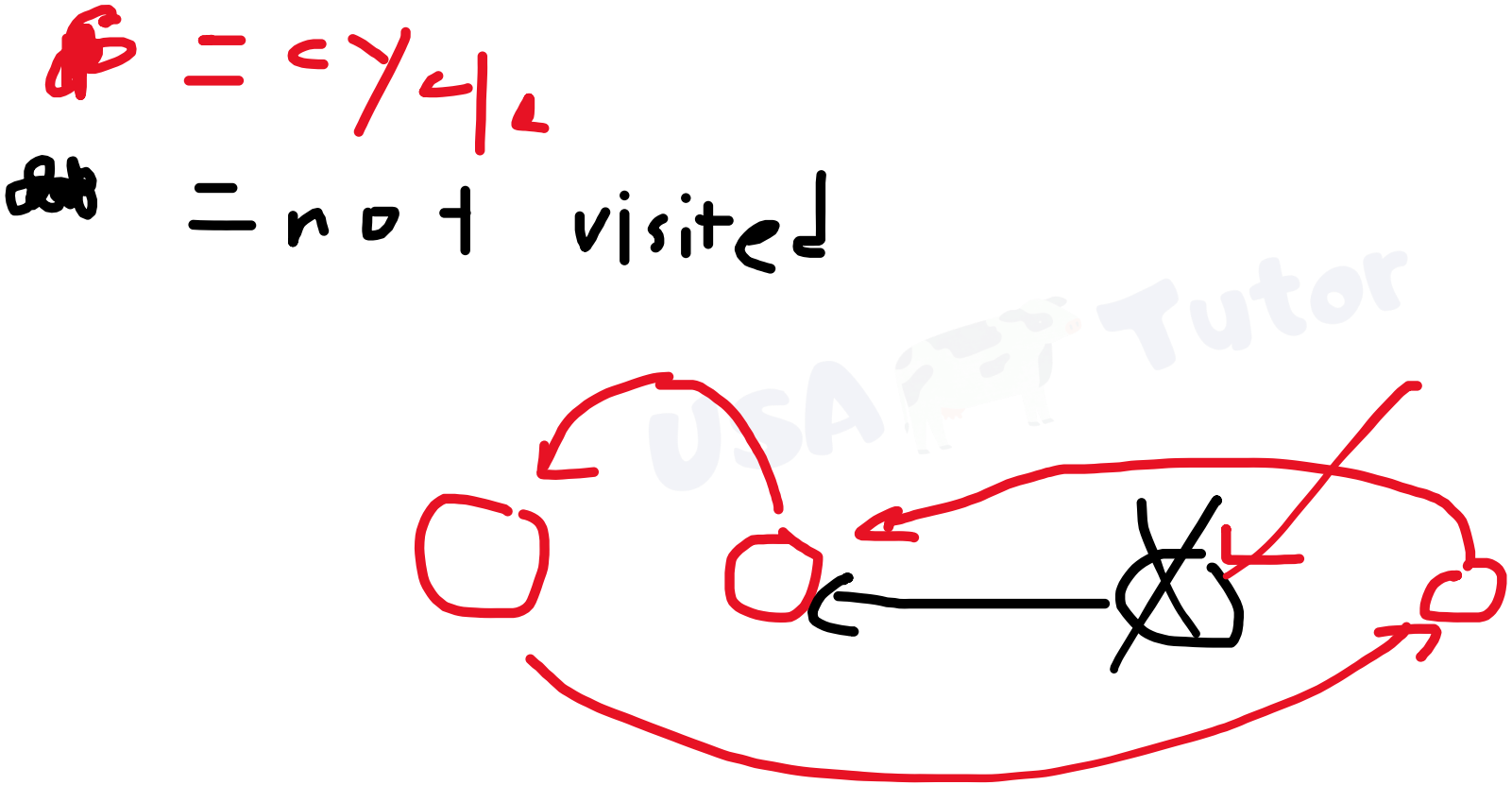
How not in cycle

= cycle

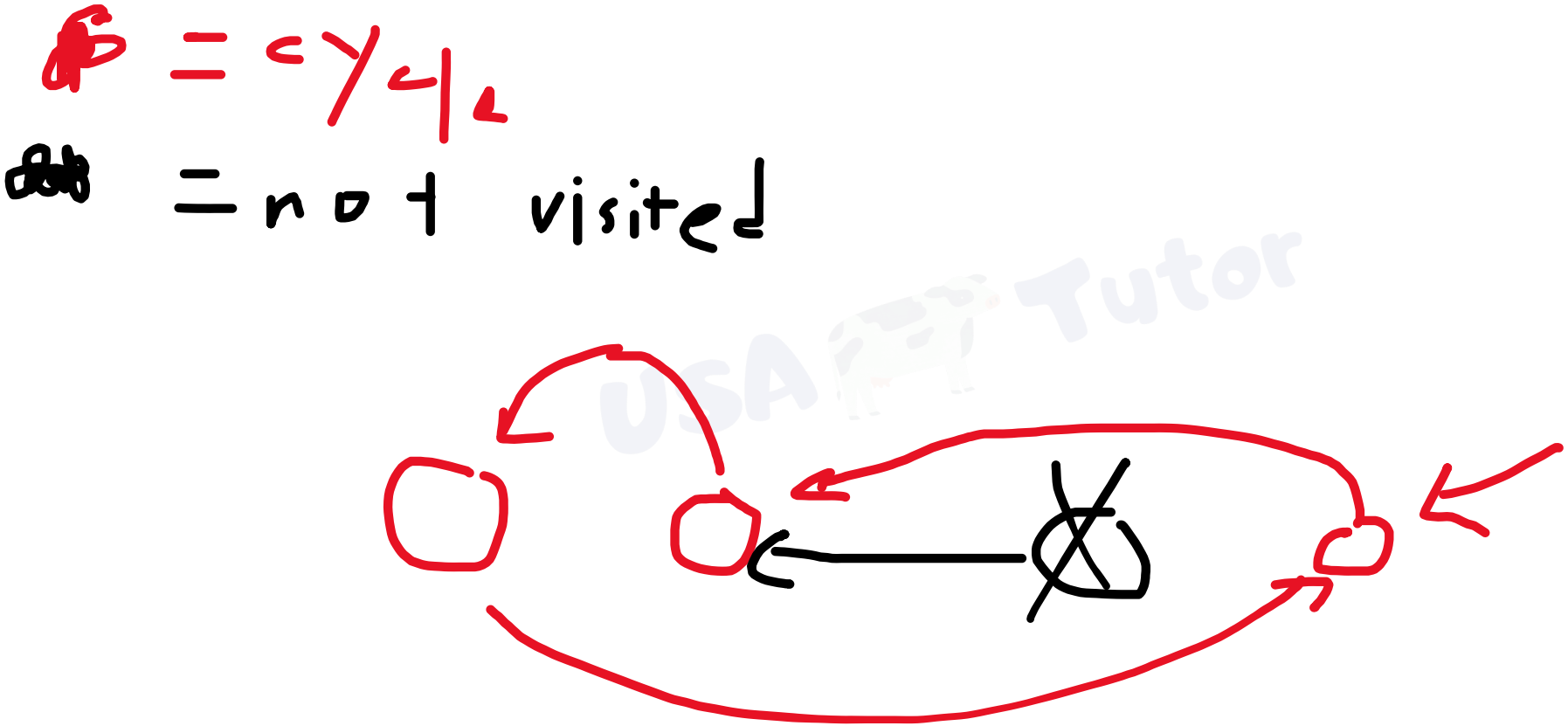
= not visited



How not in cycle



How not in cycle

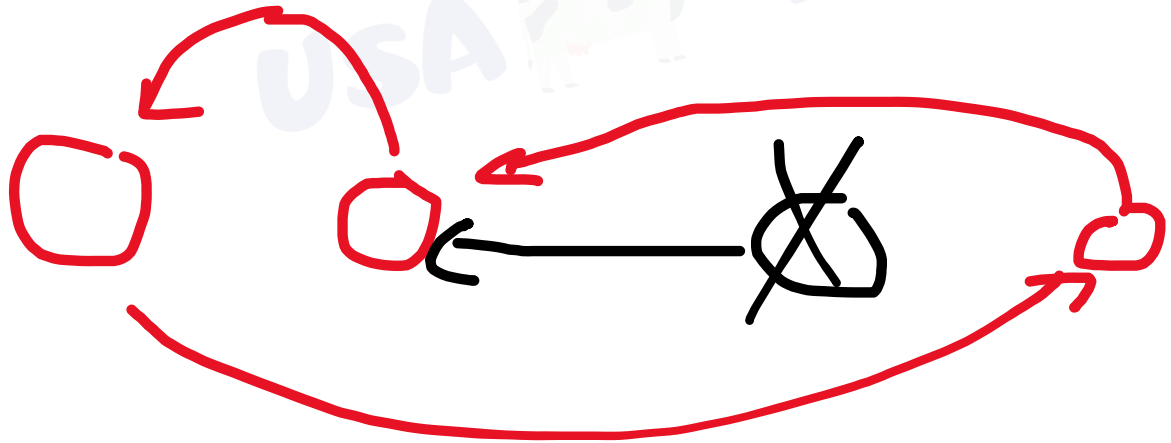


How not in cycle

= cycle

= not visited

gn5 = 3



Coding

The Cycle Case:

How do you even find you are in a cycle?

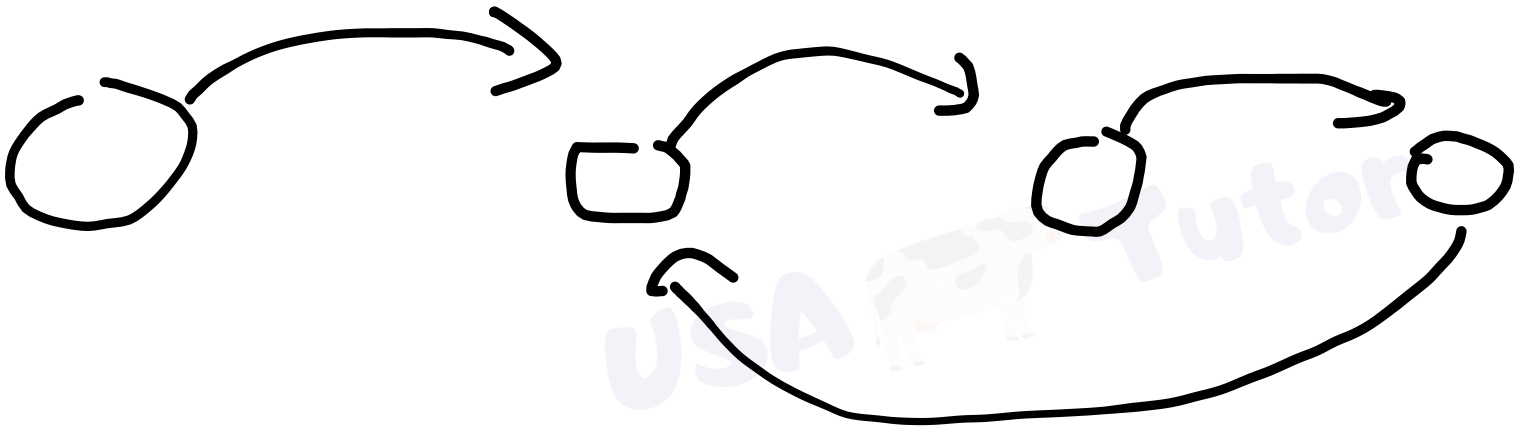
While loop, take 2 pointers (like you did for swapity swap)

Find which nodes are in the cycle and which ones are not

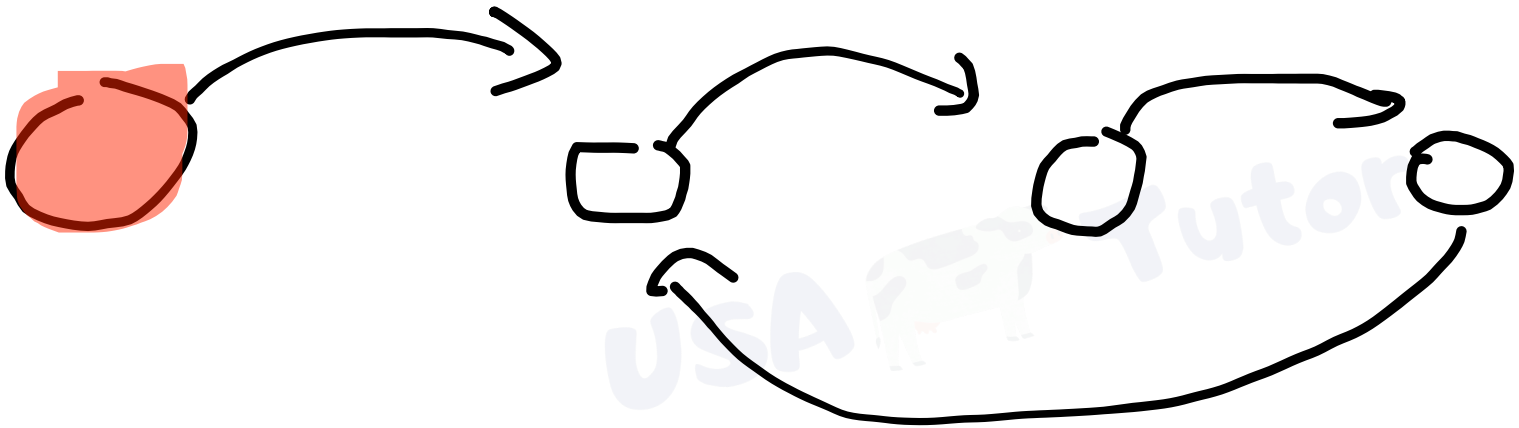
Mark them with their respective values

(0 for not visited, -1 for bad, 1 for good)

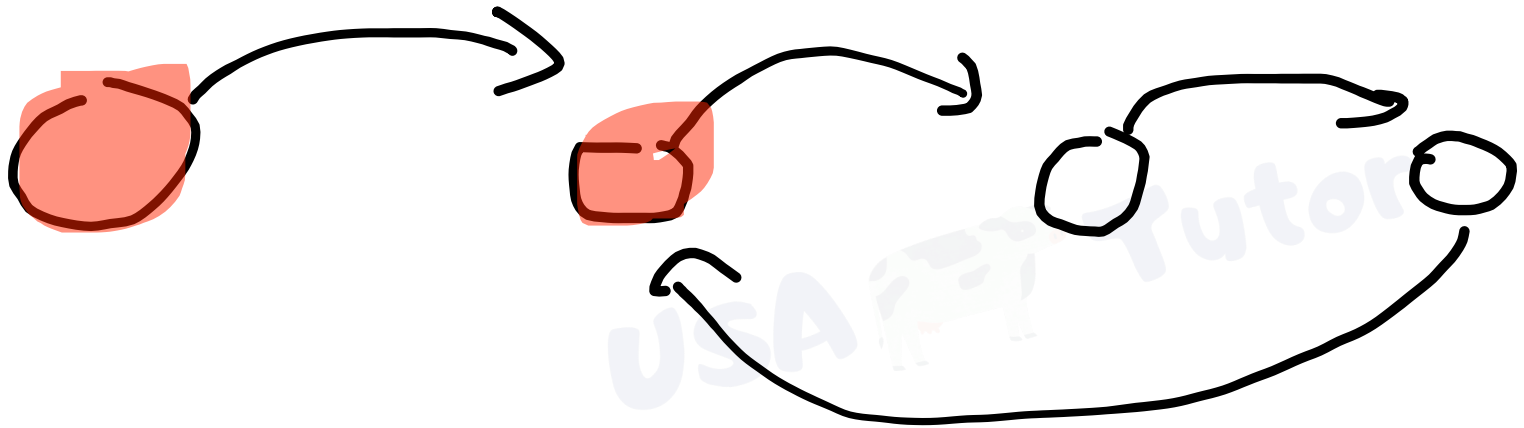
Sample that will help coding



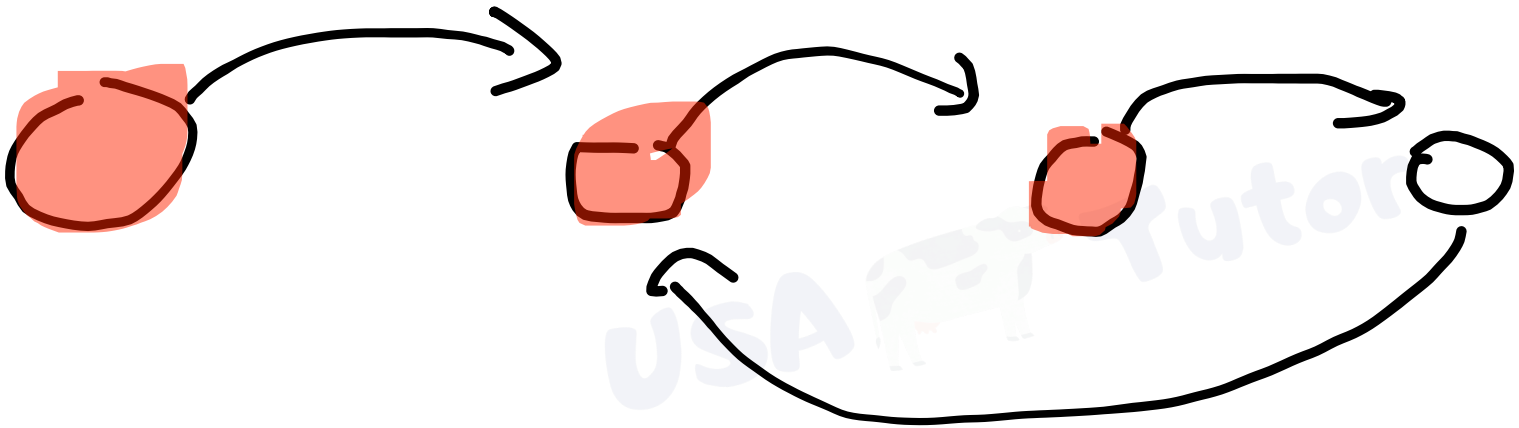
Sample that will help coding



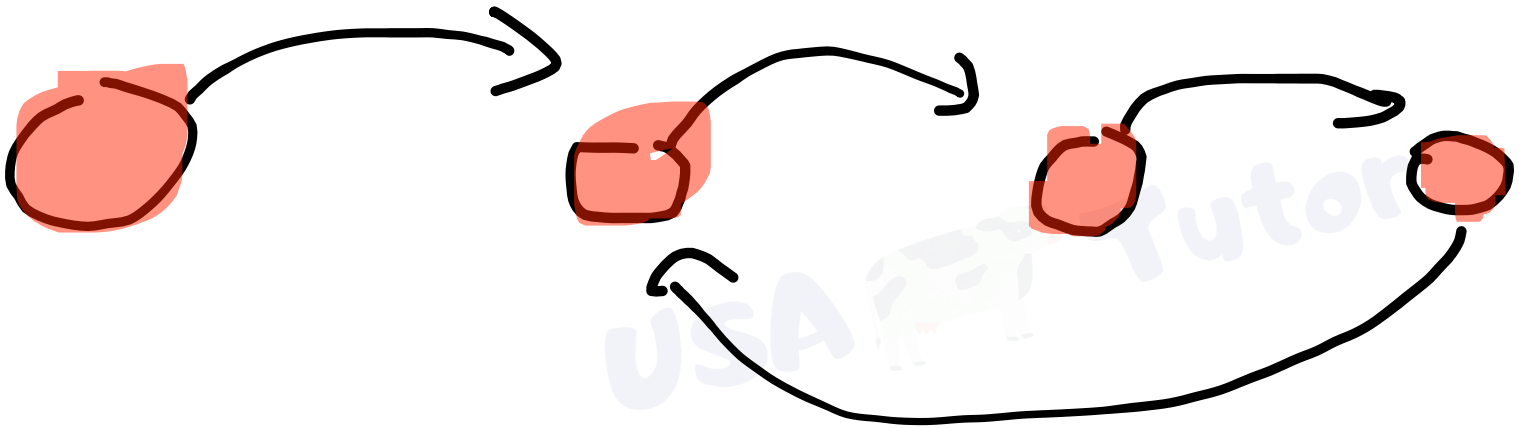
Sample that will help coding



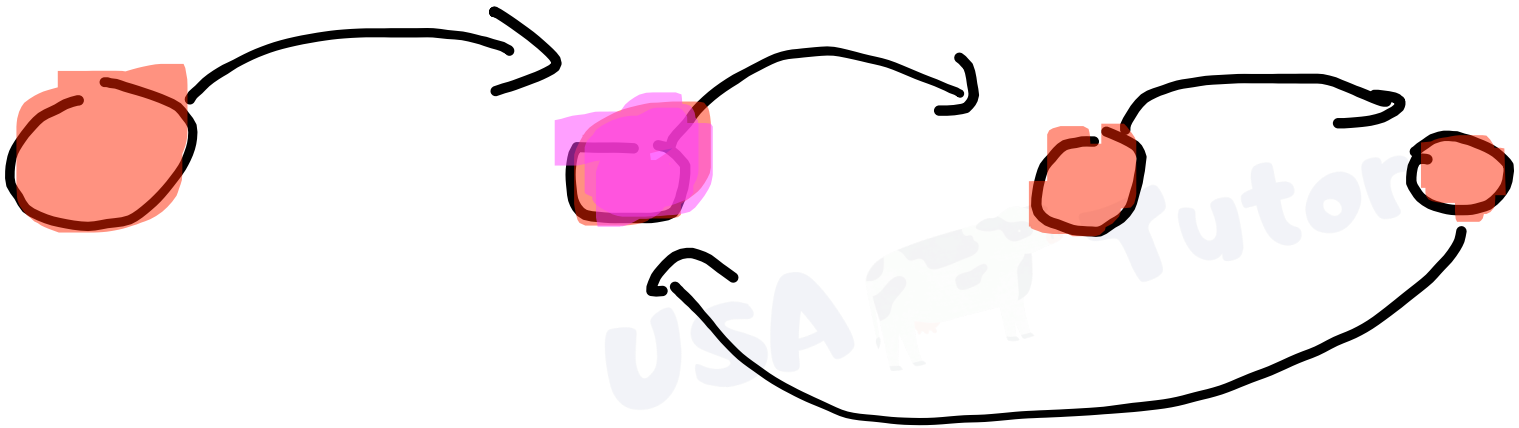
Sample that will help coding



Sample that will help coding



Sample that will help coding



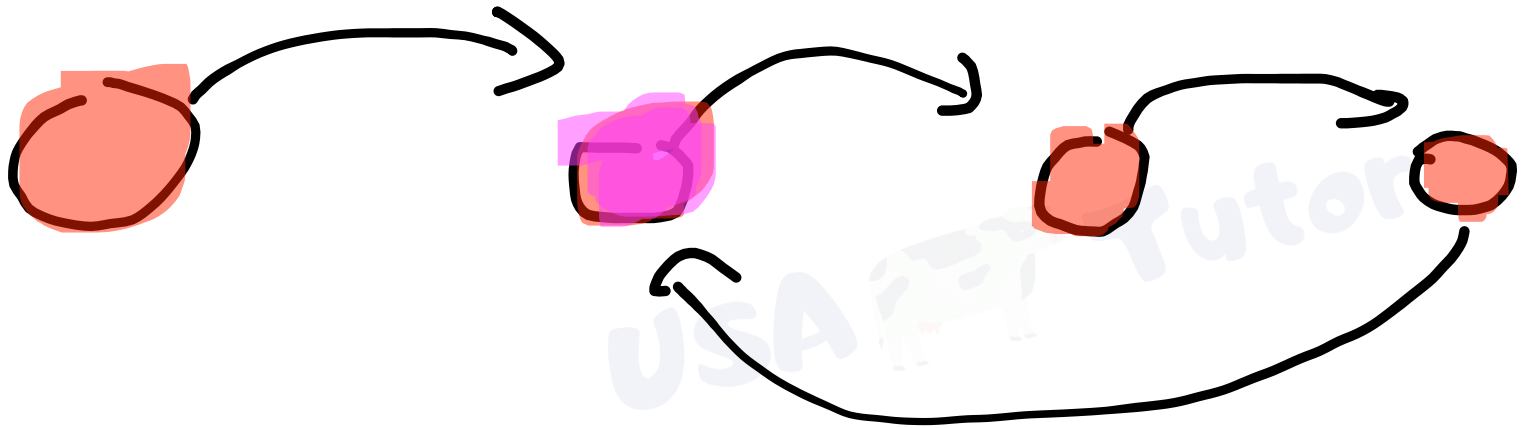
Implementing this “backtrack”

Store the nodes in a traversal into an ArrayDeque or a Stack

An ArrayDeque is a structure that can remove/add nodes from the front and/or the back of the structure in O(1) time.

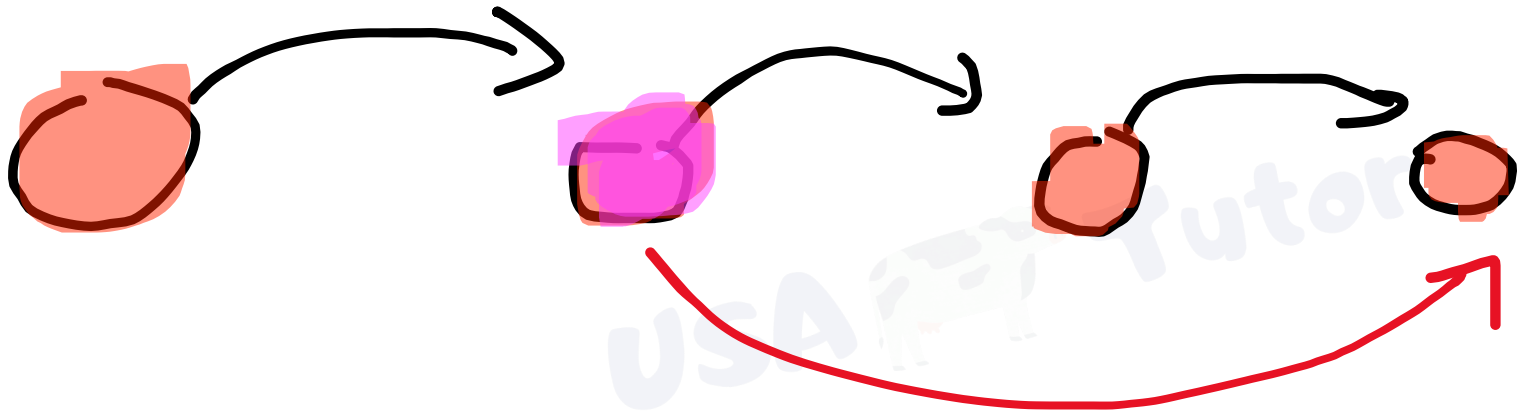
You will pop off every node UNTIL you reach the doubly visited node (highlighted in pink)

Sample that will help coding



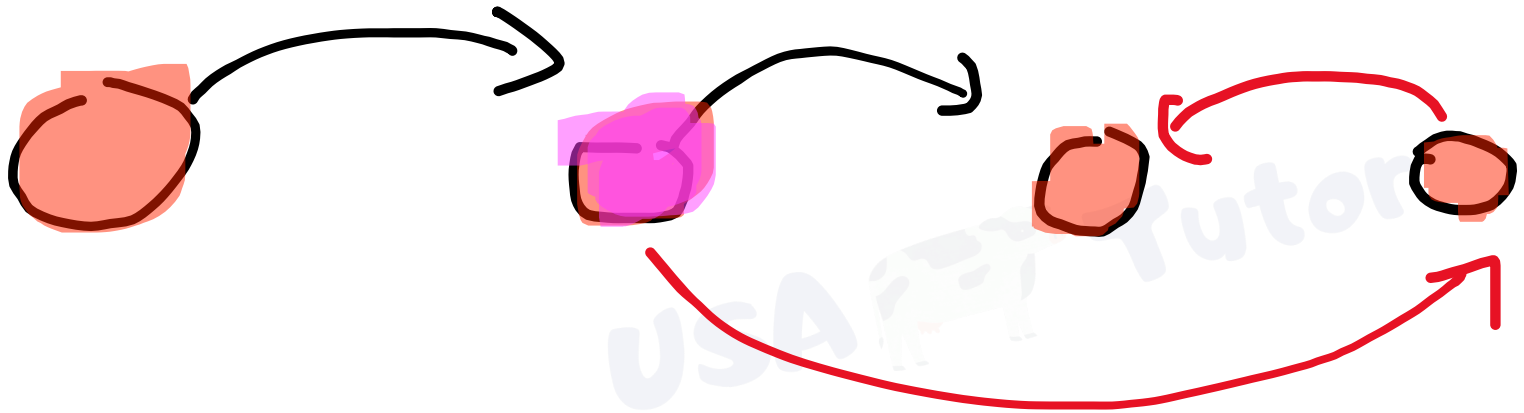
1 2 3 4 2

Backtrack



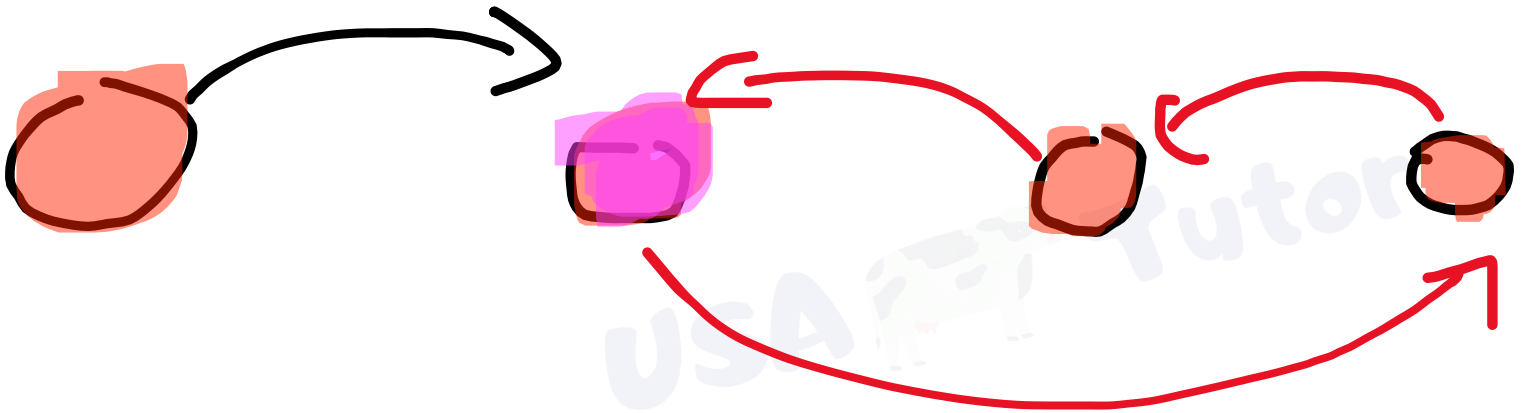
1 2 3 4

Backtrack



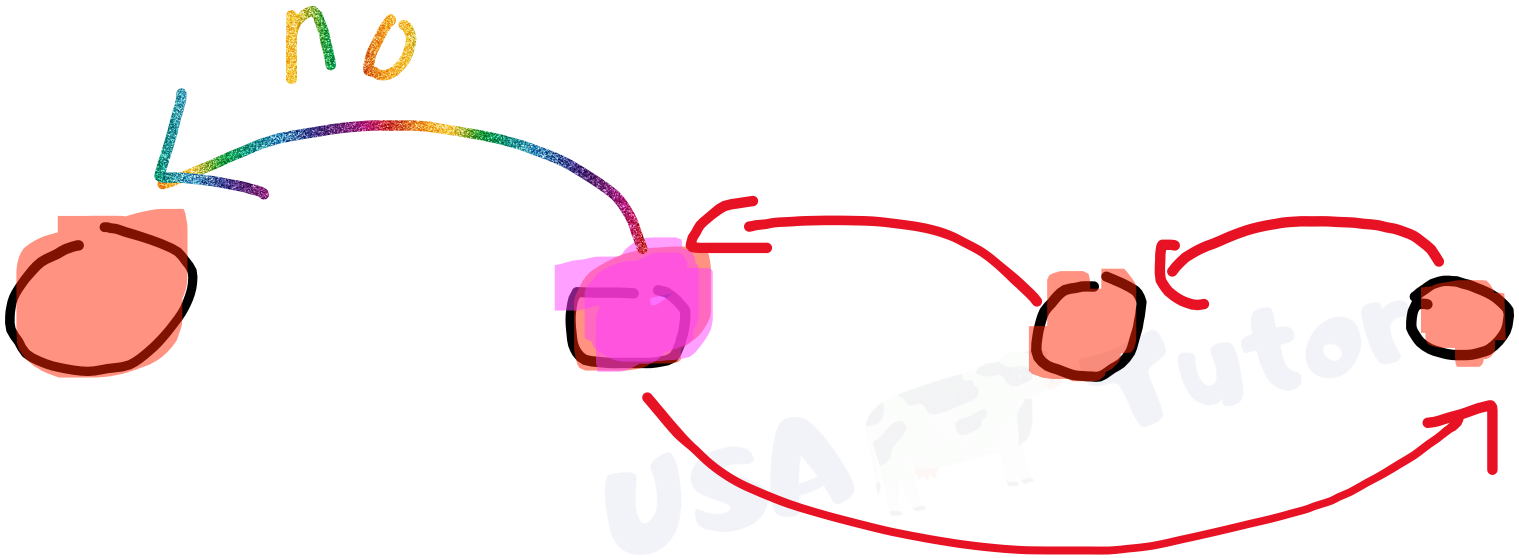
1 2 3

Backtrack



1 2

Backtrack



Coding Tutorial

Code is sent on LMS