Linked List Basics

Linked List

A linked list is a linear data structure that includes a series of connected nodes. Here, each node stores the data and the address of the next node. For example,

Linked List vs Array

ARRAY	LINKED LISTS
1. Arrays are stored in contiguous location.	1. Linked lists are not stored in contiguous location.
2. Fixed in size.	2. Dynamic in size.
3. Memory is allocated at compile time.	3. Memory is allocated at run time.
4. Uses less memory than linked lists.	4. Uses more memory because it stores both data and the address of next node.
5. Elements can be accessed easily.	5. Element accessing requires the traversal of whole linked list.
6. Insertion and deletion operation takes time.	6. Insertion and deletion operation is faster.

Traverse in a Linked List

Insert in a Linked List

Delete in a Linked List

Find the Middle Element in a Linked List

Types of Linked Lists

1. Singly Linked List
2. Doubly Linked List
3. Circular Linked List

Delete an element whose pointer is given in a Linked List.

Practice Problems

1. Find the Kth Element from last in a Linked List.
2. Remove duplicates from a Sorted Linked List.
3. Sort a Linked List using Bubble sort.
4. Find the intersection of Two sorted Linked List.
    - First linked list: 1->2->3->4->6
    - Second linked list be 2->4->6->8,
    - Output: 2->4->6.
5. Check if a Singly Linked List is a Palindrome.

Linked List Problems - I

Reverse a Linked List - Iteratively

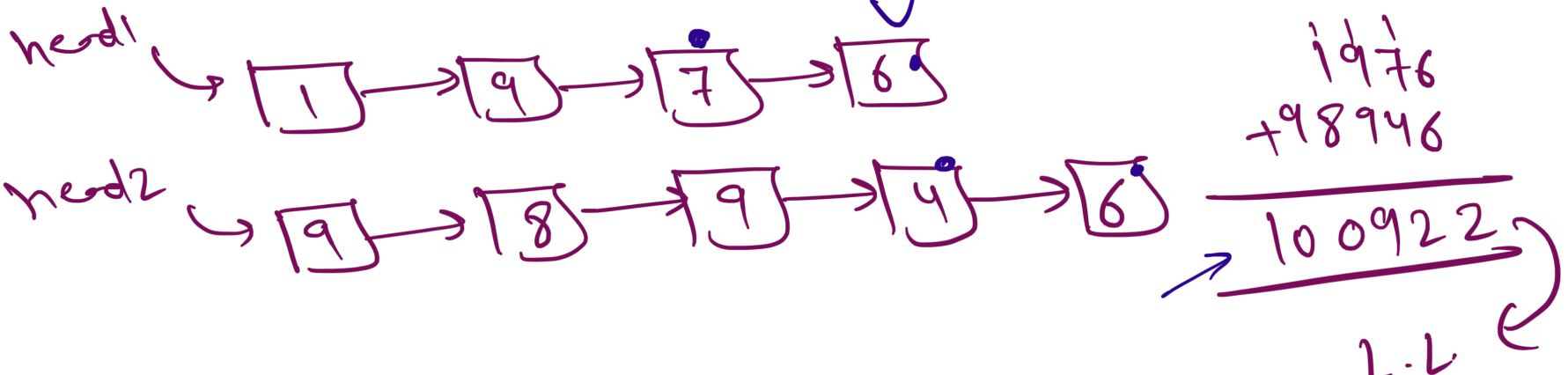
Reverse a Linked List - Recursively

Reverse a Linked List in a group of k

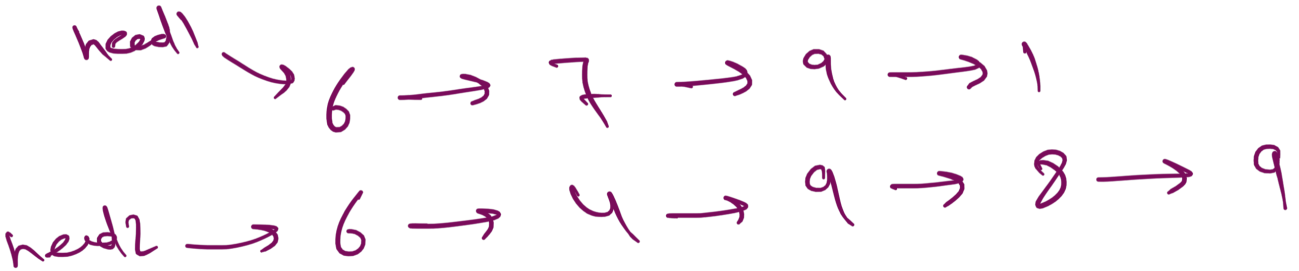
Detect a cycle in a Linked List

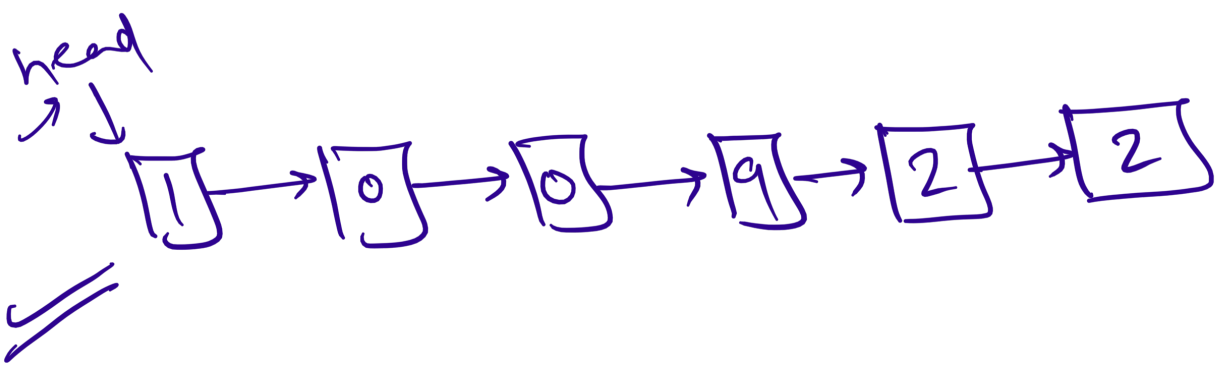
Practice Problems

1. Palindrome Linked List.
2. Rotate a Linked List by k nodes.
3. Add two numbers (each digit is present inside a node of linked list)
4. Merge a linked list into another linked list at alternate positions.



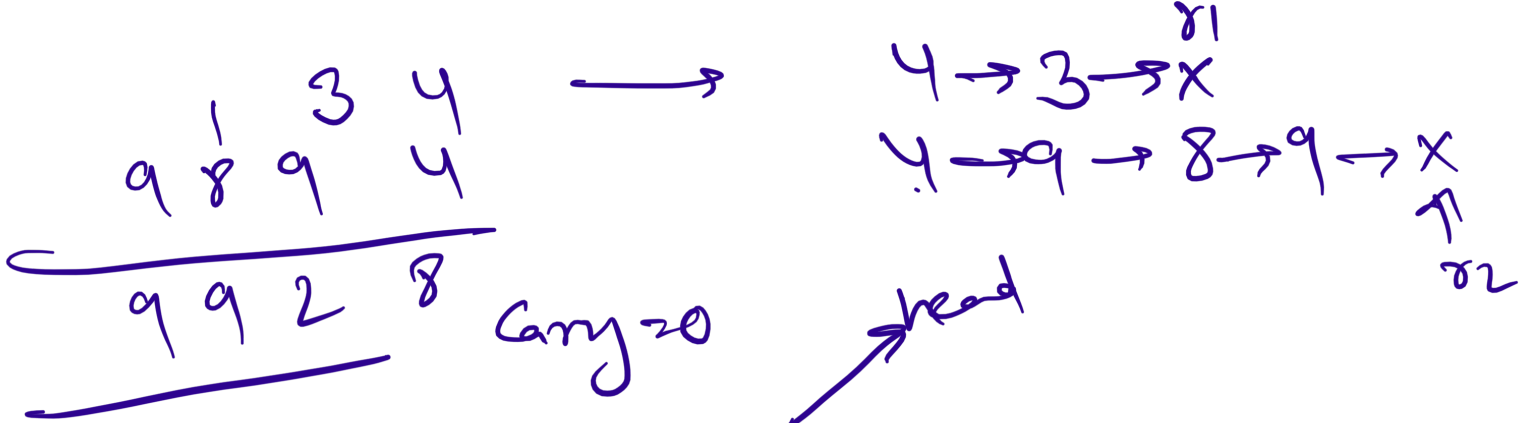
Reverse both LL.

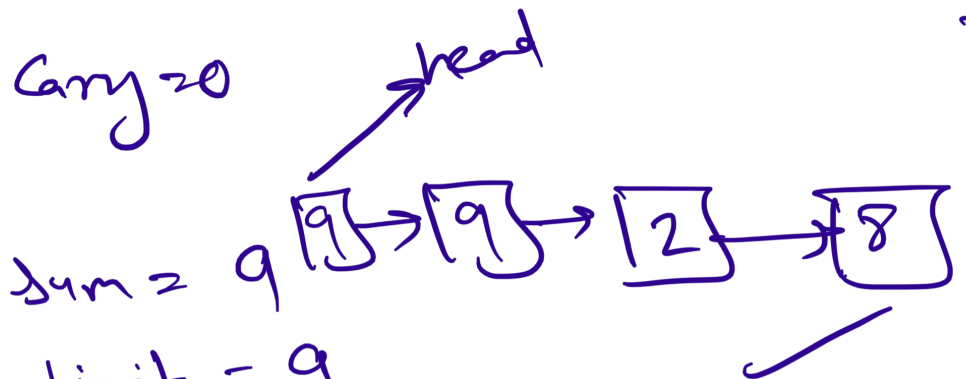




Carry = 0

O(n)





Sum = 9

digit = 9

Carry = 0

Linked List Problems - II

Merge two sorted Linked List

Palindrome Linked List

Doubly Linked List Implementation

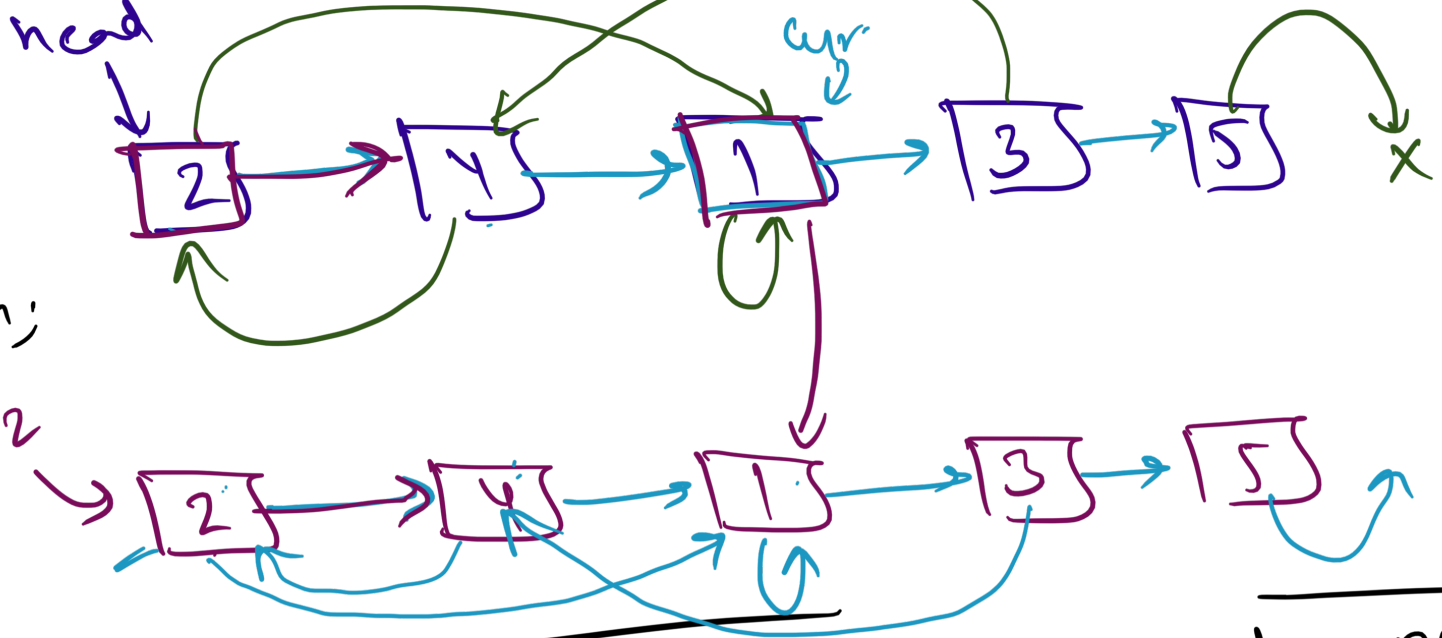
Implement LRU Cache

We are given the total possible page numbers that can be referred. We are also given a cache (or memory) size (The number of page frames that the cache can hold at a time). The LRU caching scheme is to remove the least recently used frame when the cache is full and a new page is referenced which is not there in the cache.

Practice Problems

1. Clone a Linked List with next and random pointer.
2. Given a linked list A , reverse the order of all nodes at even positions.
    - `Input =  1 -> 2 -> 3 -> 4`
    - `Output = 1 -> 4 -> 3 -> 2`
3. https://www.interviewbit.com/courses/programming/linked-lists

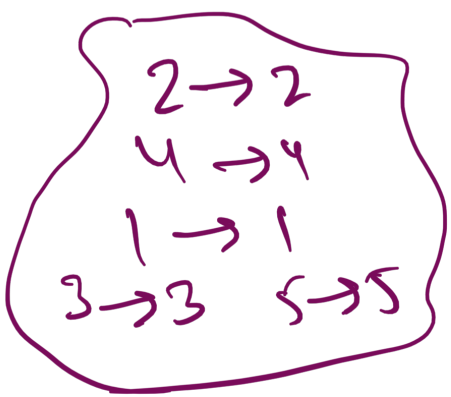
```
class Node{
  int data;
  Node next;
  Node random;
}
```



* `cur.next.random = cur.random.next`

`Map<Node, Node> map`

`map.put(2, 2)`

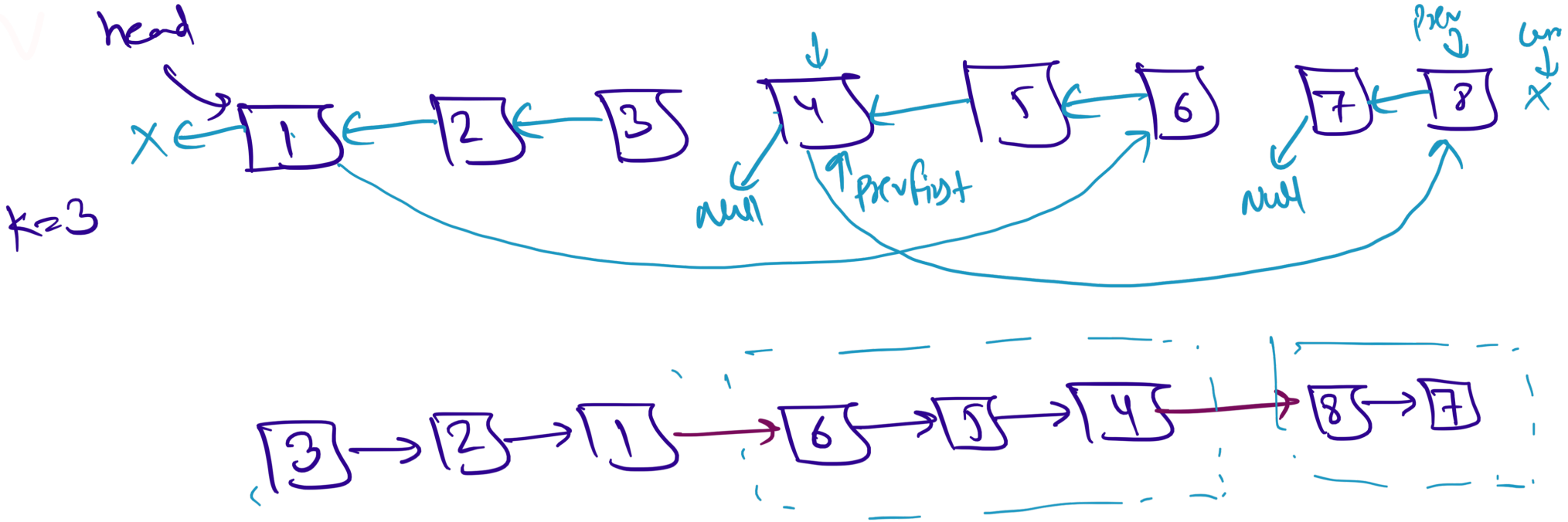


`dup.random = map.get(cur.random);`

1. Assign next pointer to another linkedlist
2. Assign random pointers
3. Rebuild the next pointers in both 1.2.

T.C. $O(N)$

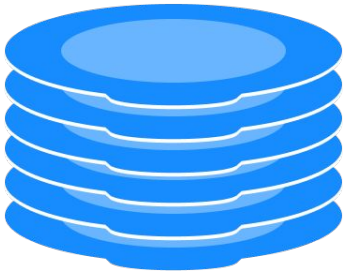
S.C $\rightarrow$ $O(1)$.

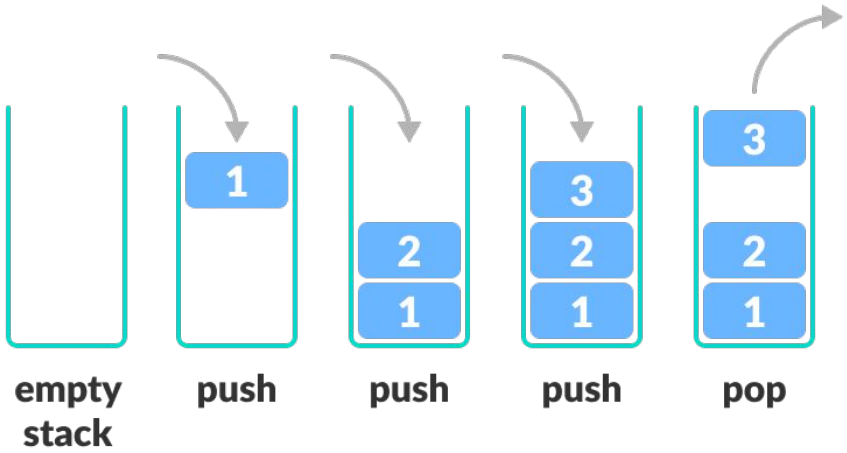

head
prev
cur
x
1
2
3
4
5
6
7
8
null
prevfirst
null
k=3

3
2
1
6
5
4
8
7

Stack Basics

Stack Data Structure

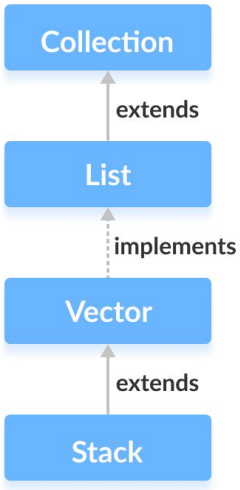


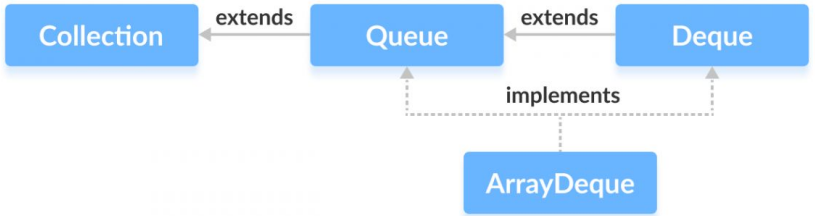


Implement Stack Using Array

Implement Stack Using Linked List

Stack & ArrayDeque in Collection Framework





- `push()` - adds an element to the top of the stack
- `peek()` - returns an element from the top of the stack
- `pop()` - returns and removes an element from the top of the stack

Parenthesis Matching Problem

Practice Problems

1. Previous / Next Greater Element
2. Previous / Next Smaller Element
3. Reverse words in a given Sentence.
4. Reverse a stack using Recursion
5. Reverse a stack without using extra space
6. Delete the mid element from a given Stack