LECTURE-6

Linear Data Structures

Queues, Circular Queues and Stacks

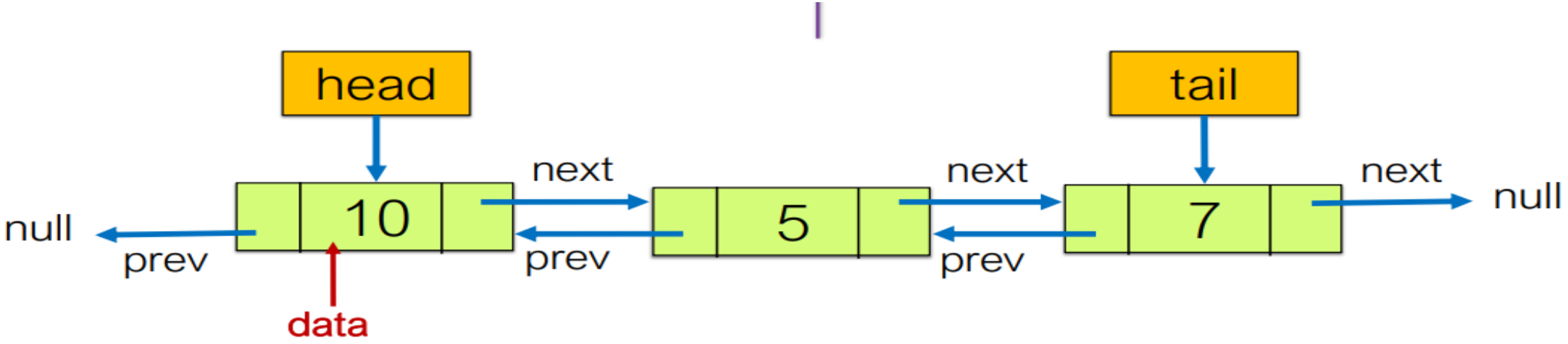
CS202: Data Structures (Fall 2025)

Dr Maryam Abdulghafur, Momina Khan

Department of Computer Science, SBASSE

Quick recap

- A Linked List is a just enough space data structure
- A DLL allows bi-directional traversal
- Both SLL and DLL perform the same for operations that requires list traversal. **Linear.**



Agenda

- Queues
- Circular Queues
- Stacks

Doubly Linked List – Some Observations

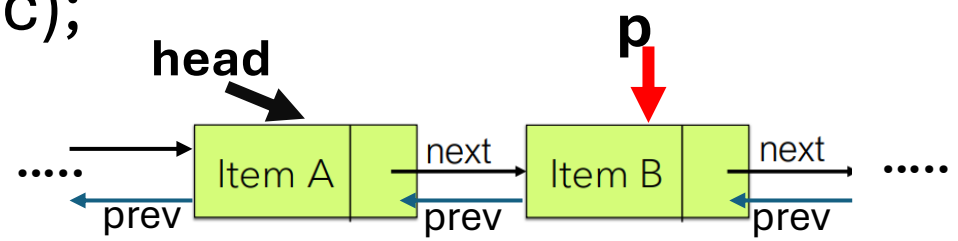
- What is the time complexity (Big O) for
  - inserting an element to the end? **O(1)**
  - inserting an element at the second-last position? **O(1)**
  - deleting an element form the end? **O(1)**
  - deleting an element from the second-last position? **O(1)**
  - Searching an element? **O(N)**
  - Inserting/deleting an element from kth location? **O(N)**

Worth noting that the last is O(N) since it involves searching first!

insert() in a DLL

```
void List<T>::insert(int loc, listNode<T>* newNode) {
        /* where p is  location of node following newNode*/
        listNode* p = find(loc);

        newNode->next = p ;
        newNode->prev = p->prev;
        p ->prev ->next = newNode;
}
```



delete() from a DLL

```
void List<T>::delete(int loc) {
        // p is the node to be deleted
         listNode* p = find(loc);

         p -> prev ->next = p -> next;
         p -> next ->prev = p -> prev;

        delete p;
}
```

Queue ADT

Queue ADT defines a certain linear ordering of data items access.

Where do you think queues will apply?

Applications of Queues

1. Processor scheduling applications

2. Printing jobs at the printer

3. Network data sent out on the internet

4. Load balancing across serves.
(Like that of google search queries!)

Queue ADT

FIFO order!

1. **Enqueue**: Add an element to the rear/back of the queue.
2. **Dequeue**: Remove an element from the front of the queue.
3. **Peek/Front**: Retrieve the element at the front without removing it.
4. **IsEmpty**: Check if the queue is empty.
5. **IsFull** (optional): Check if the queue is full (in case of a fixed-size queue).

A Queue is a collection. Do you feel that it is missing any essential functions a collection must offer?

Design Decision:

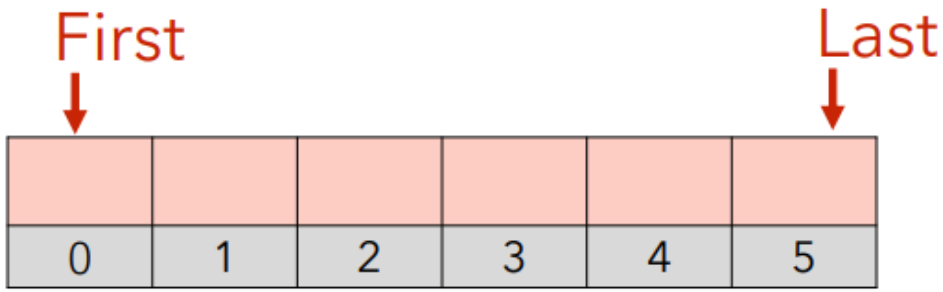
Should we implement with an array, SLL or a DLL?

We can do any!

Queue using an Array

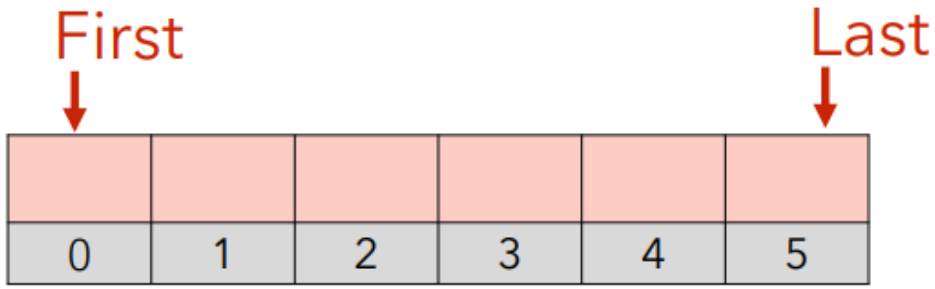
Most recently added item	enqueue	dequeue
At index 0 (first)	O()	O()
Index n (last)	O()	O()

Find the average-case time complexity



Queue using an Array

Most recently added item	enqueue	dequeue
At index 0 (first)	O(N)	O(1)
Index n (last)	O(1)	O(N)



Design Decision:

Some Observations:

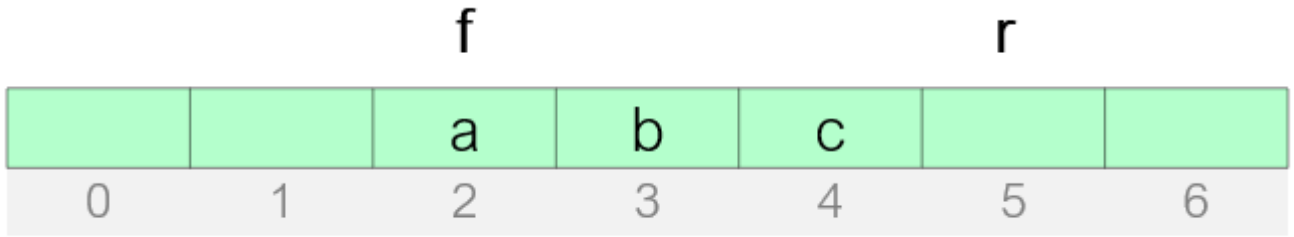
Can we make both **enqueue** and **dequeue** efficient?

Design Decision:

Some Observations:

Can we make both **enqueue** and **dequeue** efficient?

Ans: Circular Queue!



Is it a good idea to grow the array when we still have room at the front of the array? What to do??

Queue – Singly LL based

```
class LinkedListQueue {
    // what will be private members?

    //How to implement the public functions?
    //enqueue()?
    //dequeue()?
    //getSize()?
    .....
}
```

Assumption: We have a tail pointer!

O() Complexity of these operations

- O(1)
- O(1)
- O(1)

Would it help to implement using a DLL??

Stack ADT

Stack ADT defines a certain linear ordering of data items access.

This order is Last In First Out or LIFO!

Applications of Stacks

- Page-visited history in a browser
- Undo sequence in a text editor
- Saving local variables when one function calls another

Interface:

- **push()** // add item to top of stack
- **pop()** // remove item from the top of the stack
- **peek()** // return item from top of stack without removing it

Design Decision:

Should we implement with an array, SLL or a DLL?

We can do any!

What will be private data members of this class?

1) top pointer

2)size variable

Questions???