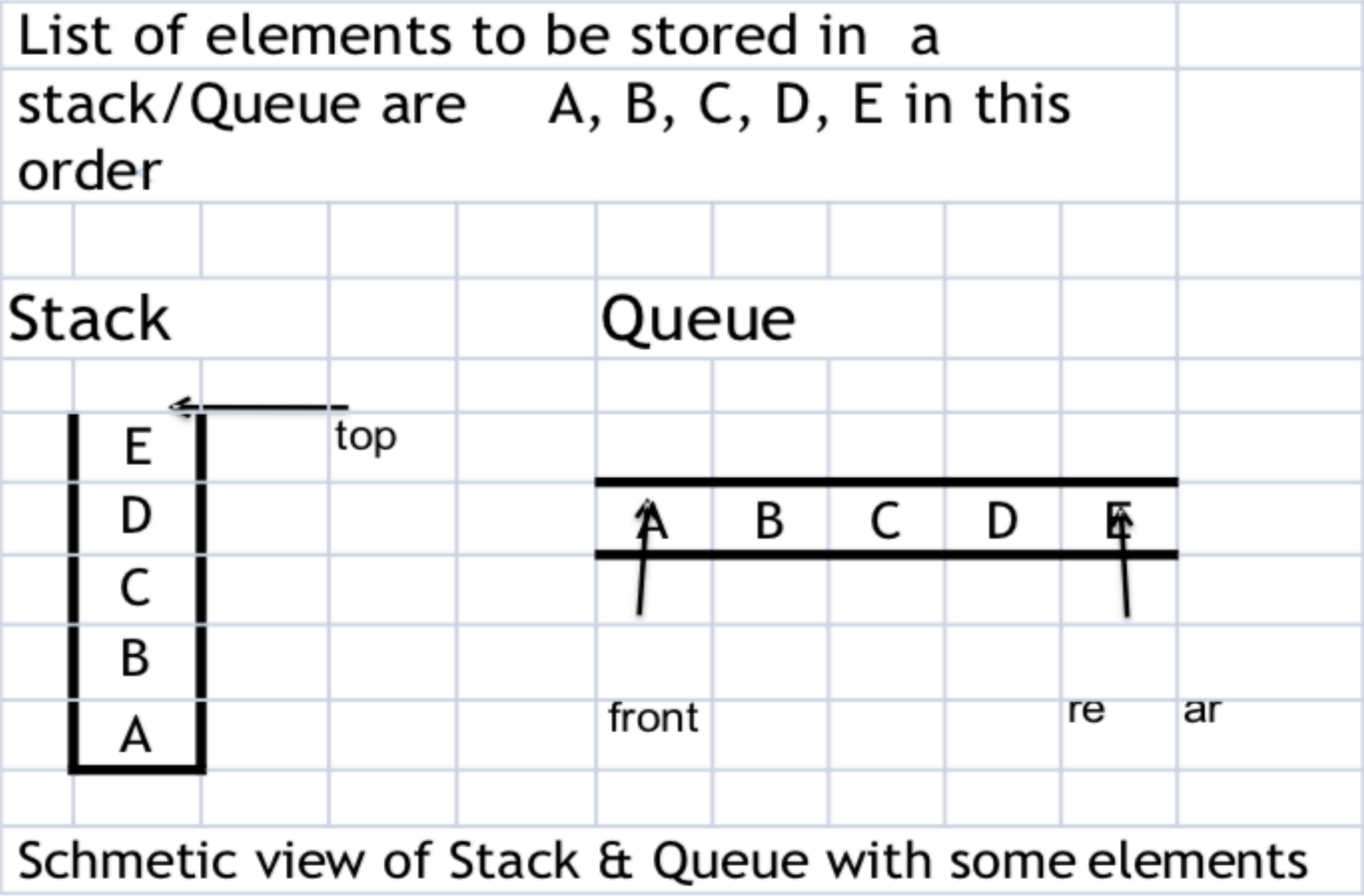
Unit II
Data Structure

Stack & Queue

Elementary Data Structure

Stack & Queue



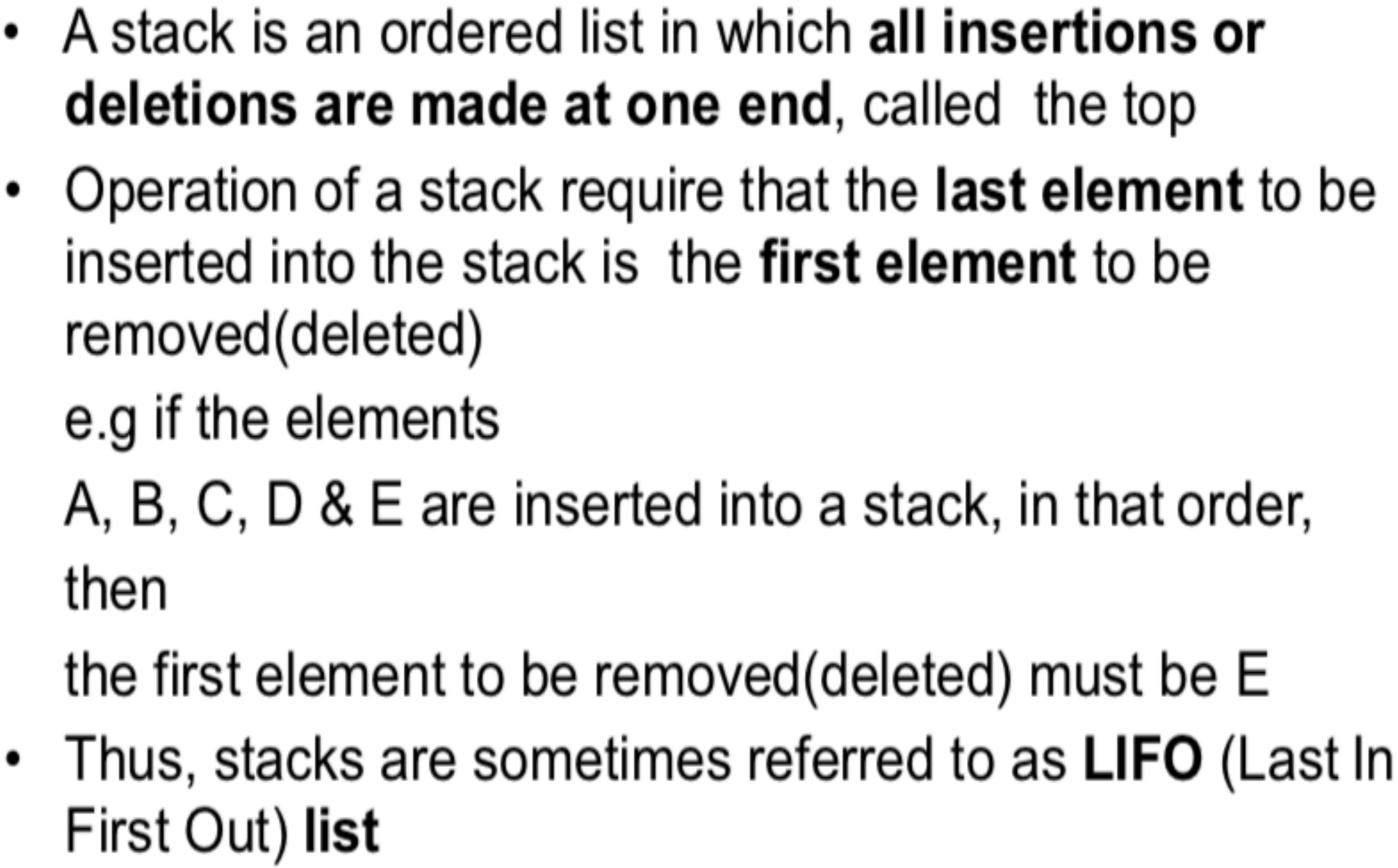
List of elements to be stored in a stack/Queue are A, B, C, D, E in this order

Stack

Queue

Schmetic view of Stack & Queue with some elements

Stack (LIFO List)



- A stack is an ordered list in which **all insertions or deletions are made at one end**, called the top
- Operation of a stack require that the **last element** to be inserted into the stack is the **first element** to be removed(deleted)

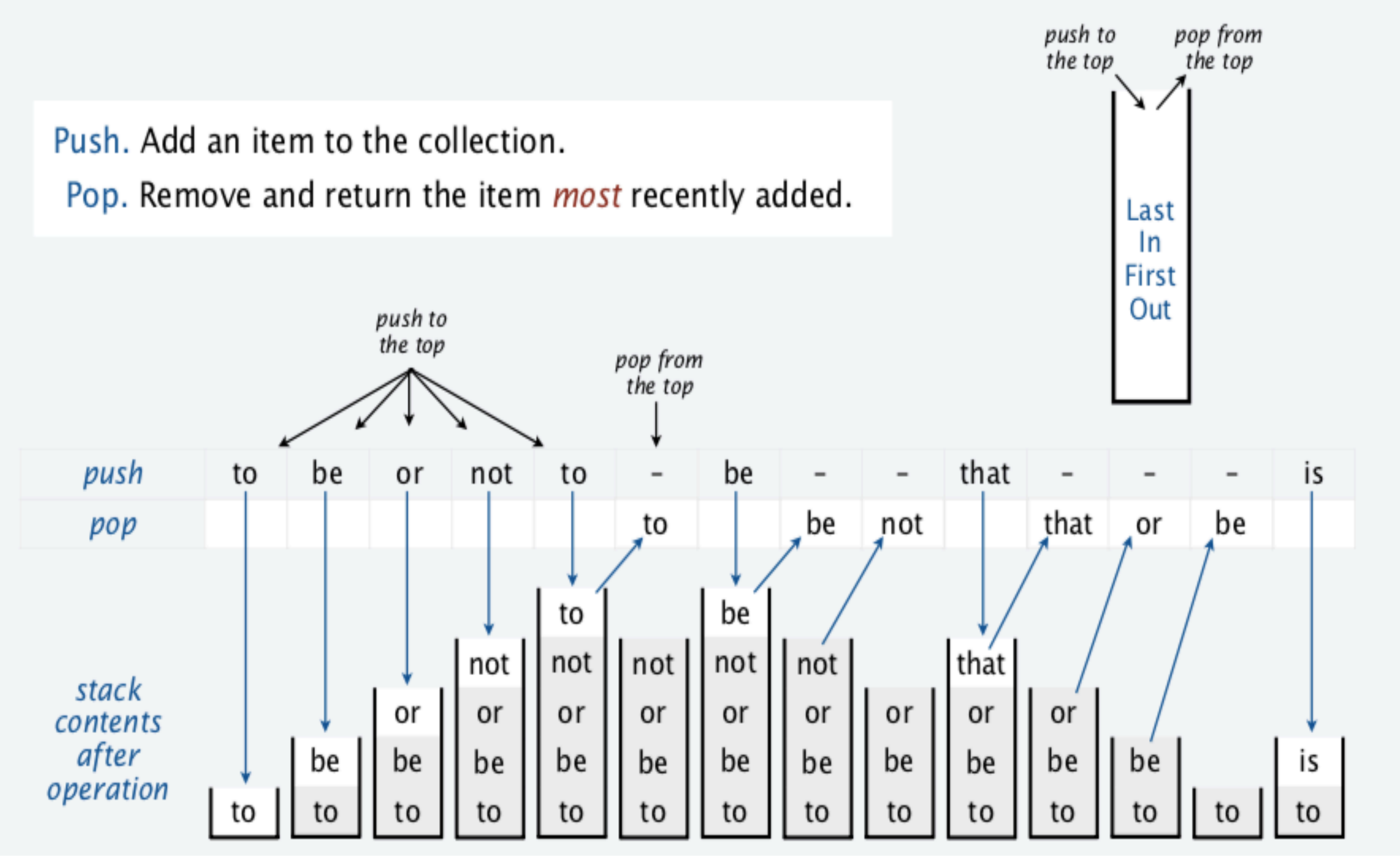
  e.g if the elements

  A, B, C, D & E are inserted into a stack, in that order,

  then

  the first element to be removed(deleted) must be E
- Thus, stacks are sometimes referred to as **LIFO** (Last In First Out) **list**

Example of Stack Operation



Push. Add an item to the collection.

Pop. Remove and return the item *most* recently added.

push	to	be	or	not	to	-	be	-	-	that	-	-	-	is
pop						to		be	not		that	or	be	
stack contents after operation	to	be to	or be to	not or be to	to not or be to	not or be to	be not or be to	not or be to	or be to	that or be to	or be to	be to	to	is to

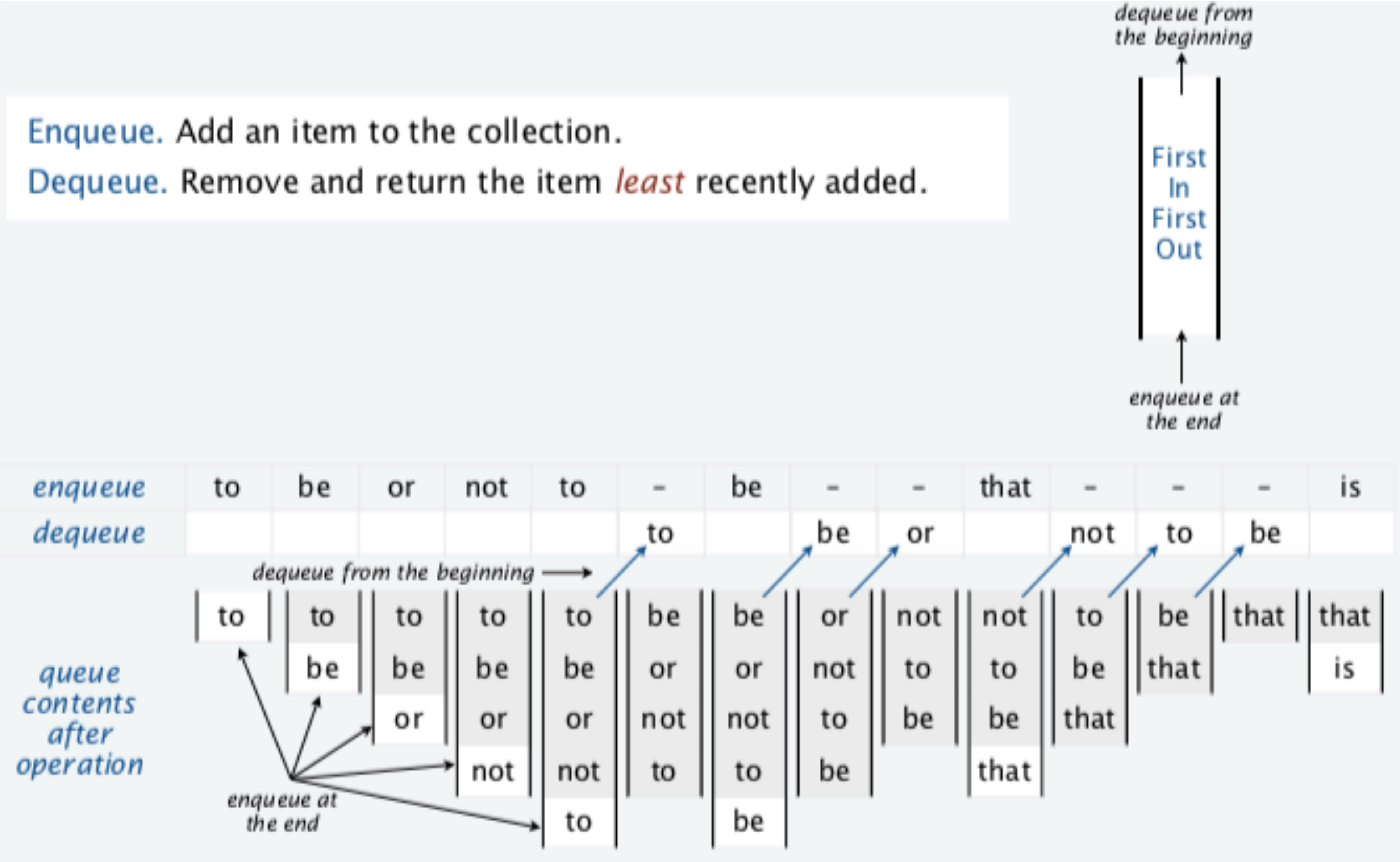
Queue (FIFO List)

- A queue is an ordered list in which **all insertions take place at one end (the rear)** whereas **all deletions are made at the other end(the front)**
- Operation of a queue require that the **first** element to be inserted into the queue is the **first element** to be removed(deleted)

  e.g if the elements
  A, B, C, D & E are inserted into a queue, in that order, then

  the first element to be removed(deleted) must be A
- Thus, queue are sometimes referred to as **FIFO** (First In First Out) **list**

Example of Queue Operation



Enqueue. Add an item to the collection.

Dequeue. Remove and return the item *least* recently added.

enqueue	to	be	or	not	to	-	be	-	-	that	-	-	-	is
dequeue						to		be	or		not	to	be	
queue contents after operation	to	to be	to be or	to be or not	to be or not to	be or not to	be or not to be	or not to be	not to be	not to be that	to be that	be that	that	that is

Stack ADT

push(*e*): Adds element *e* to the top of the stack.

pop(): Removes and returns the top element from the stack (or null if the stack is empty).

Additionally, a stack supports the following accessor methods for convenience:

top(): Returns the top element of the stack, without removing it (or null if the stack is empty).

size(): Returns the number of elements in the stack.

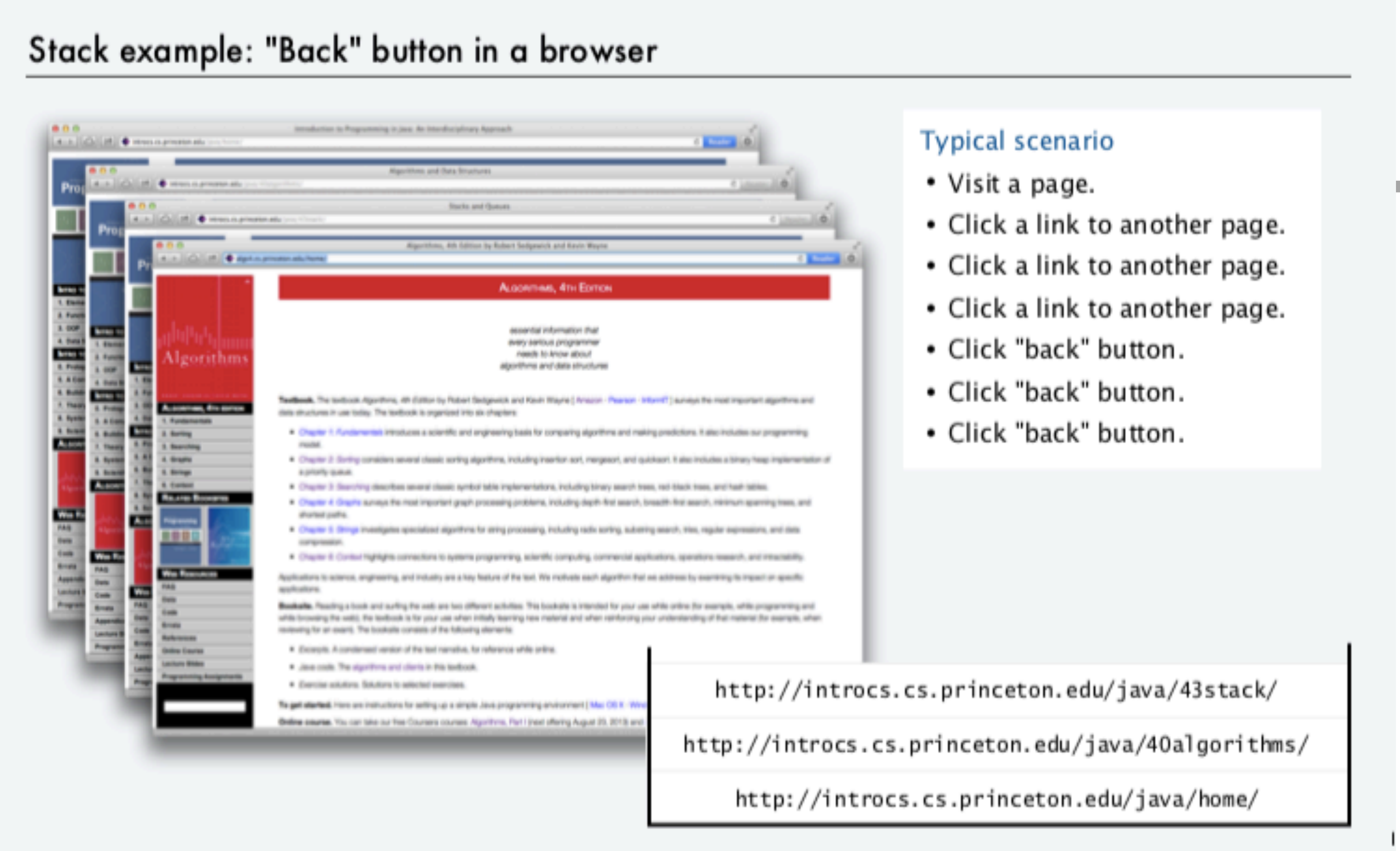
isEmpty(): Returns a boolean indicating whether the stack is e

Stack ADT

The following table shows a series of stack operations and their effects on an initially empty stack *S* of integers.

Operation	Return Value	Stack Content
S.push(5)	-	[5]
S.push(3)	-	[5, 3]
len(S)	2	[5,3]
S.pop()	3	[5]
S.is_empty()	False	[5]
S.pop()	5	[]
S.push(7)	-	[7]
S.push(9)	-	[7,9]
S.top()	9	[7,9]
S.push(4)	-	[7,9,4]
len(S)	3	[7,9,4]

Stack Application



Stack example: "Back" button in a browser

Algorithms, 4th Edition

essential information that
every serious programmer
needs to know about
algorithms and data structures

Typical scenario

- Visit a page.
- Click a link to another page.
- Click a link to another page.
- Click a link to another page.
- Click "back" button.
- Click "back" button.
- Click "back" button.

```
http://introcs.cs.princeton.edu/java/43stack/
http://introcs.cs.princeton.edu/java/40algorithms/
http://introcs.cs.princeton.edu/java/home/
```

Queue ADT

enqueue(e): Add element e to the back of the Queue Q.

dequeue(): Remove and return the first element from queue Q.

Additional Methods:

first(): Return a reference to the front of the queue Q, without removing it; an error occurs if the queue is empty

is_empty(): Returns true if queue Q does not contain any elements.

len(Q): Return the number of elements in queue Q;

Queue ADT

The following table shows a series of Queue operations and their effects on an initially empty Queue *Q* of integers.

Operation	Return Value	First ← Q ← last
Q.enqueue(5)	-	[5]
Q.enqueue(3)	-	[5, 3]
len(Q)	2	[5,3]
Q.dequeue()	5	[3]
Q.is_empty()	False	[3]
Q.dequeue()	3	[]
Q.is_empty()	True	[]
Q.dequeue()	“error”	[]
Q.enqueue(7)	-	[7]
Q.enqueue(9)	-	[7,9]
Q.first()	7	[7,9]
Q.enqueue(4)	-	[7,9,4]
len(Q)	3	[7,9,4]

Stack & Queue

Elementary Data Structure

a. A simple array-based Stack and Queue Implementation
b. Implementing Stack and Queue with a Singly Linked List.