Basic Data Structures

CSCI 1030U - Intro to Computer Science
@IntroCS

Randy J. Fortier
@randy_fortier

OntarioTech UNIVERSITY

Outline

- Basic data structures:
  - Stacks
  - Queues

Stacks

Stacks - Conceptual

```
01   stack.push(7)
02   stack.push(-3)
03   stack.push(1)
04   stack.pop()
05   stack.push(8)
06   stack.pop()
07   stack.pop()
08   stack.pop()
```

Stacks

- We've discussed stacks in a previous lecture section
- A stack is a last in, first out (LIFO) list
- e.g. A stack of papers, books, boxes
- Operations:
  - `top()`: Returns the top of the stack
  - `pop()`: Removes and returns the top of the stack
  - `push()`: Places a new element on top of the stack
  - `isEmpty()`: Returns True if the stack is empty

Stacks in Python

- Lists in Python can be used as stacks:

```
stack = []
stack.append(1)                          # push()
stack.append(2)                          # push()
stack.append(3)                          # push()
print("top:", stack[-1])                 # top()
print("isEmpty?", (len(stack) == 0))     # isEmpty()
print(stack.pop())                       # prints 3
print(stack.pop())                       # prints 2
print(stack.pop())                       # prints 1
```

Stacks - Implementation

- Arrays
  - Use an array to store the values
  - Use an integer variable (top) to store the index of the next available space
- Linked list
  - Use a set of node elements:
    - A value
    - A pointer to the next node (or null, if none)
  - Use a pointer (top) which points to the element at the top of the stack

Stacks - Array Implementation

```
01   stack.push(7)
02   stack.push(-3)
03   stack.push(1)
04   stack.pop()
05   stack.push(8)
06   stack.pop()
07   stack.pop()
08   stack.pop()
```

Stacks - Discussion

- What is the key disadvantage of using arrays to implement a stack?

- What is the key advantage?

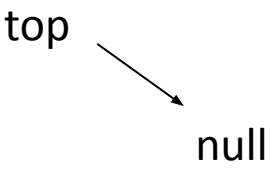
Pointers

- A pointer stores the address of a value in memory
  - Unlike a non-pointer variable, which refers to the value directly, a pointer variable refers to the address of that value
- Pointers take time to learn
  - Dealing with memory addresses involves ensuring that:
    - The memory address is valid and accessible
    - The memory address lines up with the starting boundary of the value
  - For this reason, many programming languages (e.g. Python) do not have pointers (but do have references)

Stacks - Linked List Implementation

```
01   stack.push(7)
02   stack.push(-3)
03   stack.push(1)
04   stack.pop()
05   stack.push(8)
06   stack.pop()
07   stack.pop()
08   stack.pop()
```

Stack - Linked List Implementation

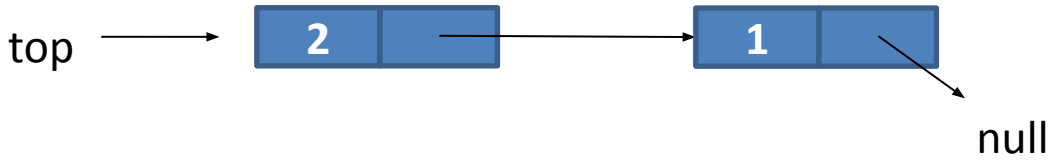



```
stack.push(1)
stack.push(2)
stack.push(3)
stack.pop()
stack.pop()
stack.pop()
```

Stack - Linked List Implementation

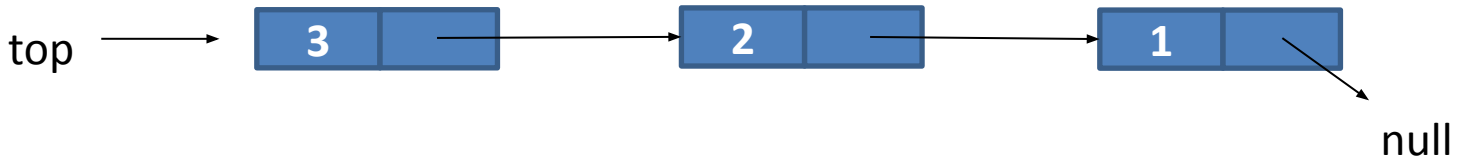

```
stack.push(1)
stack.push(2)
stack.push(3)
stack.pop()
stack.pop()
stack.pop()
```

Stack - Linked List Implementation



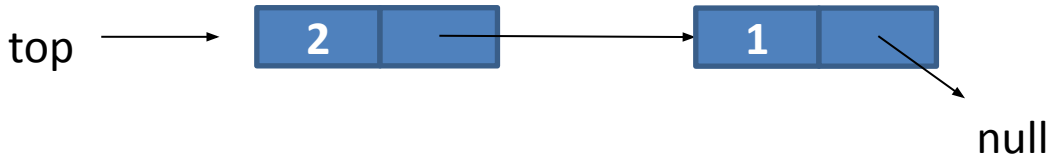

```
stack.push(1)
stack.push(2)
stack.push(3)
stack.pop()
stack.pop()
stack.pop()
```

Stack - Linked List Implementation




```
stack.push(1)
stack.push(2)
stack.push(3)
stack.pop()
stack.pop()
stack.pop()
```

Stack - Linked List Implementation

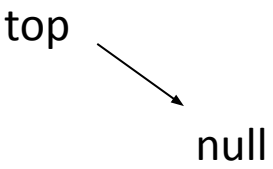



```
stack.push(1)
stack.push(2)
stack.push(3)
stack.pop()
stack.pop()
stack.pop()
```

Stack - Linked List Implementation


```
stack.push(1)
stack.push(2)
stack.push(3)
stack.pop()
stack.pop()
stack.pop()
```

Stack - Linked List Implementation




```
stack.push(1)
stack.push(2)
stack.push(3)
stack.pop()
stack.pop()
stack.pop()
```

Stacks - Discussion

- What is the key disadvantage of using linked lists to implement a stack?

- What is the key advantage?

Coding Exercise 10a.1

- Create an implementation of a stack using Python's built-in list type
- Be sure to implement the following methods:
  - `push()`
  - `pop()`
  - `top()`
  - `is_empty()`

Queues

OntarioTech UNIVERSITY

Queues - Conceptual

```
01   queue.enqueue(7)
02   queue.enqueue(-3)
03   queue.enqueue(1)
04   queue.dequeue()
05   queue.enqueue(8)
06   queue.dequeue()
07   queue.dequeue()
08   queue.dequeue()
```

Queues

- A queue is sort of the reverse of a stack
  - It is a first in, first out (FIFO) structure
- e.g. A waiting list for a parking lot on campus
- Operations:
  - `enqueue()`: Adds an element to the back of the queue
  - `front()`: Returns the front element
  - `dequeue()`: Removes the front element
  - `isEmpty()`: Returns True if the queue is empty

Queues - Python

- Lists in Python can be used as queues:

```
queue = []
queue.append(1)                          # enqueue()
queue.append(2)                          # enqueue()
queue.append(3)                          # enqueue()
print("front:", queue[0])                # front()
print("isEmpty?", (len(queue) == 0))     # isEmpty()
print(queue.pop(0))                      # prints 1
print(queue.pop(0))                      # prints 2
print(queue.pop(0))                      # prints 3
```

Queues - Implementation

- Arrays
  - Use an array to store the values
  - Use an integer variable (back) to store the index of the next available space
  - Use an integer variable (front) to store the index of the element at the front of the list (FIFO)
- Linked list
  - Use a set of node elements:
    - A value
    - A pointer to the next node (or null, if none)
  - Use a pointer (front), which points to the element at the front of the queue
  - Use a pointer (back), which points to the element at the back of the queue

Queues - Array Implementation

```
01   queue.enqueue(7)
02   queue.enqueue(-3)
03   queue.enqueue(1)
04   queue.dequeue()
05   queue.enqueue(8)
06   queue.dequeue()
07   queue.dequeue()
08   queue.dequeue()
```

Queues - Array Implementation


```
queue.enqueue(1)
queue.enqueue(2)
queue.enqueue(3)
queue.dequeue()
queue.dequeue()
queue.dequeue()
```

Queues - Array Implementation


```
queue.enqueue(1)
queue.enqueue(2)
queue.enqueue(3)
queue.dequeue()
queue.dequeue()
queue.dequeue()
```

Queues - Array Implementation


```
queue.enqueue(1)
queue.enqueue(2)
queue.enqueue(3)
queue.dequeue()
queue.dequeue()
queue.dequeue()
```

Queues - Array Implementation

front

back

```
queue.enqueue(1)
queue.enqueue(2)
queue.enqueue(3)
queue.dequeue()
queue.dequeue()
queue.dequeue()
```

Queues - Array Implementation

front

back

```
queue.enqueue(1)
queue.enqueue(2)
queue.enqueue(3)
queue.dequeue()
queue.dequeue()
queue.dequeue()
```

Queues - Array Implementation

front back

```
queue.enqueue(1)
queue.enqueue(2)
queue.enqueue(3)
queue.dequeue()
queue.dequeue()
queue.dequeue()
```

Queues - Array Implementation

front back

```
queue.enqueue(1)
queue.enqueue(2)
queue.enqueue(3)
queue.dequeue()
queue.dequeue()
queue.dequeue()
```

Queues - Discussion

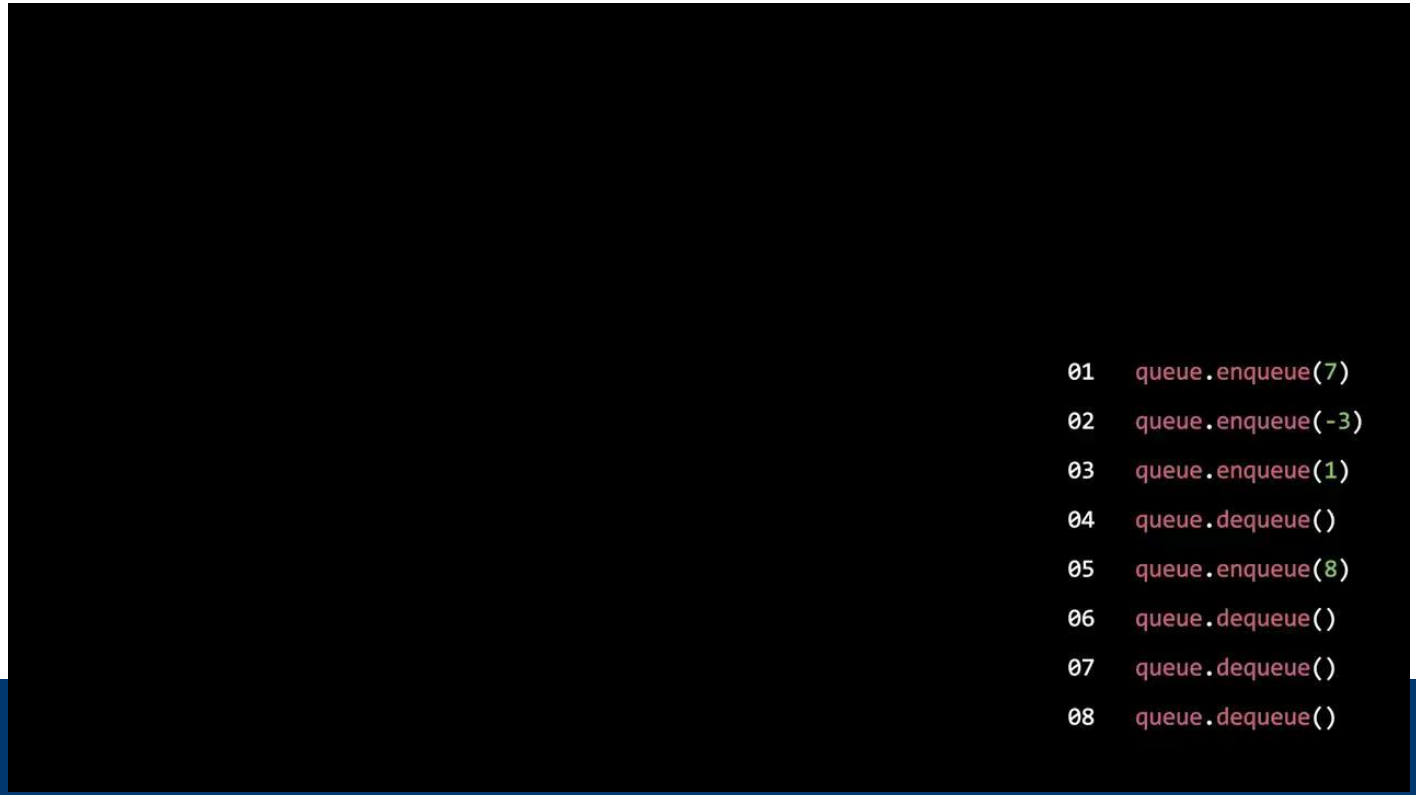
- What is the key disadvantage of using arrays to implement a queue?

- What is they key advantage?

Queues - Linked List Implementation

- As you can see, when arrays are used to implement queues, we have a shift to the right of both pointers
  - Eventually we will run out of space in the array/list
  - The solution is to let both pointers wrap around again to the beginning of the array/list

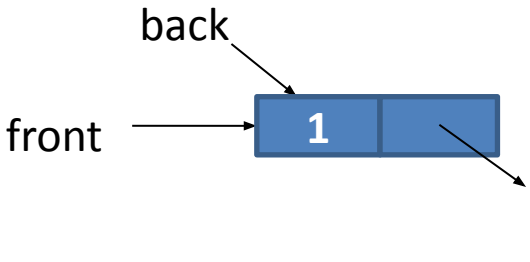
Queues - Linked List Implementation



Queues - Linked List Implementation

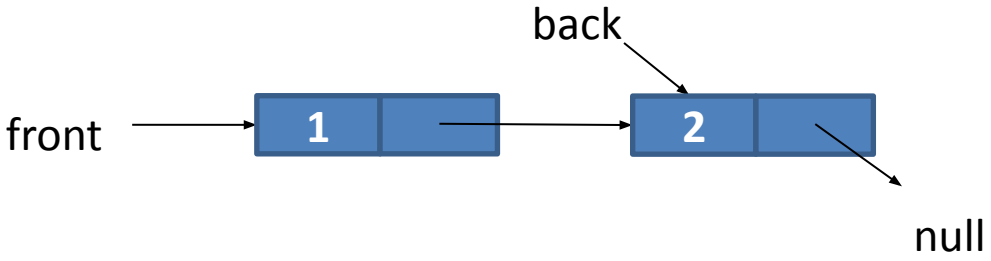

```
queue.enqueue(1)
queue.enqueue(2)
queue.enqueue(3)
queue.dequeue()
queue.dequeue()
queue.dequeue()
```

Queues - Linked List Implementation



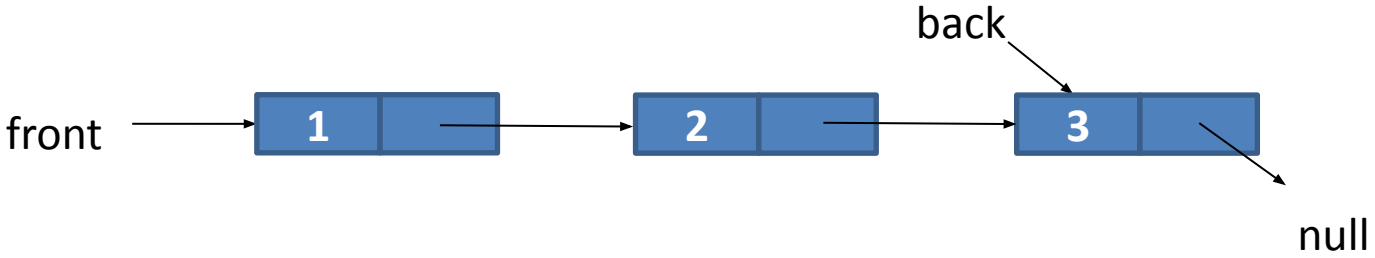

```
queue.enqueue(1)
queue.enqueue(2)
queue.enqueue(3)
queue.dequeue()
queue.dequeue()
queue.dequeue()
```

Queues - Linked List Implementation



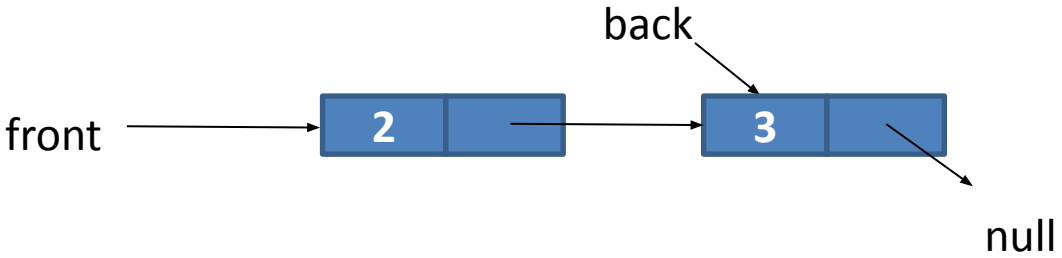

```
queue.enqueue(1)
queue.enqueue(2)
queue.enqueue(3)
queue.dequeue()
queue.dequeue()
queue.dequeue()
```

Queues - Linked List Implementation



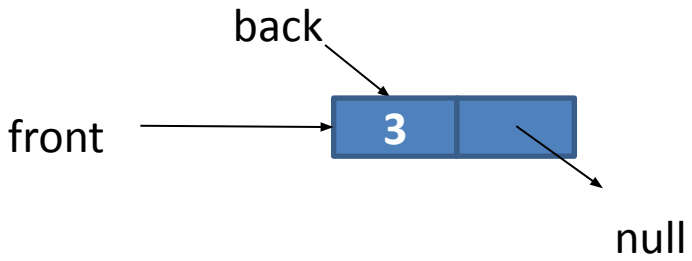

```
queue.enqueue(1)
queue.enqueue(2)
queue.enqueue(3)
queue.dequeue()
queue.dequeue()
queue.dequeue()
```

Queues - Linked List Implementation




```
queue.enqueue(1)
queue.enqueue(2)
queue.enqueue(3)
queue.dequeue()
queue.dequeue()
queue.dequeue()
```

Queues - Linked List Implementation




```
queue.enqueue(1)
queue.enqueue(2)
queue.enqueue(3)
queue.dequeue()
queue.dequeue()
queue.dequeue()
```

Queues - Linked List Implementation


```
queue.enqueue(1)
queue.enqueue(2)
queue.enqueue(3)
queue.dequeue()
queue.dequeue()
queue.dequeue()
```

Queues - Discussion

- What is the key disadvantage of using linked lists to implement a queue?

- What is the key advantage?

Priority Queues

- A variation on queues is a priority queue
  - Each element in the queue has a priority value
  - The values with the highest priority value are selected first
- Implementations:
  - Sorted list/array
  - Unsorted list/array
  - More advanced data structures (heaps)
- Heaps will be covered in CSCI 2010U

Wrap-up

- Basic data structures:
  - Stacks
  - Queues

Coming Up

- Advanced data structures:
  - Trees
  - Binary trees
    - Binary tree implementations
  - Binary search trees
    - Insert
    - Delete