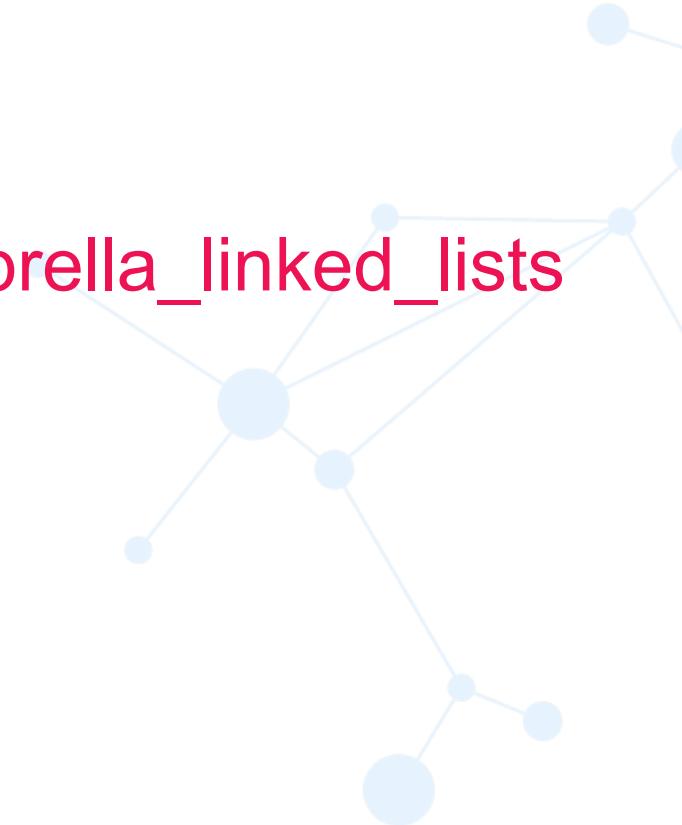


Arrays, Linked Lists, and Graphs: Why Connecting your Data Makes Sense

Clair J. Sullivan, PhD
Data Science Advocate
Neo4j

@CJLovesData1

https://github.com/cj2001/data_umbrella_linked_lists



Where are we going?



- Arrays
- Lists, a special array
- Linked lists
 - Singly-linked
 - Doubly-linked
- What to do when all of the above fails?

Arrays

```
my_data = [42, 186,  
365, 414, 1001]
```

0x00200

0x00204

0x00208

0x0020C

0x0020F

42
186
365
414
1001

```
my_data.insert(1, 238)
```

```
my_data = [42, 238, 186, 365, 414, 1001]
```

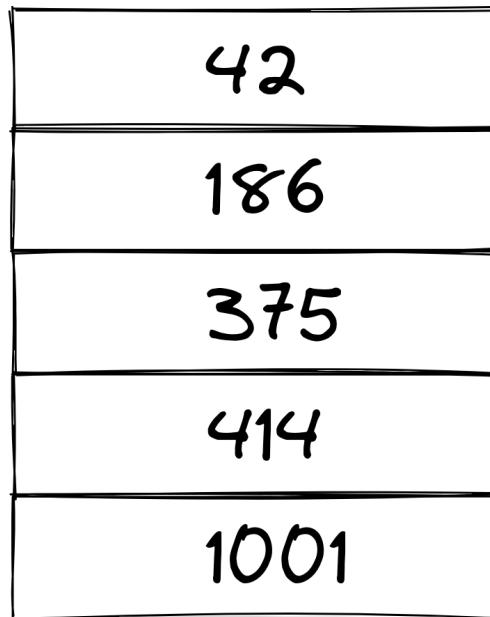
0x00200

0x00204

0x00208

0x0020C

0x0020F



238

- Copy element 2 to element 3
- Copy element 3 to element 4
- ...

O(n)

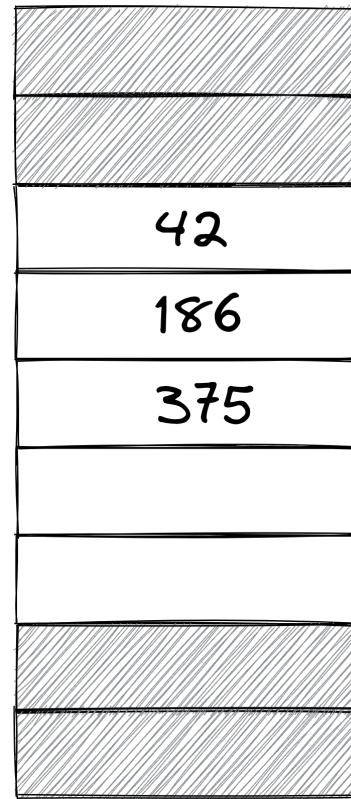
0x00200
0x00204
0x00208
0x0020C
0x0020F

42
186
375
414
1001

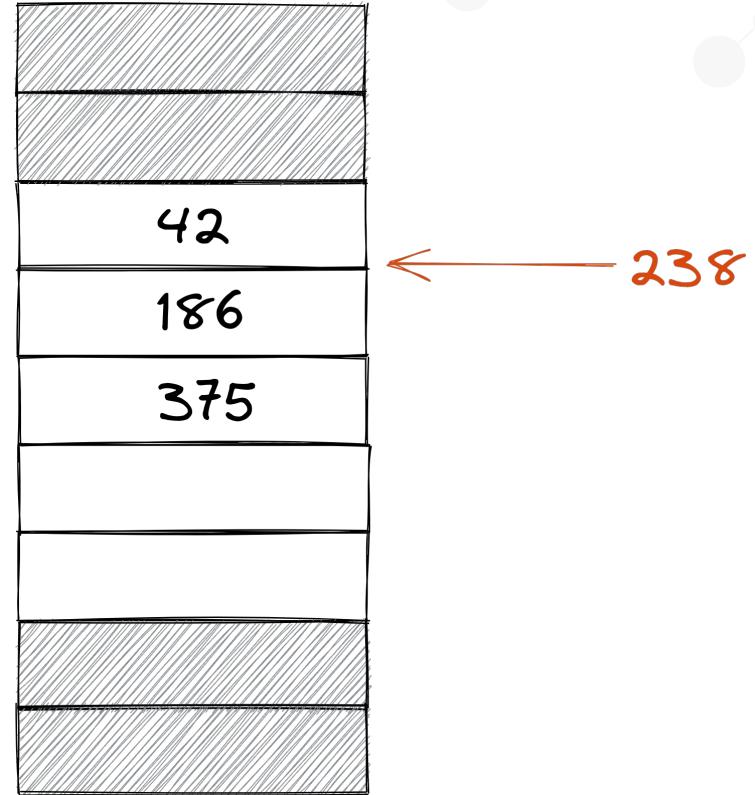
← 238

Lists: Dynamic Arrays

```
my_data = []  
my_data.append(42)  
my_data.append(186)  
my_data.append(375)
```

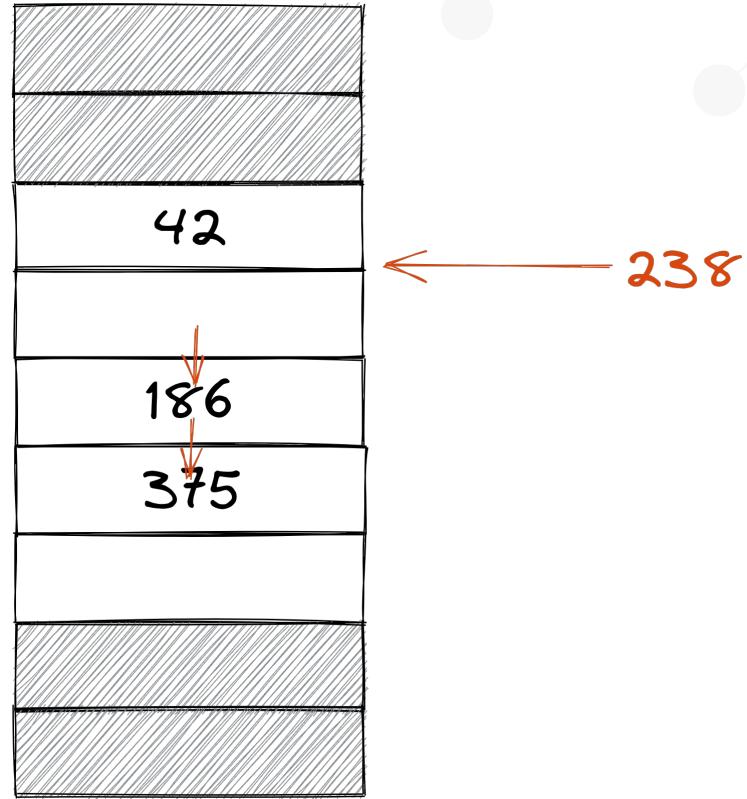


```
my_data.insert(1, 238)
```



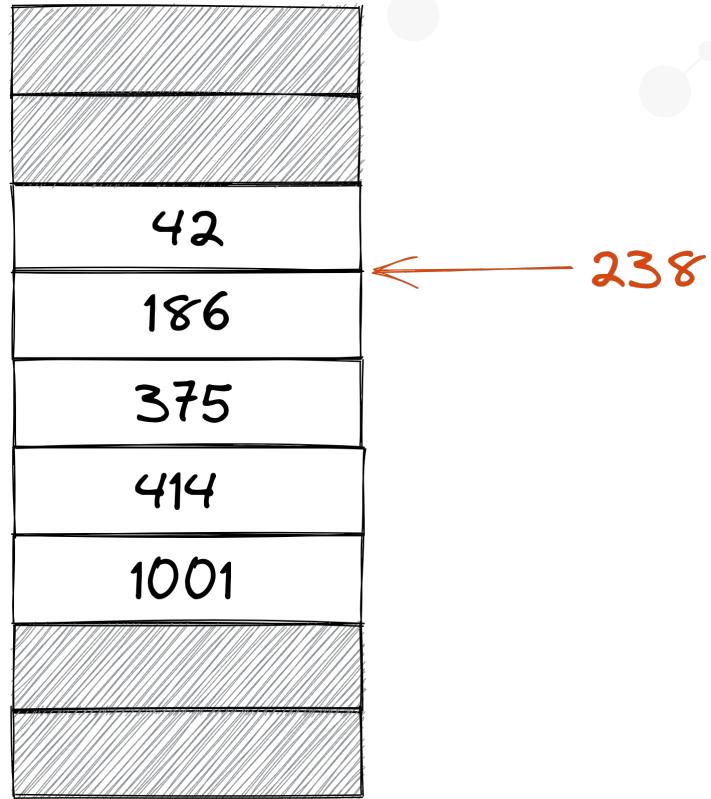
my_data.insert(1, 238)

$O(n)$



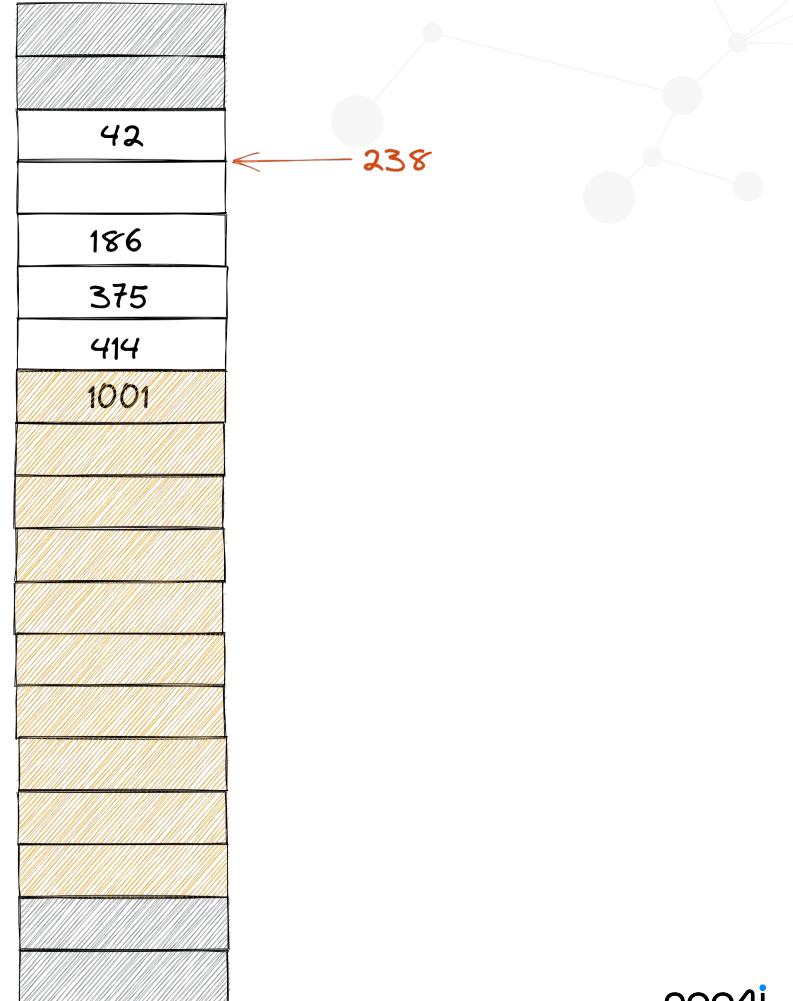
```
my_data = []
my_data.append(42)
my_data.append(186)
my_data.append(375)
my_data.append(414)
my_data.append(1001)

my_data.insert(1, 238)
```



Inefficient because:

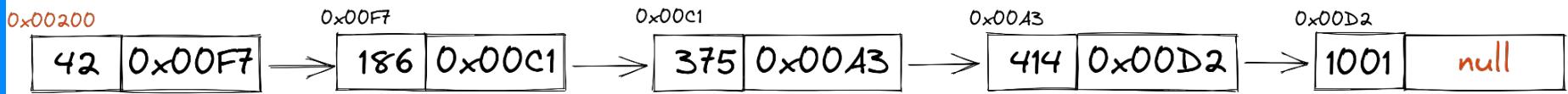
1. Swap elements
2. Allocate new memory
3. Copy elements from old memory into new memory



There must be a better way!!!

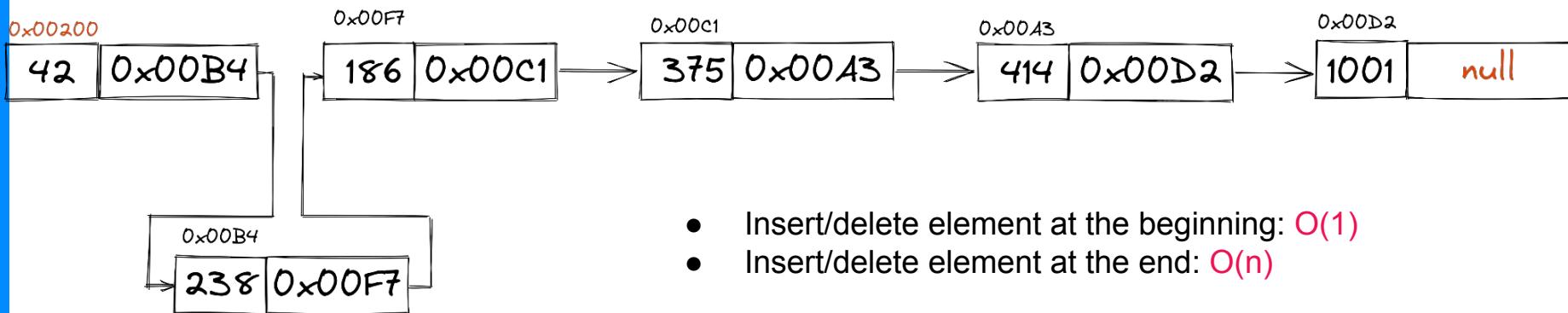


(Singly) Linked List



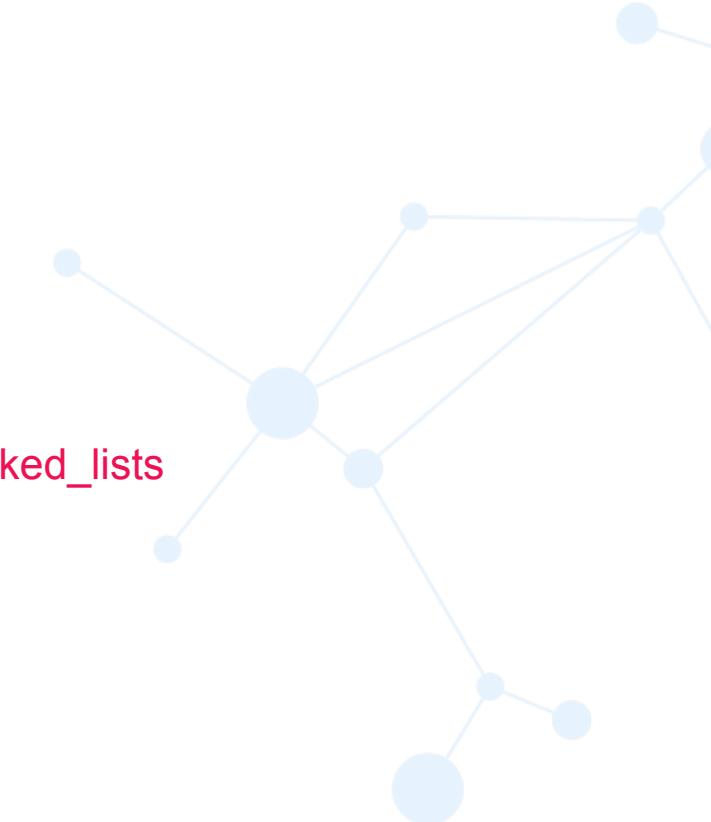
- Linked list traversal: $O(n)$
- Accessing elements by value: $O(n)$

List Insertion

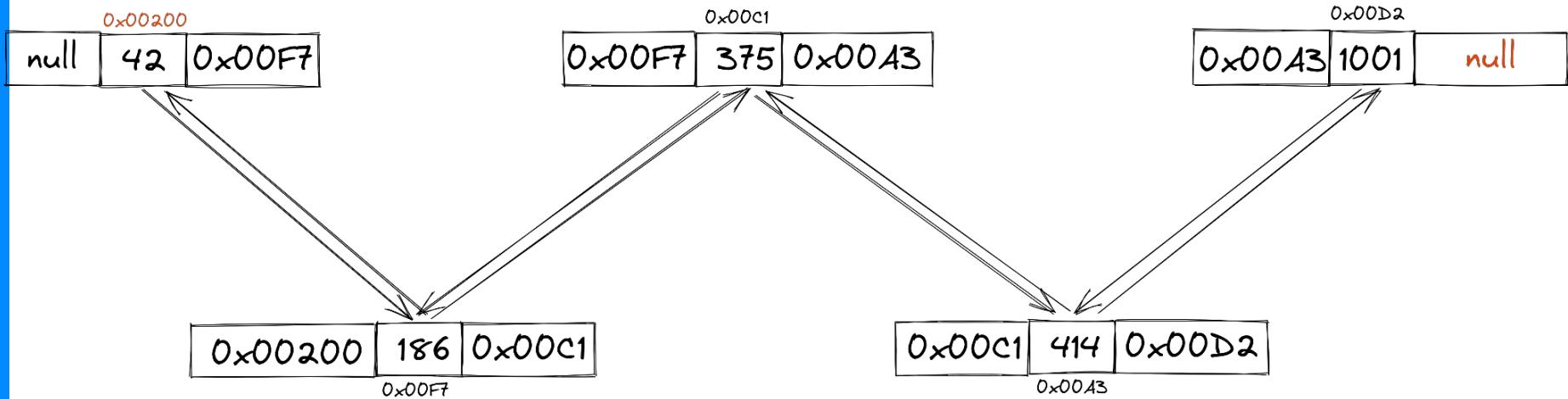


basic_linked_list.ipynb

Repository: https://github.com/cj2001/data_umbrella_linked_lists



(Doubly) Linked List



Benefits of Linked Lists vs. Arrays

Arrays win when:

- You are able to identify the index of the element you need: O(1)
- You expect to insert or delete elements at the end (assuming you are not expecting to go beyond allocated memory): O(1)

Linked lists win when:

- You need fast, easy insertion of new data: O(1)
- You don't want the burden of pre-allocating memory for the data structure

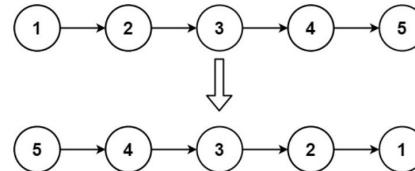
What You Need to Implement a Linked List

- A node class
 - Holds the data itself
 - Integers, numbers, complex objects
 - Requires:
 - Data
 - Next
 - Previous (only for doubly linked lists)
- A linked list class
 - Holds the pointers to the next element
 - Requires:
 - Head variable (pointer to first node)

LeetCode #206: Reverse Linked List (Iterative Solution)

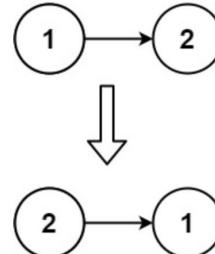
Given the head of a **singly linked list**, reverse the list, and return the reversed list.

Example 1:



Input: head = [1, 2, 3, 4, 5]
Output: [5, 4, 3, 2, 1]

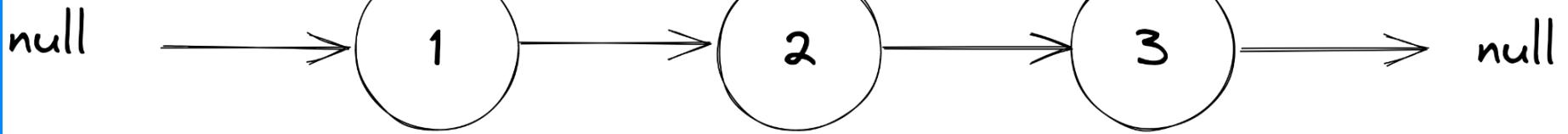
Example 2:

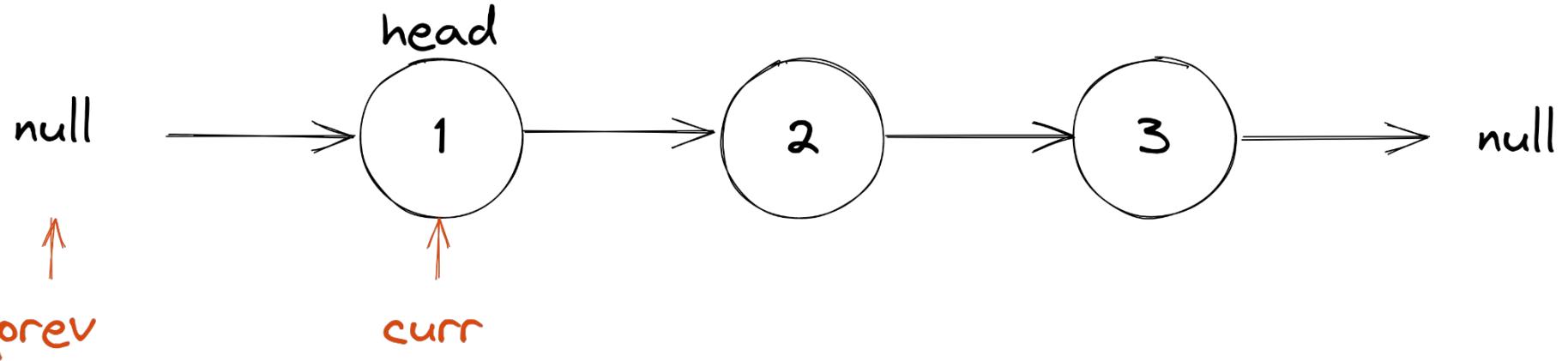


Input: head = [1, 2]
Output: [2, 1]

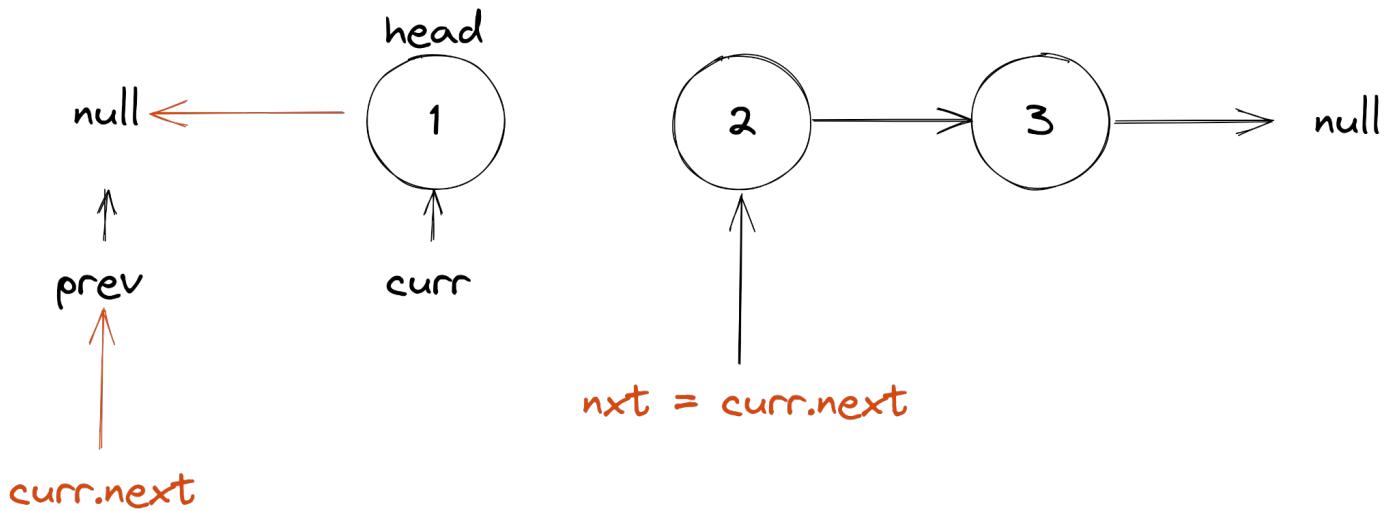
Example 3:

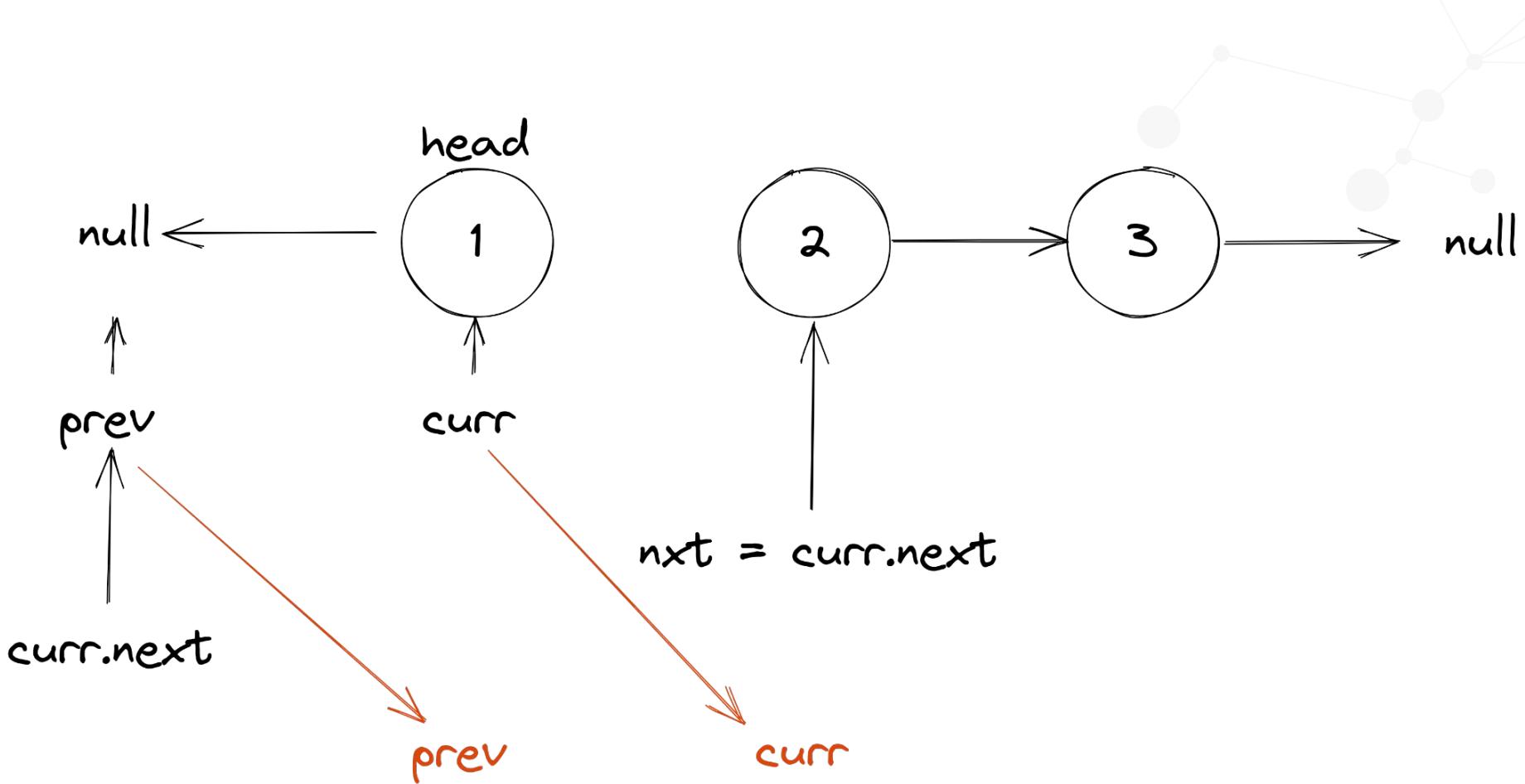
Input: head = []
Output: []

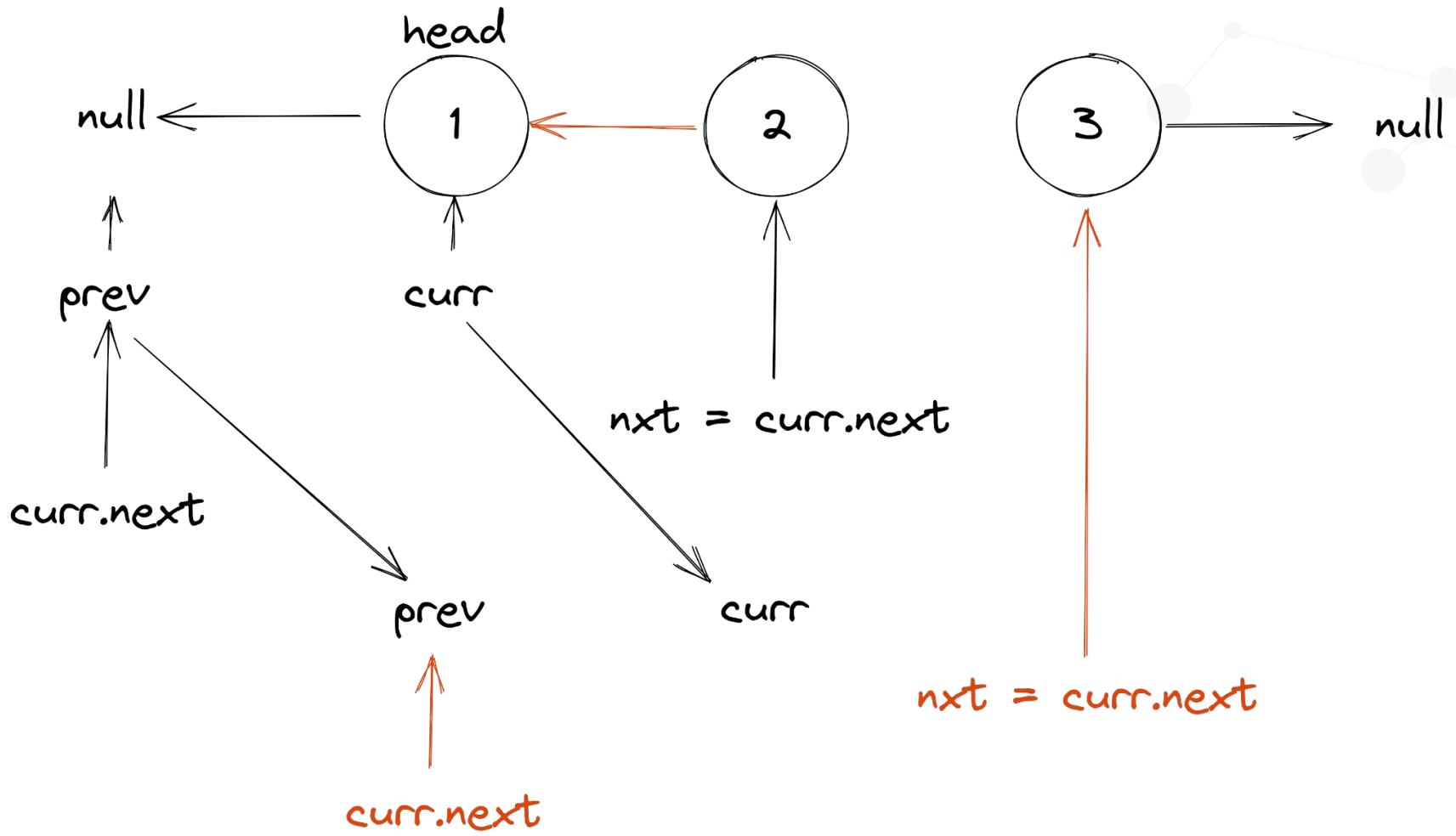


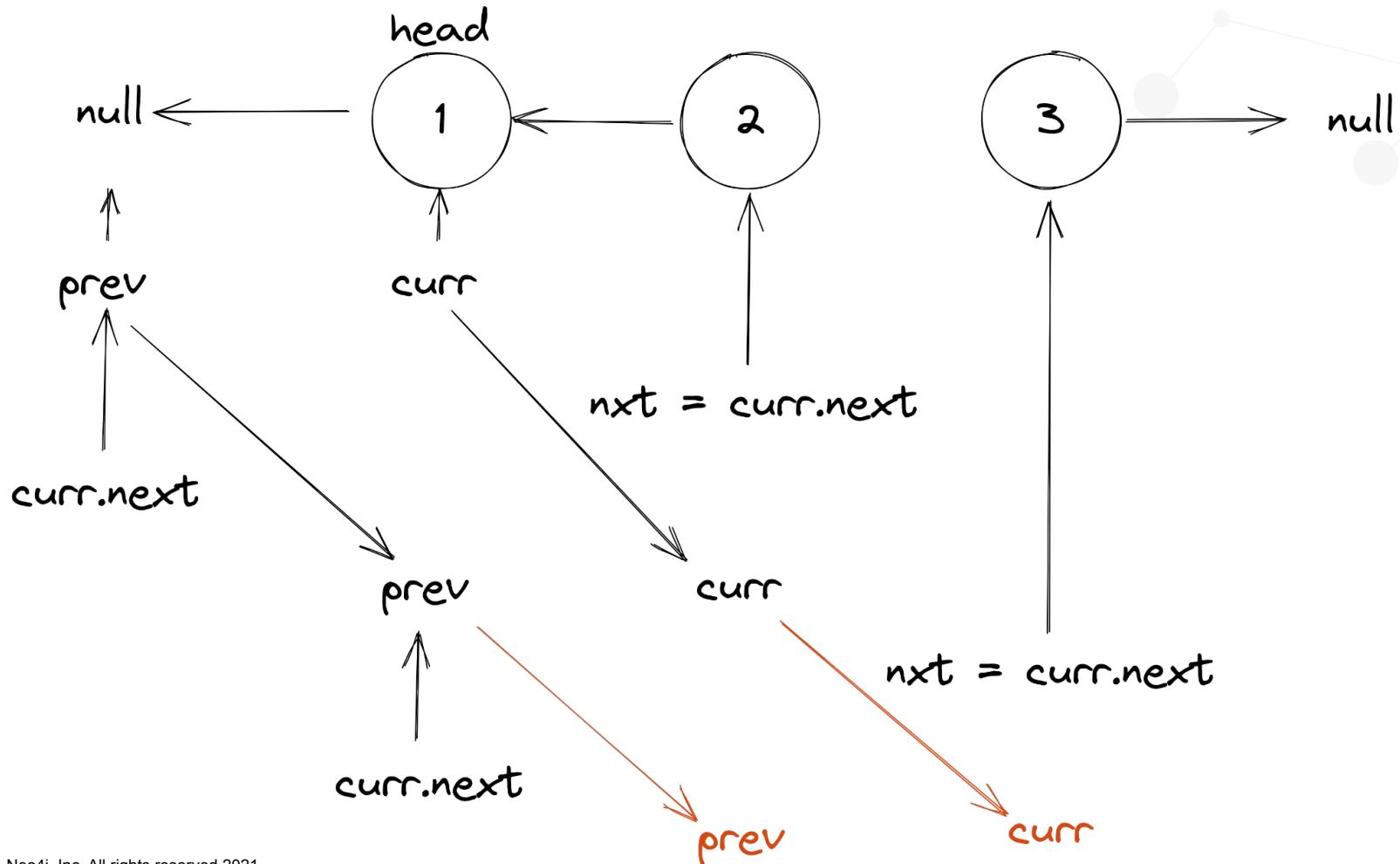


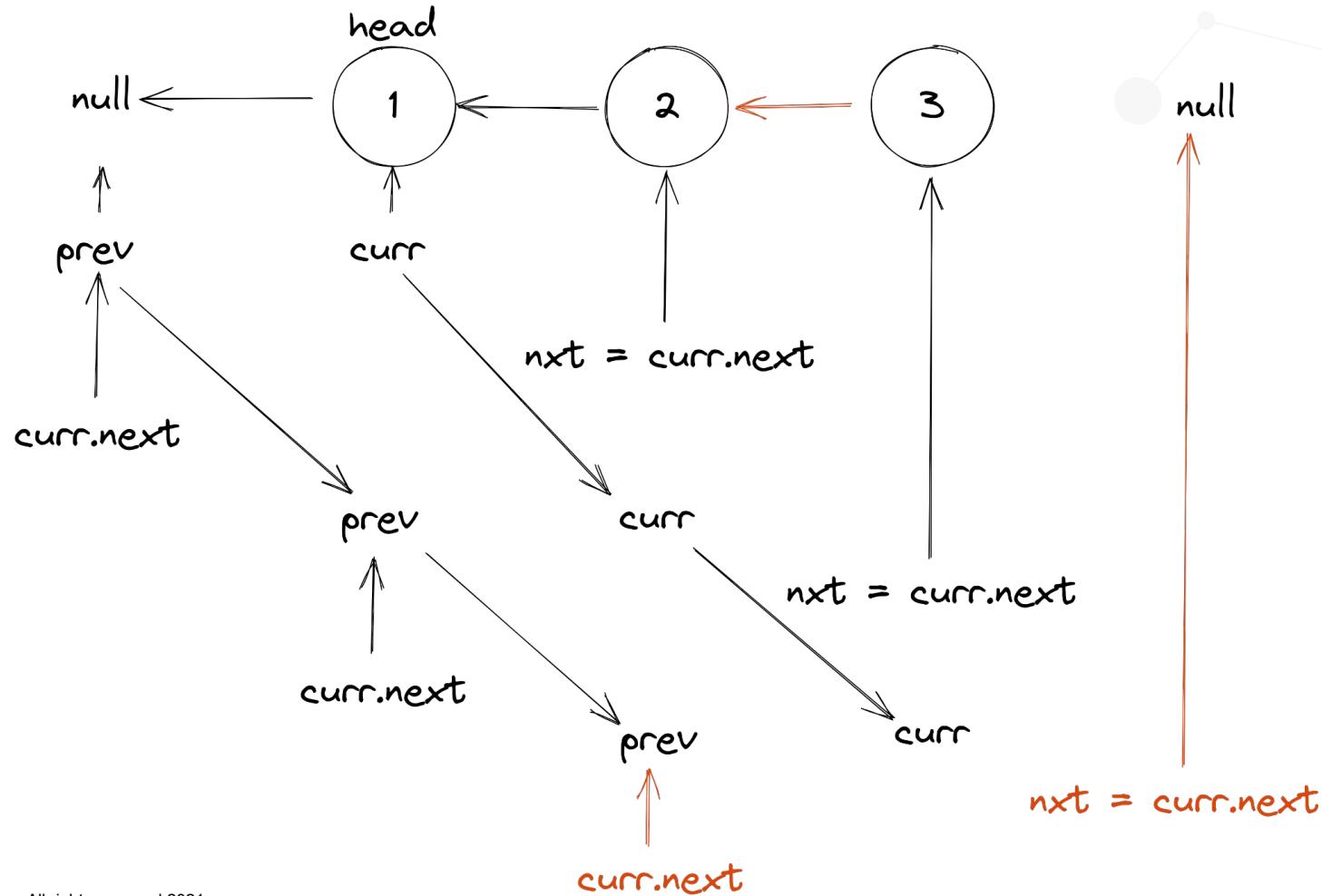
while curr:

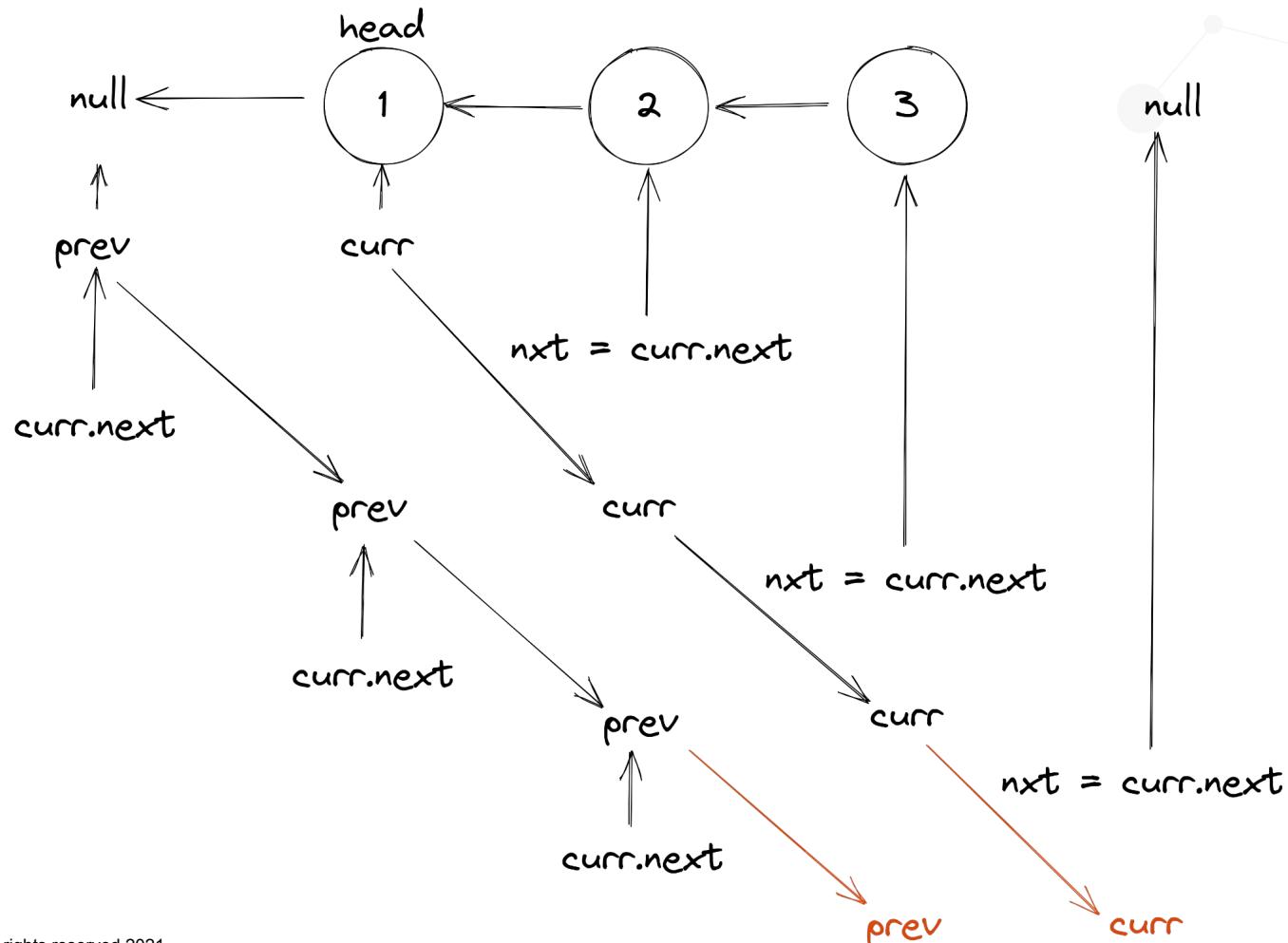


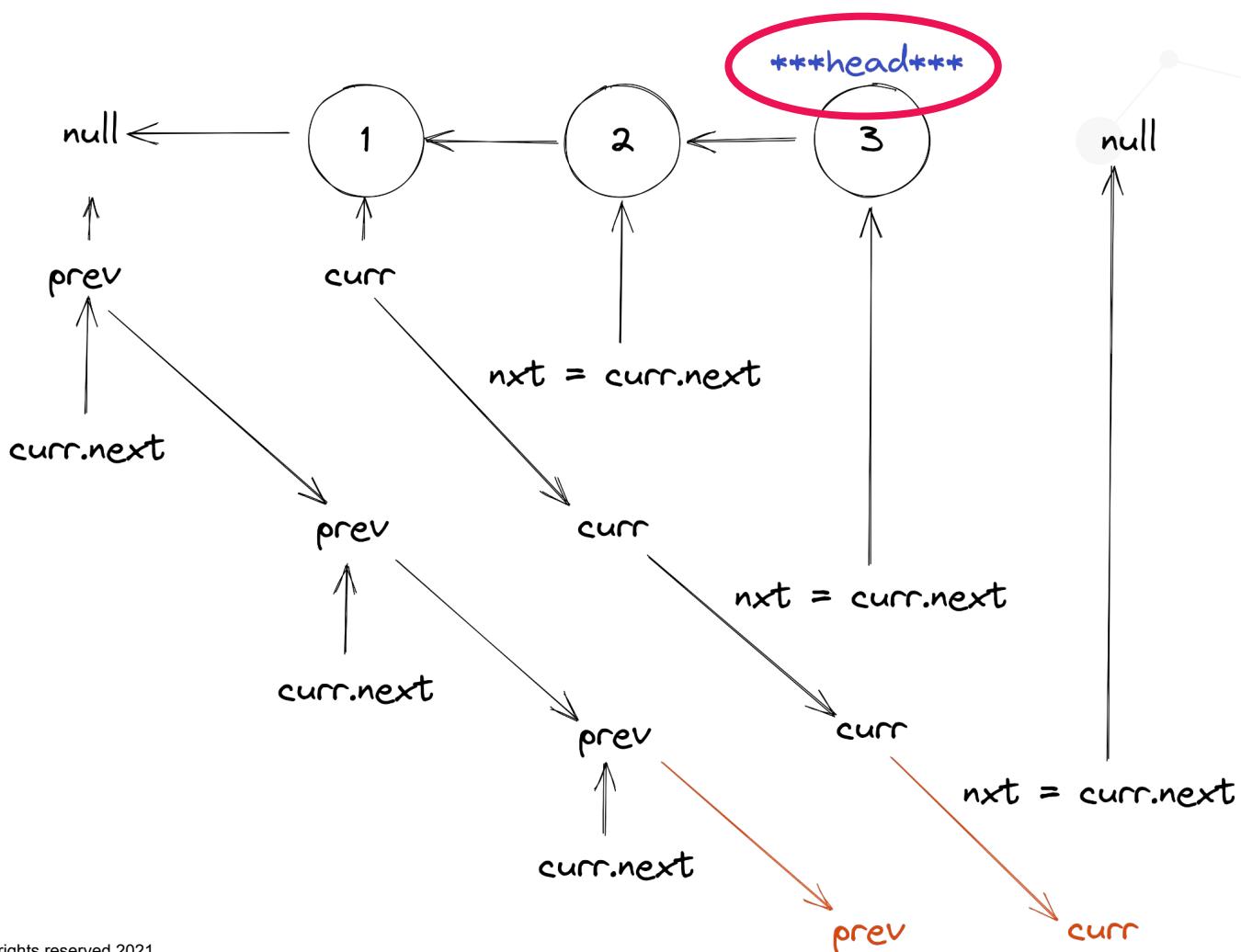






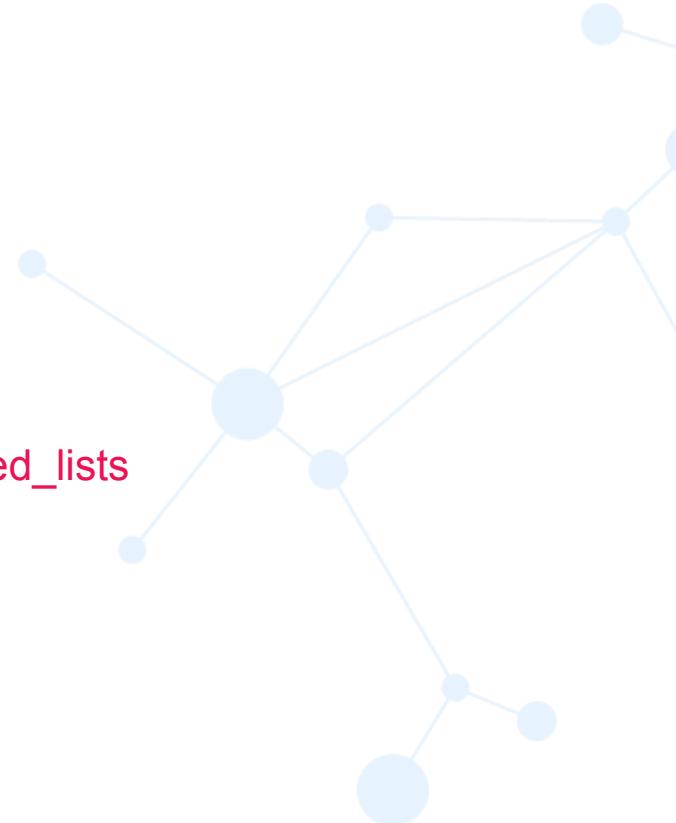






leetcode206.ipynb

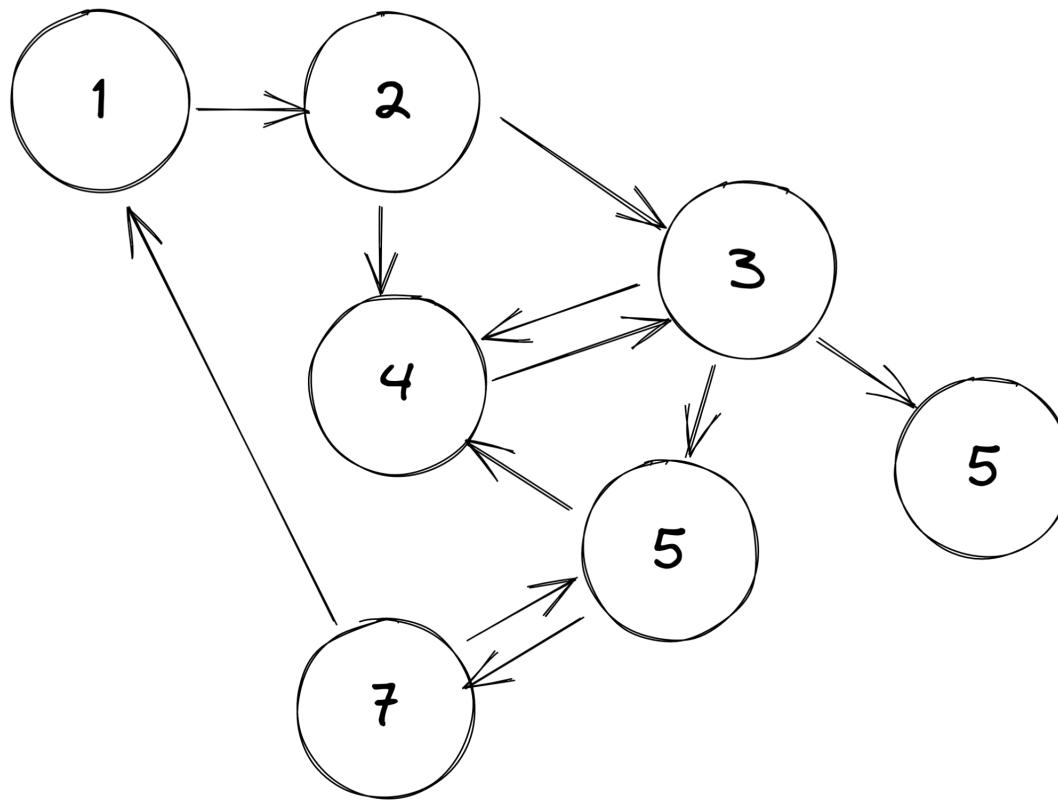
Repository: https://github.com/cj2001/data_umbrella_linked_lists



**(This problem would have been a
lot easier if we could have used a
doubly-linked list!!!)**



But what if...



Graphs to the rescue!



- A collection of nodes and relationships (AKA edges)
- Nodes can have:
 - Labels (multiple?)
 - Properties (multiple?)
- Relationships can have:
 - Labels
 - Properties (multiple?)
- Graphs can have multiple node and relationship types

How to Get into Graphs with Python

In-memory via Python packages

- Simple
- Minimal overhead in terms of database management
- Easy to query directly from Python

Graph databases

- Can involve infrastructure complexity
- Like SQL, typically uses its own query language
- Scalable
- Suited for heavy compute, complicated queries, data science, machine learning tasks

sandbox.neo4j.com



neo4j\$ match (n) return n limit 300



*(171) Movie(38) Person(133)

(253)

ACTED_IN(172)

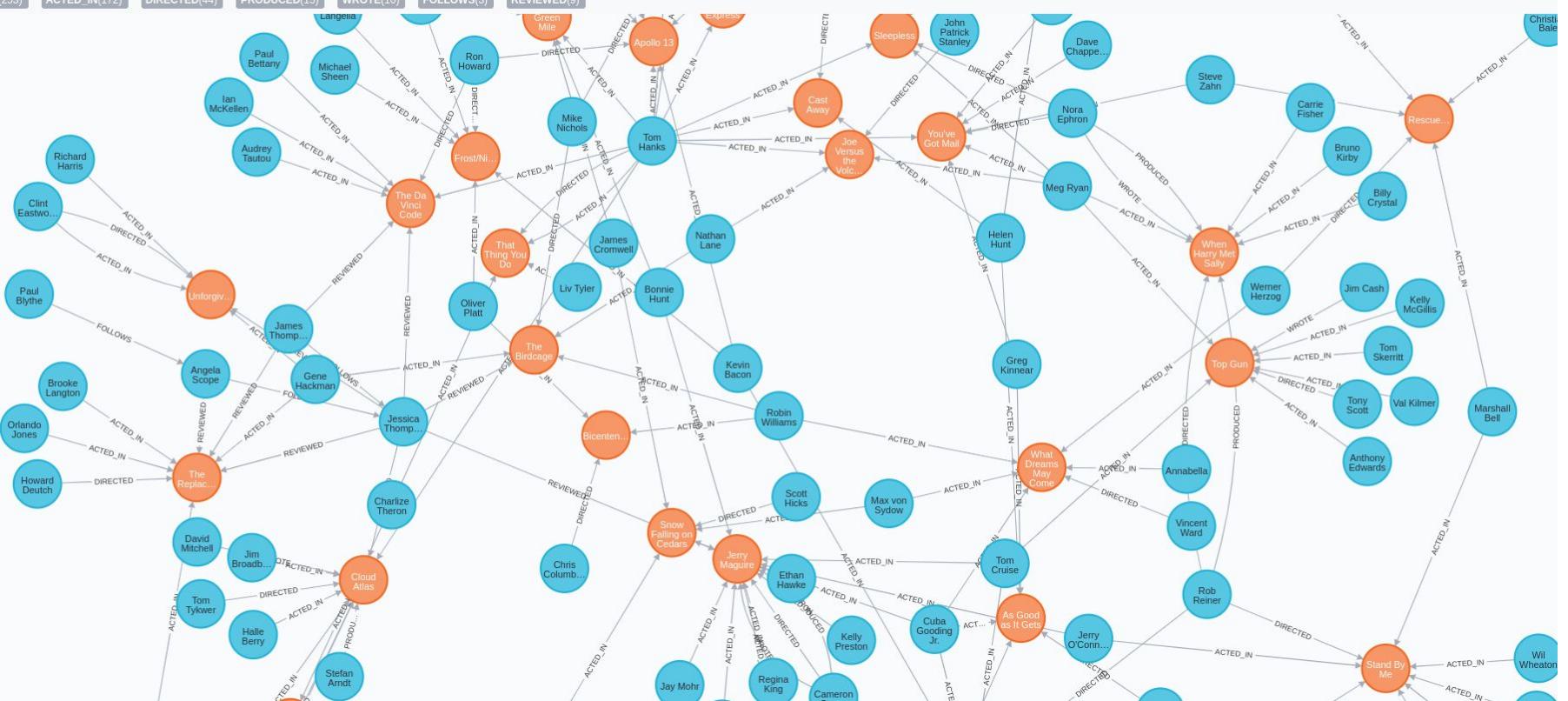
DIRECTED(44)

PRODUCED(15)

WROTE(10)

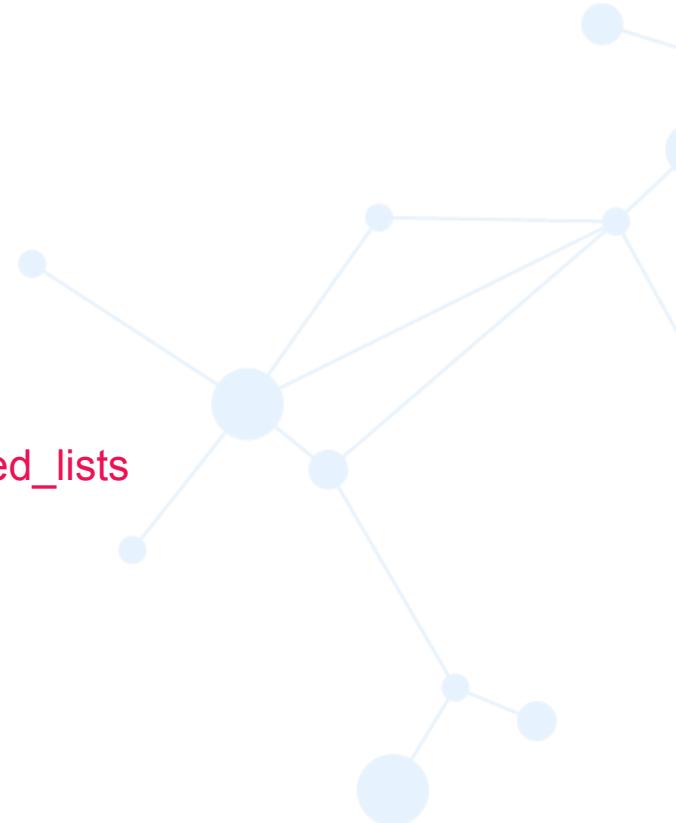
FOLLOWS(3)

REVIEWED(9)



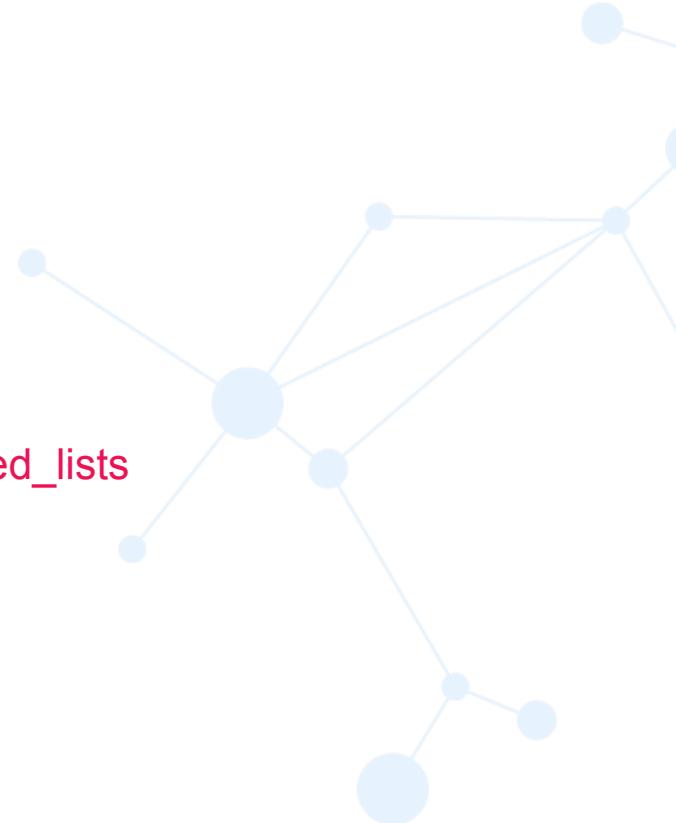
basic_graph.cql

Repository: https://github.com/cj2001/data_umbrella_linked_lists



got_queries.cql

Repository: https://github.com/cj2001/data_umbrella_linked_lists



graph_data_science.ipynb

Repository: https://github.com/cj2001/data_umbrella_linked_lists

Recap



- Arrays and lists are limited by insertion problems ($O(n)$)
- Linked lists are beneficial because:
 - Insertion is $O(1)$
 - You don't need to pre-allocate memory
- As network of nodes becomes more complicated, it is best to move to a graph representation because:
 - Easy query of complicated networks
 - Can take advantage of a variety of Python packages
 - Can scale when implemented in a graph database

https://github.com/cj2001/data_umbrella_linked_lists



Thank you!

@CJLovesData1

