STACKS & QUEUES

Ryan Rusich
rusichr@cs.ucr.edu

Department of Computer Science
University of California, Riverside

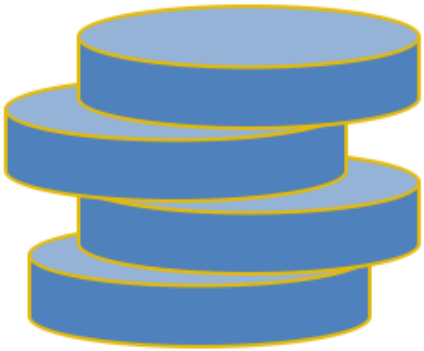
CS014: Intro to
Data Structures
and Algorithms

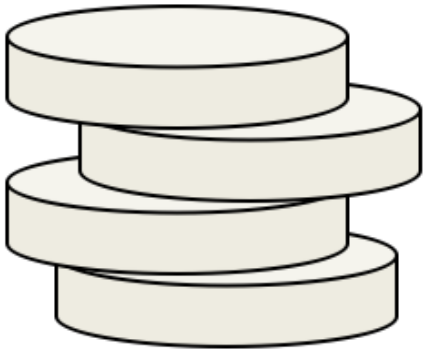
ADT - Abstract Data Type

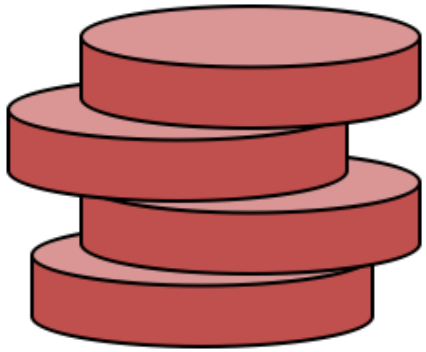
Definition: An abstract data type (ADT) is mathematical model of the set of objects that make-up a data type along with the set of operations allowed on those objects.

- An ADT is a contract between the user of a data structure and its implementer.
- An ADT specifies:
  - type of data stored
  - available methods/functions, with parameter and return types
  - error conditions associated with methods
  - performance guarantees, in terms of space and/or time

Stack







Stack ADT

Operation	Description	Stack S = 9, 4, 5
push(**o**)	insert item onto top of stack	push(1), S = 1, 9, 4, 5
item top( )	read top item on stack	top( ) == 1
pop()	remove top item from stack	pop( ), pop( ), S = 4, 5
int size( )	return number of items in stack	size( ) == 2
boolean isEmpty( )	checks if stack empty	isEmpty( ) == false

Table: LIFO: *Last-In-First-Out*

Stack pseudocode (array)

Algorithm push(***item***):
  if size( ) == N **then**
    throw *StackFullException*
  $t \leftarrow (t + 1)$
  $S[t]$= ***item***

Algorithm pop():
  if isEmpty() **then**
    throw *StackEmptyException*
  $t \leftarrow (t - 1)$

Algorithm size( ):
  return $t + 1$

Algorithm isEmpty( ):
  return $(t < 0)$

Algorithm top( ):
  if isEmpty( ) **then**
    throw *StackEmptyException*
  return $S[t]$

Stack: Running Time

Operation	Description	Running Time
push(**o**)	insert item onto top of stack	O(1)
item top( )	read top item on stack	O(1)
pop()	remove top item from stack	O(1)
int size( )	return number of items in stack	O(1)
boolean isEmpty( )	checks if stack empty	O(1)

Table: LIFO: *Last-In-First-Out*

Queue

Figure: Ford Model T assembly line, 1926. *Car and Driver*

Queue


I'm here for a new driver's license.
DMV
Okay. You're number two.
DMV
TWO!? Nice!!
DMV
NOW SERVING:
0.001271
DMV
Ryan Kramer
TOONHOLE.COM


01
Now Serving

Queue ADT

Operation	Description	Queue Q = 4, 5
enqueue(**o**)	insert item into end of queue	push(1), Q = 4, 5, 1
item front( )	read front item of queue	front( ) == 4
dequeue( )	remove front item from queue	pop( ), Q = 5, 1
int size( )	return number of items in queue	size( ) == 2
boolean isEmpty( )	checks if queue empty	isEmpty( ) == false

Table: FIFO: *First-In-First-Out*

Queue pseudocode (array)

Algorithm size( ):
  return $(N - f + r) \bmod N$

Algorithm isEmpty( ):
  return $(f == r)$

Algorithm front( ):
  if isEmpty() **then**
    throw *QueueEmptyException*
  return Q[f]

Algorithm enqueue(o):
  if size( ) == N–1 **then**
    throw *QueueFullException*
  $Q[r] \leftarrow o$
  $r \leftarrow (r + 1) \bmod N$

Algorithm dequeue( ):
  if isEmpty( ) **then**
    throw *QueueEmptyException*
  $f \leftarrow (f + 1) \bmod N$

Queue: Running Time

Operation	Description	Running Time
enqueue(**o**)	insert item into end of queue	O(1)
item front( )	read front item of queue	O(1)
dequeue( )	remove front item from queue	O(1)
int size( )	return number of items in queue	O(1)
boolean isEmpty( )	checks if queue empty	O(1)

Table: FIFO: *First-In-First-Out*