CS2040C Tut 3

List ‘variants’

Applications

List ‘*variants*’

Singly vs Doubly Linked List

Singly/Doubly Linked List are **implementations**, not **ADT**.

Singly Linked List (SLL) only has *next* pointers.

- Can only iterate *forward*

Doubly Linked List (DLL) has both *next* and *prev* pointers.

- Can iterate both *forward* and *backward*

List ADT ‘variants’

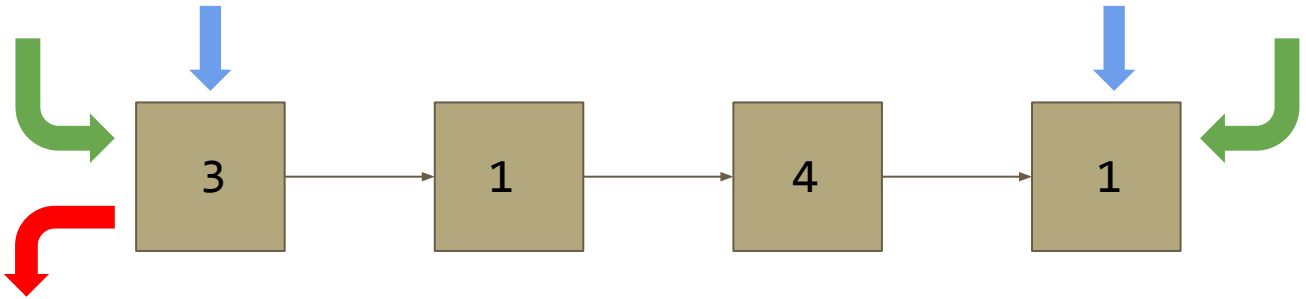
- You have seen List ADT in Tut 01.
- Stack, Queue and Deque ADTs are similar to List ADT.
  - Subset of operations

List ADT ‘variants’

- Singly Linked List can be used to implement:
  - Stack
  - Queue
- Doubly Linked List can be used to implement:
  - Deque (***C++ STL implementation varies***)

Singly Linked List

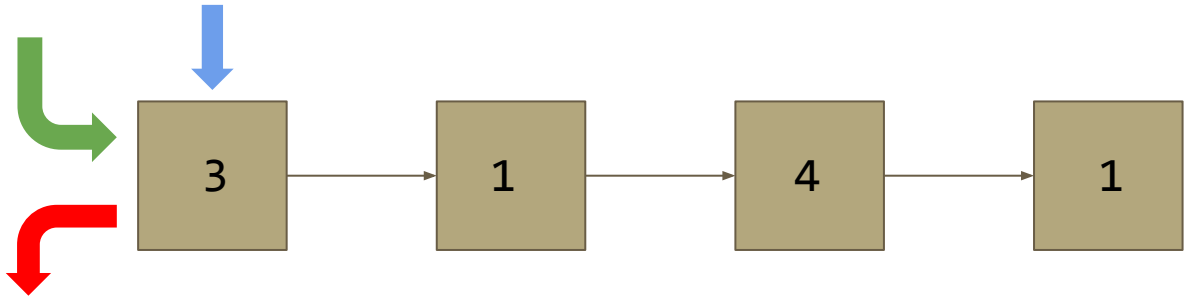
Below operations in O(1)



Stack

Subset of List ADT

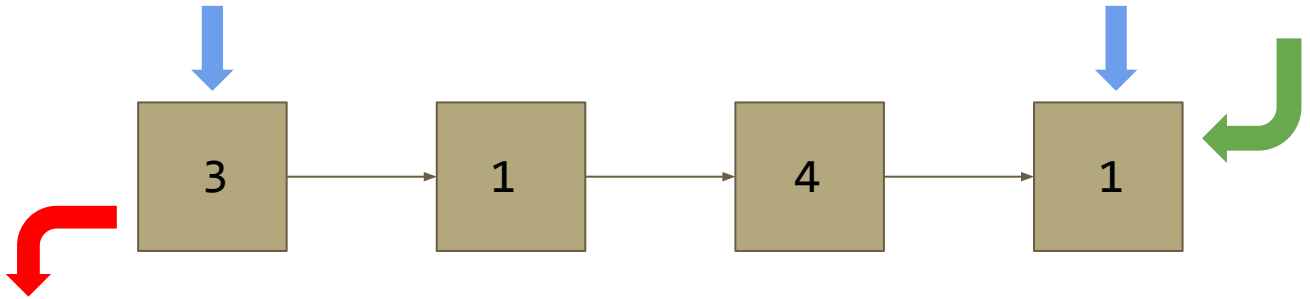
Below operations in O(1)



Queue

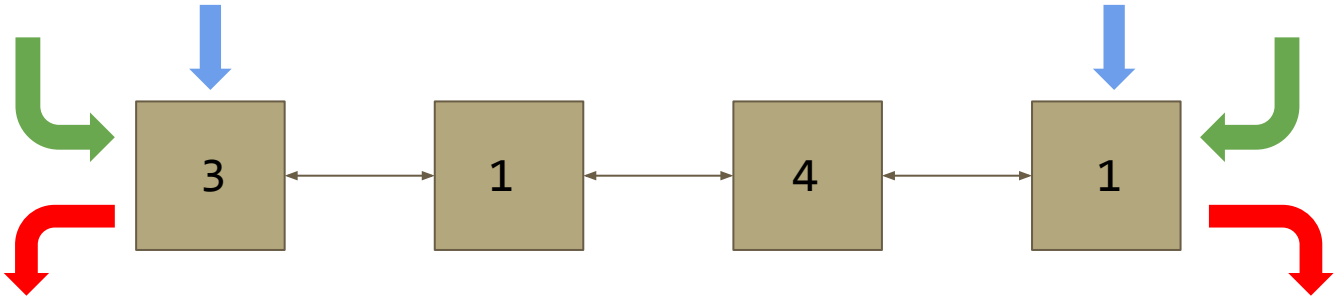
Subset of List ADT

Below operations in O(1)



Doubly Linked List

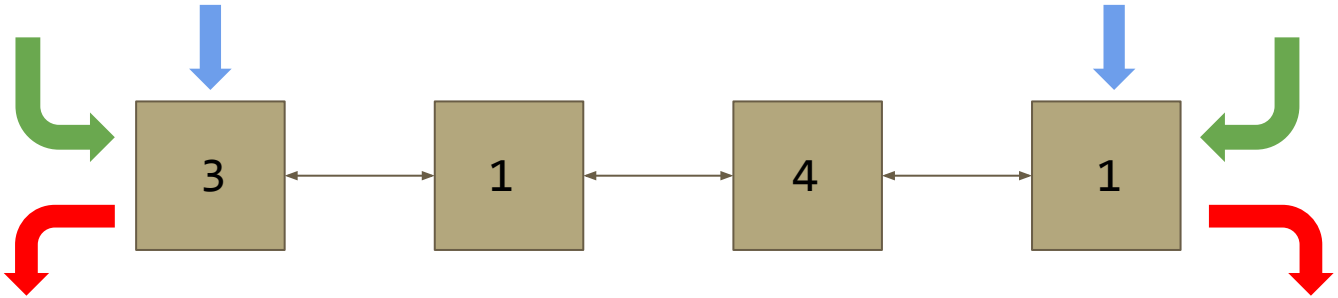
Below operations in O(1)



Deque

Just doubly linked list without *search* and *operations in the middle*.

Below operations in O(1).



Q1 Answers

Mode → ↓ Action	Singly Linked List	Stack	Queue	Doubly Linked List	Deque
search(any-v)	$O(N)$	not allowed	not allowed	$O(N)$	not allowed
peek-front()	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
peek-back()	$O(1)$	not allowed	$O(1)$	$O(1)$	$O(1)$
insert(0, new-v)	$O(1)$	$O(1)$	not allowed	$O(1)$	$O(1)$
insert(N, new-v)	$O(1)$	not allowed	$O(1)$	$O(1)$	$O(1)$
insert(i, new-v), $i \in$[1..N-1]	$O(N)$	not allowed	not allowed	$O(N)$	not allowed
remove(0)	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
remove(N-1)	$O(N)$	not allowed	not allowed	$O(1)$	$O(1)$
remove(i), $i \in$[1..N-2]	$O(N)$	not allowed	not allowed	$O(N)$	not allowed

Exercise

Reverse SLL

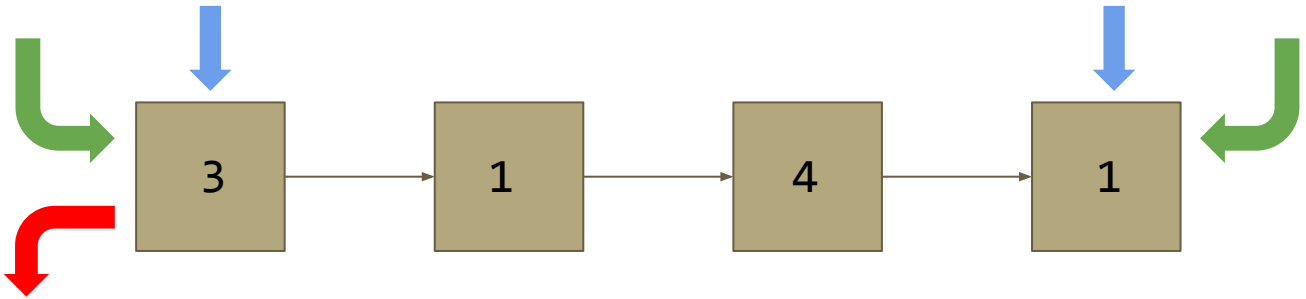
Reversing a SLL

Would anyone like to share?

Describe your solution.

The others: figure out the time complexity

Singly Linked List



Reversing a SLL

Method 1

Let the original list be **L** and another empty list be **R**

Iterate through **L** (forward),

Push elements successively at the *head* of **R**

R will be **L**, but in reverse order

Reversing a SLL

Method 2

Using recursion:

- Reach the end of **L**
- Reverse the direction of the last 2 elements
  - U → V becomes U ← V
- Repeat for the remaining as the recursion unwinds
- Swap the head and tail pointers

Reversing a SLL

Method 3 (Cheat)

Loop through **L**, store pointers to every element in an array.

(Then deconstruct **L**, if memory is tight)

Loop through the array in reverse order, construct the reversed linked list, **R**

Reversing a SLL

Can we do this faster than O(N)?

No.

Reversing a SLL

Can we do this faster than O(N)?

No.

Why?

At least **N** items that need to change their *next* pointers.

Hence, time complexity is Ω(**N**).

(Omega: at least this time complexity regardless of input)

Practice Problems

https://open.kattis.com/problems/backspace
https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&page=show_problem&problem=3139
https://open.kattis.com/problems/integerlists

Backspace

What *operations* do we need?

- Insert? (at where?)
- Delete? (from where?)
- Access (iterate through? Random access?)

What data structures can we use?

Backspace

What data structures can we use?

Many!

Which is easier to implement? :D

Broken Keyboard

What *operations* do we need?

- Insert? (at where?)
- Delete? (from where?)
- Access (iterate through? Random access?)

What data structures can we use?

Broken Keyboard

What data structures can we use?

List

Can we use other data structures?

Why / why not?

Integer Lists

What *operations* do we need?

- Delete? (from where?)
- *Reverse?*
  - Can we reverse ***faster* than O(N)**?

What data structures can we use?

Integer Lists

Direct simulation using Linked List

- Delete from front in **O(1)**
- Reverse in **O(N)**

Total complexity: O(NP)

N is up to 100,000

P is up to 100,000

Integer Lists

What happens if you reverse twice in succession?

Instead of actually reversing the list, can we just keep track of whether the list *should* be reversed or not?

Integer Lists

boolean isReversed

if (isReversed == false)

    Remove from the back

if (isReversed == true)

    Remove from the front

PS2

Editor Simulator

PS2

Isn’t it just ‘Backspace’ and ‘Broken Keyboard’ combined?

Yes

PS2A

No ‘[‘ operations.

Where will the cursor always be?

Can you ignore ‘]’ operations?

PS2B

N is still small, 2000.

O(**N^2**) can still run within the time limit.

PS2C

N is now large, 1,000,000.

O(**N**^{2}) *cannot* run within the time limit.

Questions?