CSC 212
Data Structures and Abstractions
Spring 2016

Lecture 08: Stacks and Queues
Previously ...

More on Pointers

Linked Lists
Today …

More on Linked Lists
Stacks
Queues
Singly Linked List

Head

Tail

5 -> 7 -> 2 -> 6

NULL pointer
Doubly Linked List

Head

5

7

2

6

Tail

NULL pointer

NULL pointer
Circular Lists

Can also have a circular doubly linked list
Basic Methods

**InsertKey**
- front, end, by index, by value

**DeleteKey**
- front, end, by index, by value

**GetKey/Search**
- by value, by index

**Traverse**
- visit all nodes (e.g. print)

**Destroy**

When implementing methods for linked lists, always draw the state of the linked list.
Stacks

LIFO Last In First Out
Stacks

Push 5

LIFO Last In First Out
Stacks

Push 5
Push 3

LIFO Last In First Out
Stacks

LIFO Last In First Out

Push 5
Push 3
Pop

5
Stacks

LIFO Last In First Out

Push 5
Push 3
Pop
Push 9

[9]
[5]
Stacks

LIFO Last In First Out
Stacks

Push 5
Push 3
Pop
Push 9
Push 8
Push 2

LIFO Last In First Out
Stacks

LIFO Last In First Out

Push 5
Push 3
Pop
Push 9
Push 8
Push 2
Pop

8
9
5
Stacks

LIFO Last In First Out
Stacks

LIFO Last In First Out

Push 5
Push 3
Pop
Push 9
Push 8
Push 2
Pop
Pop
Pop

5
Stack Implementation

Arrays

can be fixed-size or **dynamic array**

**push** and **pop** at the end of array
Stack Implementation

Arrays can be fixed-size or **dynamic array**

**push** and **pop** at the end of array

$O(1)$

amortized cost for dynamic arrays
Stack Implementation

Arrays

can be fixed-size or **dynamic array**

**push** and **pop** at the end of array

Linked Lists

**push** — insert at one end

**pop** — delete from the same end (needs to return value)

$O(1)$ amortized cost for dynamic arrays
Stack Implementation

Arrays
- can be fixed-size or dynamic array
- `push` and `pop` at the end of array

Linked Lists
- `push` — insert at one end
- `pop` — delete from the same end (needs to return value)
Considerations

Underflow

happens when \texttt{pop} from an empty stack
Considerations

Underflow
happens when \texttt{pop} from an empty stack

Overflow
might happen with fixed-length stacks
better use dynamic arrays or linked lists
Applications

Undo in software applications

Navigation buttons in browsers

Stack in compilers/programming languages

Parsing expressions

...
Example

Try with … fully parenthesized infix expressions

$$((5 + ((10 - 4) * (3+2))) + 25)$$
Example

Try with ... fully parenthesized infix expressions

\[((5 + ((10 - 4) \times (3+2))) + 25)\]

Two-Stack Algorithm [Dijkstra]
Try with …

\[((5 + ((10 - 4) \times (3+2))) + 25)\]

fully parenthesized infix expressions

Two-Stack Algorithm [Dijkstra]

value? push onto s1
Example

Try with … fully parenthesized infix expressions

\[(5 + ((10 - 4) \times (3+2))) + 25\]

Two-Stack Algorithm [Dijkstra]

value? push onto s1
operator? push onto s2
Example

Try with … fully parenthesized infix expressions

\[((5 + ((10 - 4) * (3+2))) + 25)\]

Two-Stack Algorithm [Dijkstra]
value? push onto s1
operator? push onto s2
left parenthesis? ignore
Example

Try with …

\[((5 + ((10 - 4) * (3+2))) + 25)\]

Two-Stack Algorithm [Dijkstra]

value? push onto s1
operator? push onto s2
left parenthesis? ignore
right parenthesis? pop two values from s1 and one operator from s2, apply operator and push resulting value to s1
Queues

FIFO First In First Out
Queues

Enqueue 5

FIFO First In First Out
Enqueue 5
Enqueue 3

Queues

FIFO First In First Out
Queues

Enqueue 5
Enqueue 3
Dequeue

FIFO First In First Out
Queues

Enqueue 5
Enqueue 3
Dequeue
Enqueue 9

3
9

FIFO First In First Out
Enqueue 5
Enqueue 3
Dequeue
Enqueue 9
Enqueue 8

Queues
FIFO First In First Out
Queues

Enqueue 5
Enqueue 3
Dequeue
Enqueue 9
Enqueue 8
Enqueue 2

FIFO First In First Out
Queues

Enqueue 5
Enqueue 3
Dequeue
Enqueue 9
Enqueue 8
Enqueue 2
Dequeue

9
8
2

FIFO First In First Out
Queues

Enqueue 5
Enqueue 3
Dequeue
Enqueue 9
Enqueue 8
Enqueue 2
Dequeue
Dequeue

FIFO First In First Out
Queues

Enqueue 5
Enqueue 3
Dequeue
Enqueue 9
Enqueue 8
Enqueue 2
Dequeue
Dequeue
Dequeue

FIFO First In First Out

2
Queue Implementation

Arrays

can be fixed-size or **dynamic array**

**enqueue** and **dequeue** (tricky to implement)
Queue Implementation

Arrays

- can be fixed-size or **dynamic array**
- **enqueue** and **dequeue** (tricky to implement)

Linked Lists

- **enqueue** — insert at one end
- **dequeue** — delete from the other end (needs to return value)
Queue Implementation

Arrays

can be fixed-size or **dynamic array**

enqueue and dequeue (tricky to implement)

Linked Lists

enqueue — insert at one end

dequeue — delete from the other end (needs to return value)
Queue Implementation

Arrays

- can be fixed-size or **dynamic array**
- enqueue and dequeue (tricky to implement)  \(O(1)\)

Linked Lists

- enqueue — insert at one end  \(O(1)\)
- dequeue — delete from the other end (needs to return value)
Applications

Media Playlists (Youtube, Spotify, Music, etc.)

Process management in Operating Systems

Simulations

Used in other algorithms

...