COMP 6481: Programming and Problem Solving

Tutorial 6:

Array List, Linked List, Stack, and Queue

Quick Background

- Array List
- Linked List
- Stack
- Queue

Array List

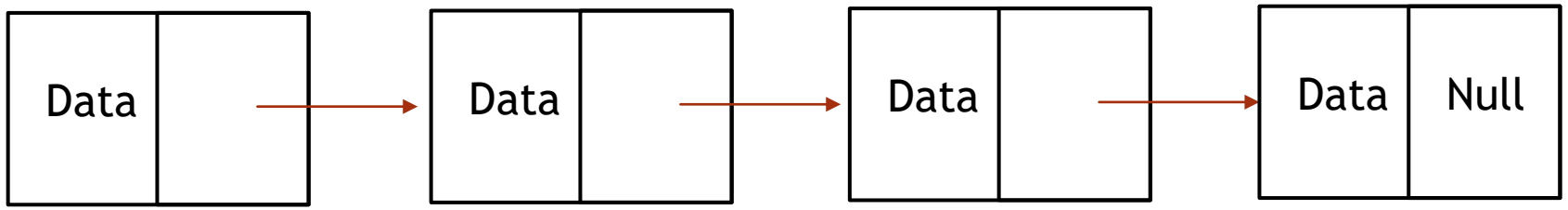
- An ***Array List*** stores its elements in a linear sequential way and supports access to these elements by their indices.
- An ***Array List*** extends an array and is far more superior than it.
- An ***Array List*** grows as items are added to it, it is searchable, sortable, etc.

The Array List ADT - Methods

- *get(i):* Return the element of S with index i; an error condition occurs if $i < 0$ or $i >$ size() − 1.
- *set(i,e):* Replace with e and return the element at index i; an error condition occurs if $i < 0$ or $i >$ size() − 1.
- *add(i,e):* Insert a new element e into S to have index i; an error condition occurs if $i < 0$ or $i >$ size().
- *remove(i):* Remove from S the element at index i; an error condition occurs if $i < 0$ or $i >$ size() − 1.
- *size():* Return the number of elements in the Array List.
- *isEmpty():* Return a Boolean indicating if the Array List is empty.

Linked list

A linked list is a linked data structure consisting of a single chain of nodes, each connected to the next by a link.



Linked list

Each node in the linked list consists of two parts:

1. data, which is consistent across all the nodes of the linked list
2. reference, which is a pointer to the next node.

```
Node {
   private int content;
   private Node nextNode;
   ...
}
```

Linked list

The Node constructor creates a new node and points the one passed via its parameter.

```
Node {
   private int content;
   private Node nextNode;

   Node(int value, Node extern)
   {
      content = value;
      nextNode = extern;
   }
}
```

The Stack ADT – Definition

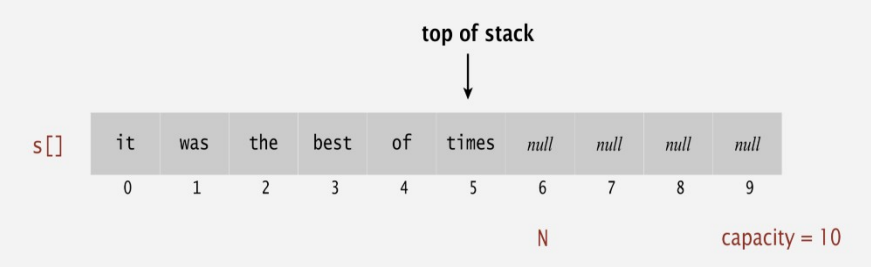
A stack is a collection of objects that are inserted and removed according to the last-in first-out (LIFO) principle. The name “Stack” is derived from the metaphor of stack of plates.

The Stack ADT - Methods

- push(*e*): Insert element *e*, to be the top of the stack.
- pop(): Remove from the stack and return the top element on the stack; an error occurs if the stack is empty.
- size(): Return the number of elements in the stack.
- isEmpty(): Return a Boolean indicating if the stack is empty.
- top(): Return the top element in the stack, without removing it; an error occurs if the stack is empty.

STACK Array/Array List Implementation

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



- Requires client to provide the capacity
- Array List (Solution with Resizing array):
  - Create a new array of twice the size when array is full and copy items.
  - Halve size of array when array is one-quarter full

STACK SINGLY LINKED LIST IMPLEMENTATION (1)

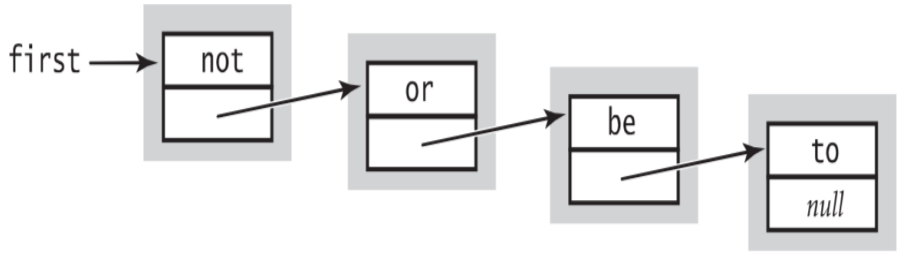
- Need a class to represent a node in the list

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

- The list keeps track of the head

STACK SINGLY LINKED LIST IMPLEMENTATION (2)

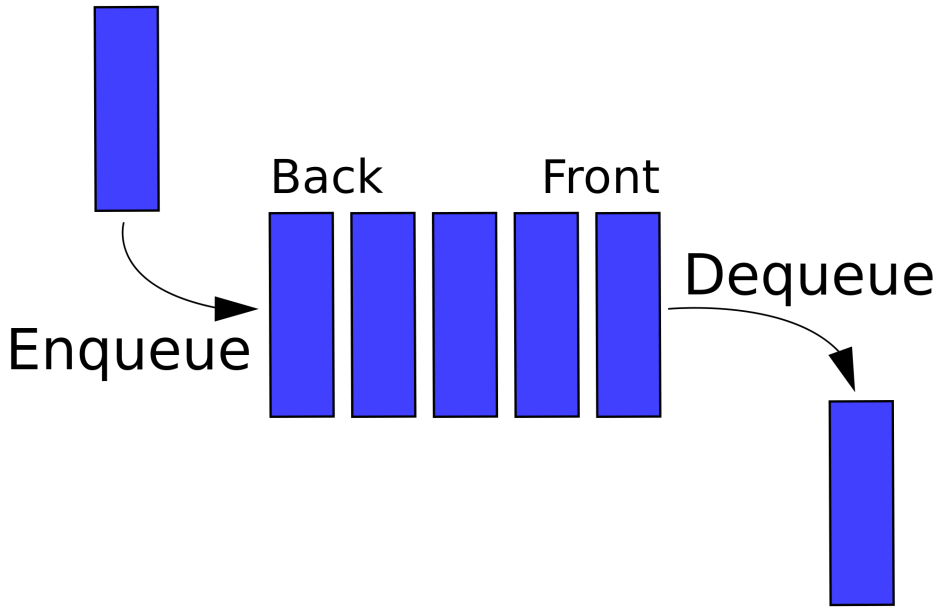
```
Class List{
Node first;
// all operations
}
```



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

The Queue ADT - Definition

Another fundamental data structure is the queue. Objects are inserted and removed according to the first-in first-out (FIFO) principal.

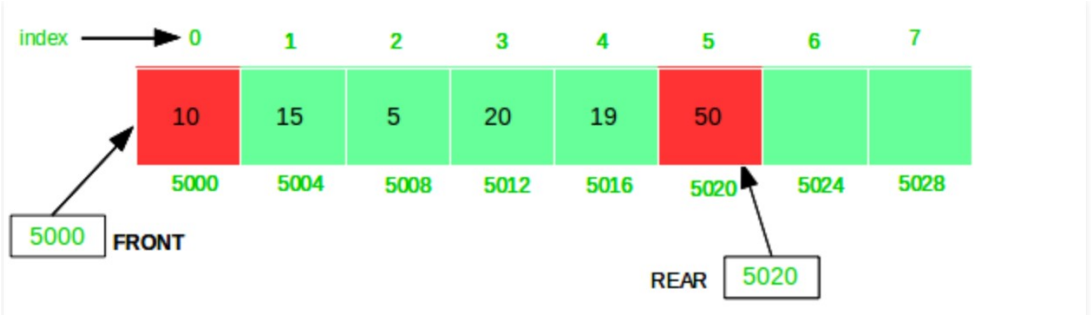


The Queue ADT - Methods

- enqueue(*e*): Insert element *e* at the rear of the queue.
- dequeue(): Remove and return from the queue the object at the front; an error occurs if the queue is empty.
- size(): Return the number of objects in the queue.
- isEmpty(): Return a Boolean value that indicates whether the queue is empty.
- front(): Return, but do not remove, the front object in the queue; an error occurs if the queue is empty.

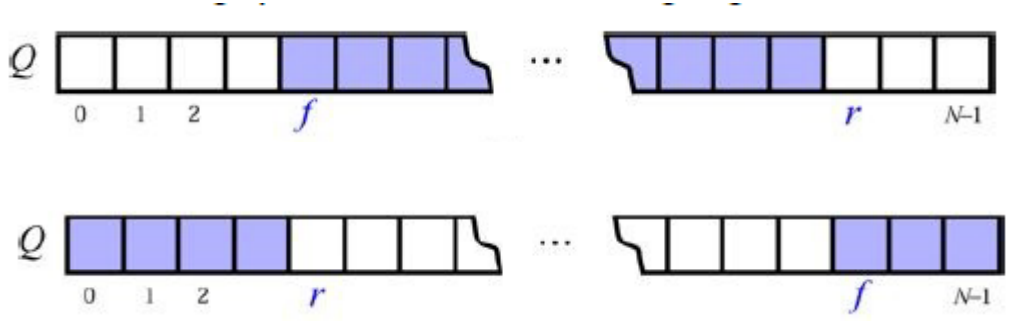
QUEUE: ARRAY IMPLEMENTATION

- 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 an ArrayList

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) modN
  - isEmpty() ▫ f = r
  - enqueue(x) ▫ r = (r + 1) modN
  - dequeue() ▫ f = (f + 1) modN

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)

Question 1

- Describe the output of the following series of stack operations:
- push(5), push(3), pop(), push(2), push(8), pop(), pop(), push(9), push(1), pop(), push(7), push(6), pop(), pop(), push(4), pop(), pop().

Question 2

- 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?

Question 3

- 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.

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.

Question 5

- Write a program that checks whether an input string is a palindrome or not using a stack.

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().

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?

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.

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?