Comp 352

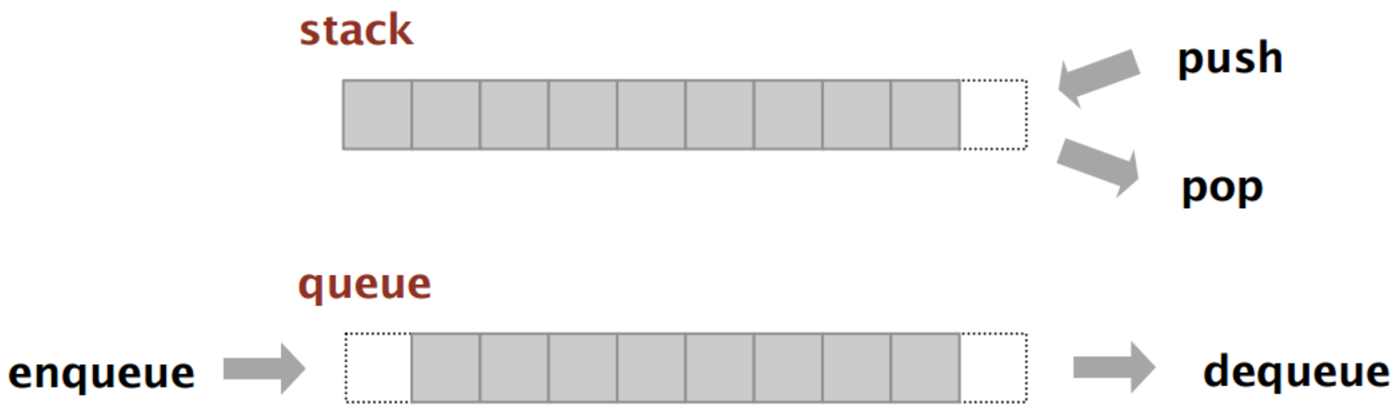
Tutorial Session 4

OUTLINE

- Queues and stacks:
  - List implementation
  - Array implementation
  - Exercise on stack

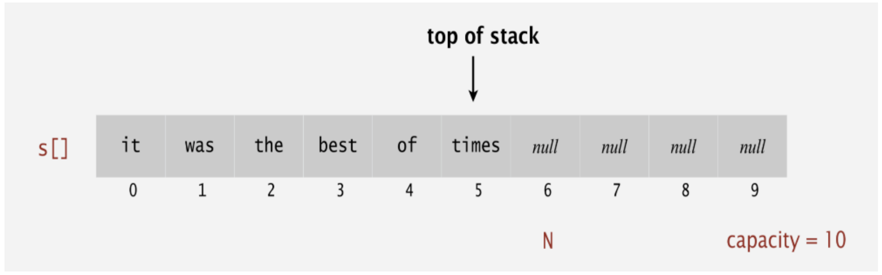
Stack and queue

- Stack: examine the item most recently add □ LIFO= Last In First Out
- Queue: examine the item least recently add □ FIFO= First In First Out



Stack Array implementation

- Simple way to implement stack
- Add element from left to right
- Keep track of the index of the top element



- Problem. Requiring client to provide capacity/ does not implement (a good) API!

Solution with Resizing array: If array is full, create a new array of twice the size, and copy items. halve size of array when array is one-quarter full

STACK SINGLY LINKED LIST IMPLEMENTATION

- Need a class to represent a node in the list

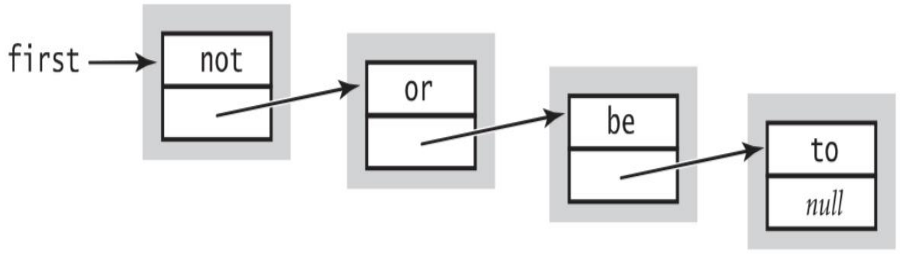
```
Class Node {
        Type data;
        Node next;
}
```

- The list keep track of the head

Stack implementation (con't)

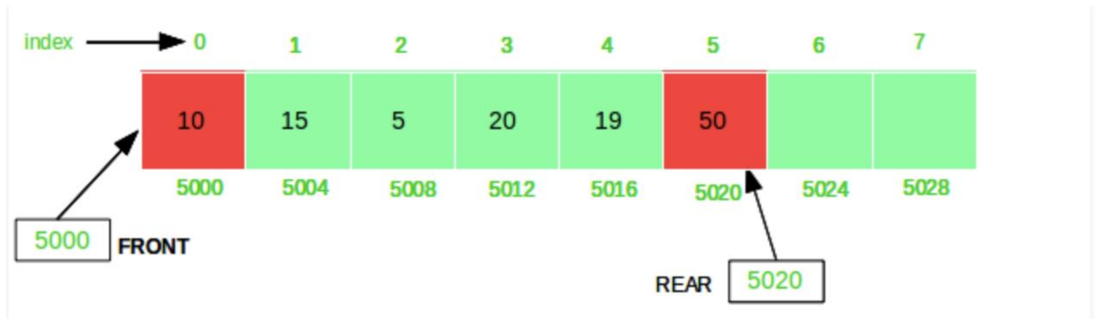
```
Class List{
Node first;
// all operations
}
```

- Add element from right to left.
- Note: one can use doubly linked list to implement stack



Queue: array implementation

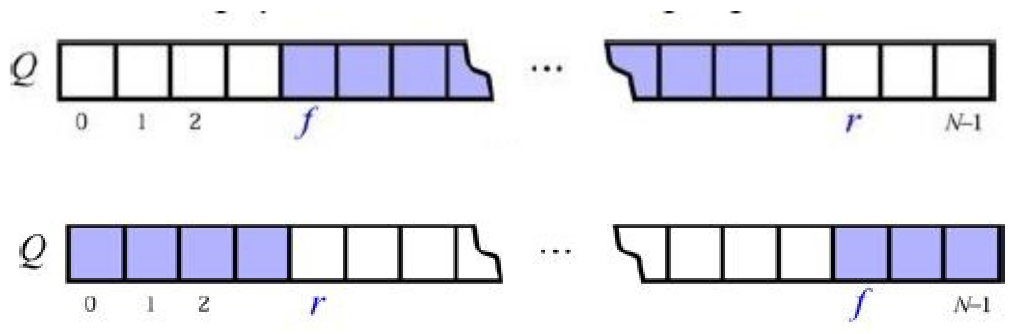
- Simple implementation
- Need to keep track of the index of the front and the rear



With a simple array when we dequeue we need to shift all the elements in the front

Solution: Make the array circular or use a list !!!!!!!

Highlights on a Queue Implementation Using a Circular Array



- The following variables are defined:
  - *f*: index to the cell storing the first element in the queue (candidate to be removed)
  - *r*: index to the next available cell
- Operations:
  - *size()* □ $(N - f + r) \bmod N$
  - *isEmpty()* □ $f = r$
  - *enqueue(x)* □ $r = (r + 1) \bmod N$
  - *dequeue()* □ $f = (f + 1) \bmod N$

Queue: list implementation

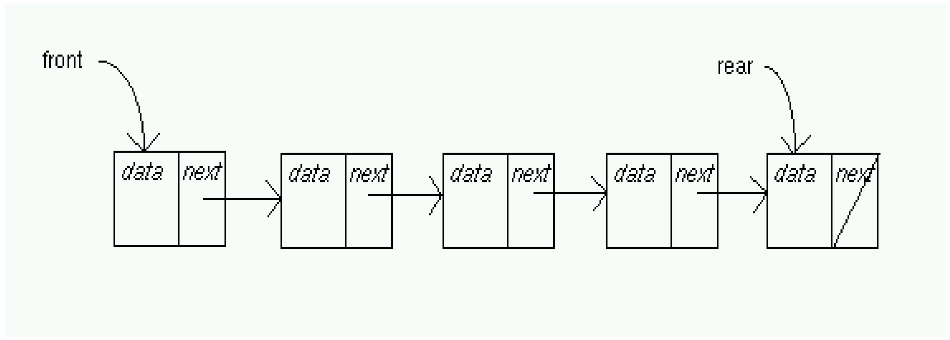
- Use doubly linked list.

Class Node {

Type data;

Node next;

Node previous;}



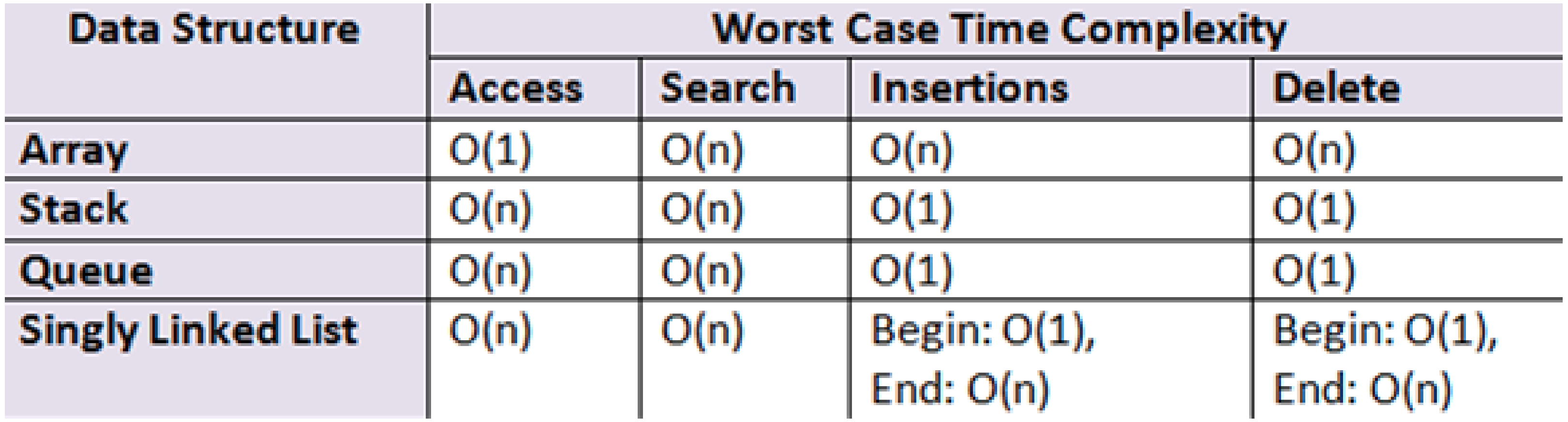
- Need to keep track of the front and rear of the list: *Class List{*

Node first;

Node rear;

// all operations}

WORST CASE TIME COMPLEXITY LINEAR DATA STRUCTURE



Data Structure	Worst Case Time Complexity			
	Access	Search	Insertions	Delete
Array	O(1)	O(n)	O(n)	O(n)
Stack	O(n)	O(n)	O(1)	O(1)
Queue	O(n)	O(n)	O(1)	O(1)
Singly Linked List	O(n)	O(n)	Begin: O(1), End: O(n)	Begin: O(1), End: O(n)

STACK EXERCISES

Question 1:

Suppose an initially empty stack S has performed a total of 25 push operations, 12 top operations, and 10 pop operations, 3 of which generated StackEmptyExceptions, which were caught and ignored. What is the current size of S?

Stack Exercises

Question 2:

Suppose you have a stack in which the values 1 through 5 must be pushed on the stack in that order, but that an item on the stack can be popped at any time. Give a sequence of push and pop operations such that the values are popped in the following order:

a) 2,4,5,3,1
b) 1,5,4,2,3
c) 1,3,5,4,2

It might not be possible in each case.

Stack Exercises

Question 3:

Give a recursive method for removing all the elements in a stack.

Stack Exercises

Question 4:

Write a program that reads in a positive integer and prints the binary representation of that integer. Hint: divide the integer by 2.

Stack Exercise

Question 5:

Given an expression string, write a program to find whether a given string has balanced parentheses or not.

Only consider the parentheses [,],(,),{,}

a) **Input :** {[]{()}} **Output :** Balanced
b) **Input :** [{}{}(] **Output :** Unbalanced

Queue Exercises

Question 6:

Describe the output for the following sequence of queue operations:

enqueue(5), enqueue(3), dequeue(), enqueue(2), enqueue(8), dequeue(), dequeue(), enqueue(9), enqueue(1), dequeue(), enqueue(7), enqueue(6), dequeue(), dequeue(), enqueue(4), dequeue(), dequeue().

Queue Exercises

Question 7:

Suppose an initially-empty queue Q has performed a total of 32 enqueue operations, 10 front operations, and 15 dequeue operations, 5 of which generated QueueEmptyExceptions, which were caught and ignored. What is the current size of Q?

Queue Exercises

Question 8:

Give an algorithm for reversing a queue Q. Only the following standard operations are allowed on queue.

- enqueue(x) : Add an item x to rear of queue.
- dequeue() : Remove an item from front of queue.
- empty() : Checks if a queue is empty or not.

Stack and Queue Exercises

Question 9:

Describe how to implement the stack ADT using two queues.

What is the running time of the push() and pop() methods in this case?