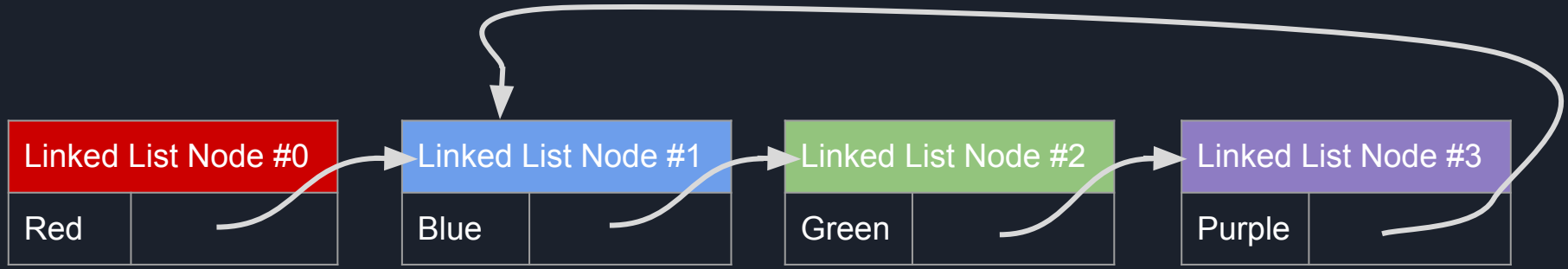
Linked Lists Cycles

Presented by Alexander Mak

Linked List Cycle

- A cycle is formed within a linked list when a node points to a previous node



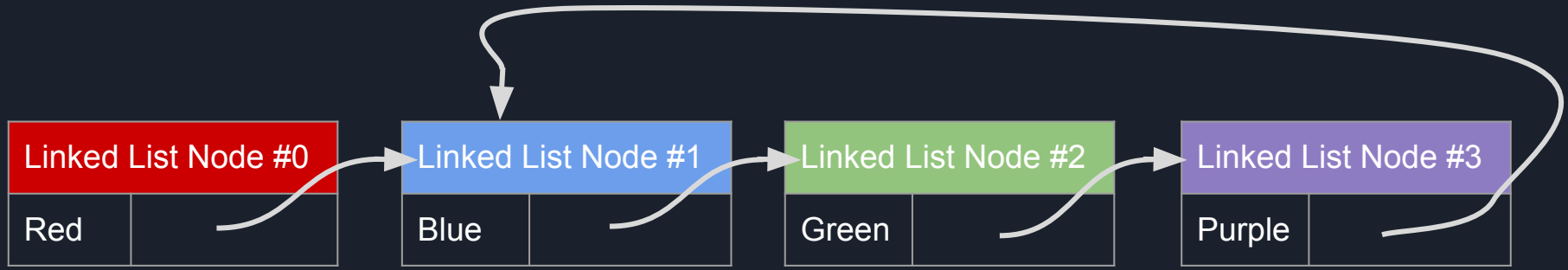
Cycle Detection

- Detecting cycles is a common algorithmic problem and is a concept
  - We will revisit this concept again in future data structures
- Cycle detection has various applications such as…
  - Ensure data integrity in data structures
  - Prevent infinite loops
  - Ensure unit tests are terminated when finding encountering repetitive execution patterns
  - Garbage collection with languages that have automatic memory management such as Python, Javascript, Ruby and Java.
    - Identify unreachable cyclic references, allowing memory to be reclaimed
- Linked List Cycle Detection Methods
  - Hashing
  - Floyd's Tortoise and Hare Algorithm - a.k.a. Fast and slow pointers

Presented by Alexander Mak

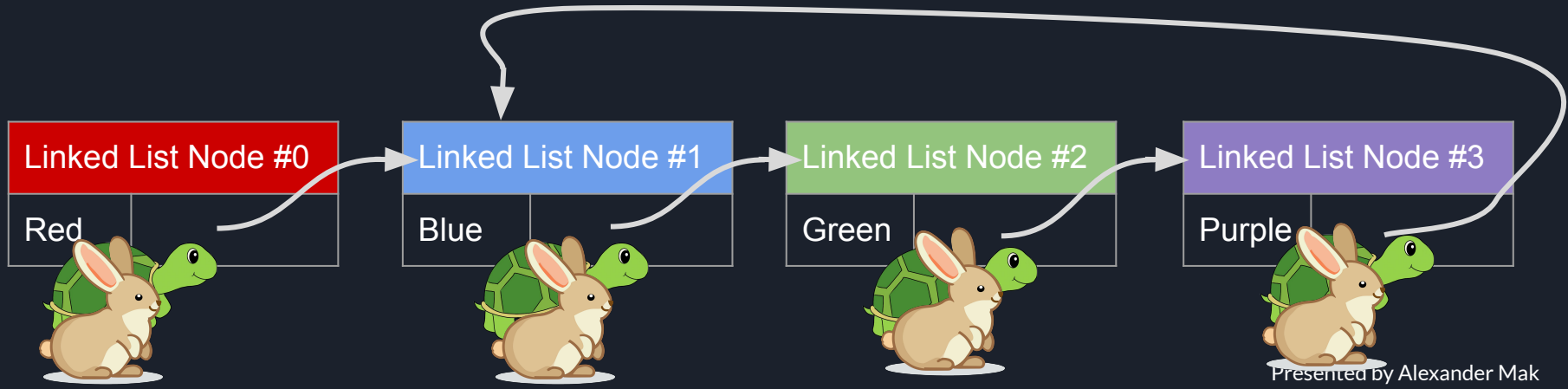
Hash Set

- As we iterate through the list, we can store visited nodes in a “visited set”
- If we find a node that we have already stored in our visited set, we know there is a cycle



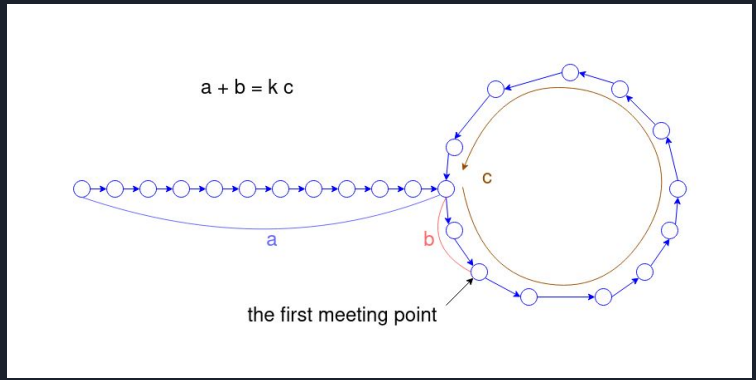
Fast and slow pointers

- Each iteration, move a slow pointer forward once and a fast pointer forward twice.
- If the fast pointer catches up to the slow pointer, there is a cycle.



Bonus: Finding the cycle starting point

- Note that this algorithm would be difficult to come up with on your own, and it is highly unlikely you would be asked to do this during an interview without assistance.
- This is an extension of Floyd's Tortoise and Hare Algorithm.
- TLDR
  - First use the usual cycle detection algorithm.
  - When you find the cycle (assuming it exists), create a third pointer at the head (a second tortoise).
  - This will be another slow pointer. Iterate until this new pointer meets the slow pointer.
  - This meeting point is the cycle entrance.



Bonus: Finding the cycle starting point proof

PROOF

- a = length of the path from the start of the list to the entrance of the cycle
- b = length of the path from the cycle's entrance to the meeting
- c = total length of the cycle

When the tortoise and the hare meet inside the cycle, the tortoise has walked a+b distance.

The hare, which moves twice as fast, has covered this distance and maybe a few more laps around the cycle. So, the total distance the hare ran is a+b+k*c.

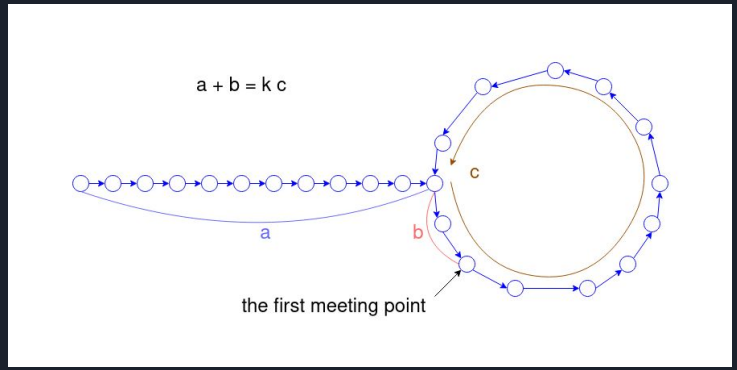
Because the hare moves twice as fast, this total distance is also equal to 2(a+b).

If we set these two equal: a+b+k*c=2(a+b)
This simplifies to k*c=a+b.

If start another tortoise back to the head at this point, it has “a” distance of until it reaches the cycle. Rearranging the above equation: a = k*c-b

The tortoise is “b” distance after from the entrance, so it has a distance of k*c-b until it reaches the entrance again.

Wait a sec… That’s the same distance!!



Note: You are NOT expected to memorize this proof for an interview. However, if an interviewer were to walk you through this, you’d be expected to follow along.

Questions?

Let’s practice!

https://github.com/Dijkstra-LLC/dsa_live_pro/tree/main/W05D02/classwork