CS 32: Discussion 1D

TA: Shichang Zhang
LA: Stephanie Doan, Rish Jain

Announcements

- Homework 2 due 11 pm Tuesday (April 27th)

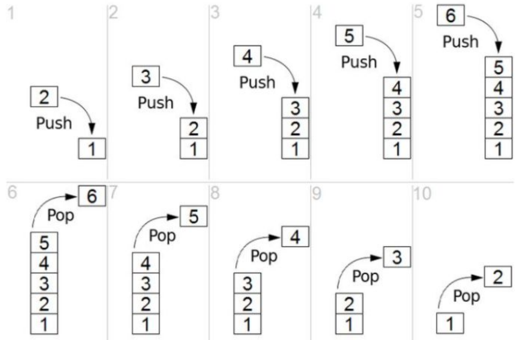
Overview

- Stacks
  - DFS
- Queues
  - BFS

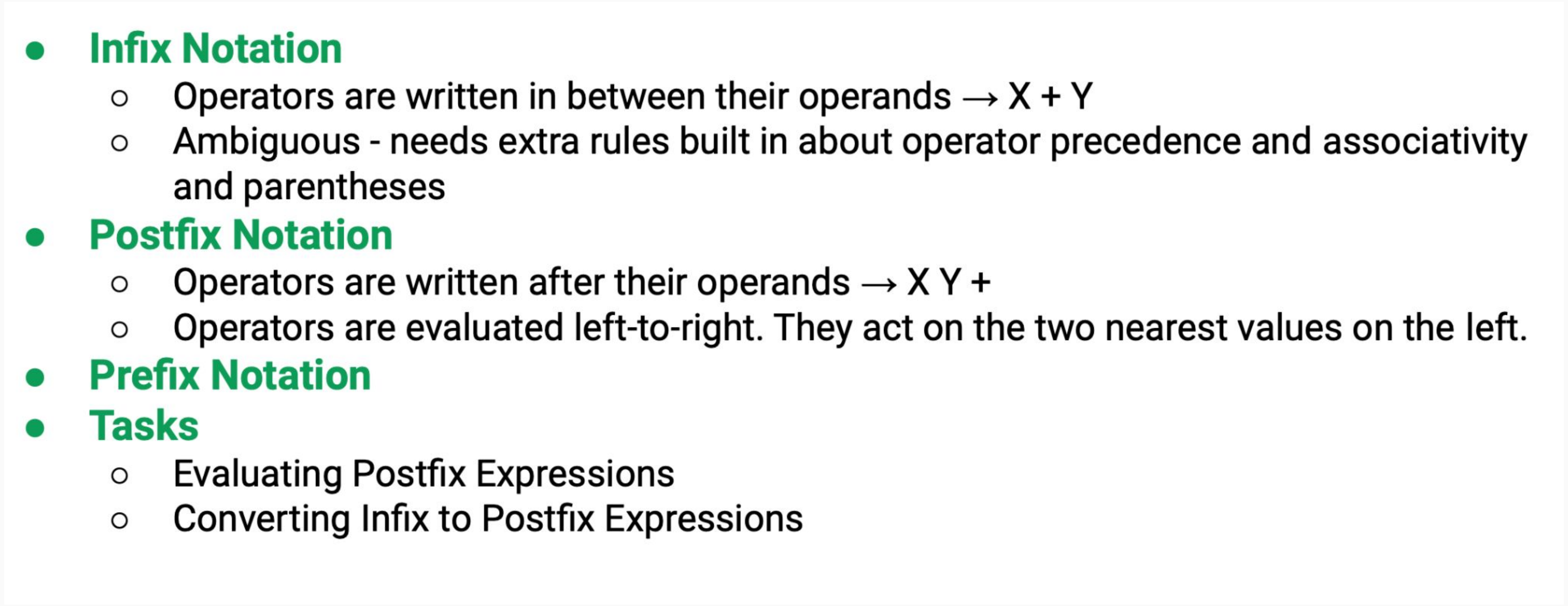
Stacks

- FILO: First In, Last Out
- A standard stack implementation
    - `Push()` and `pop()`
    - Other methods: `top()`, `count()`
- Applications:
    - Stack memory: function call
    - Check expressions: matching brackets
    - Depth-first graph search
- Question: How do you implement stack with linked list / (dynamic) arrays?

```
class Stack
{
public:
      bool push(const ItemType& item);
      ItemType pop();
      bool empty() const;
      int count() const;
private:
      // some features
};
```



Stacks

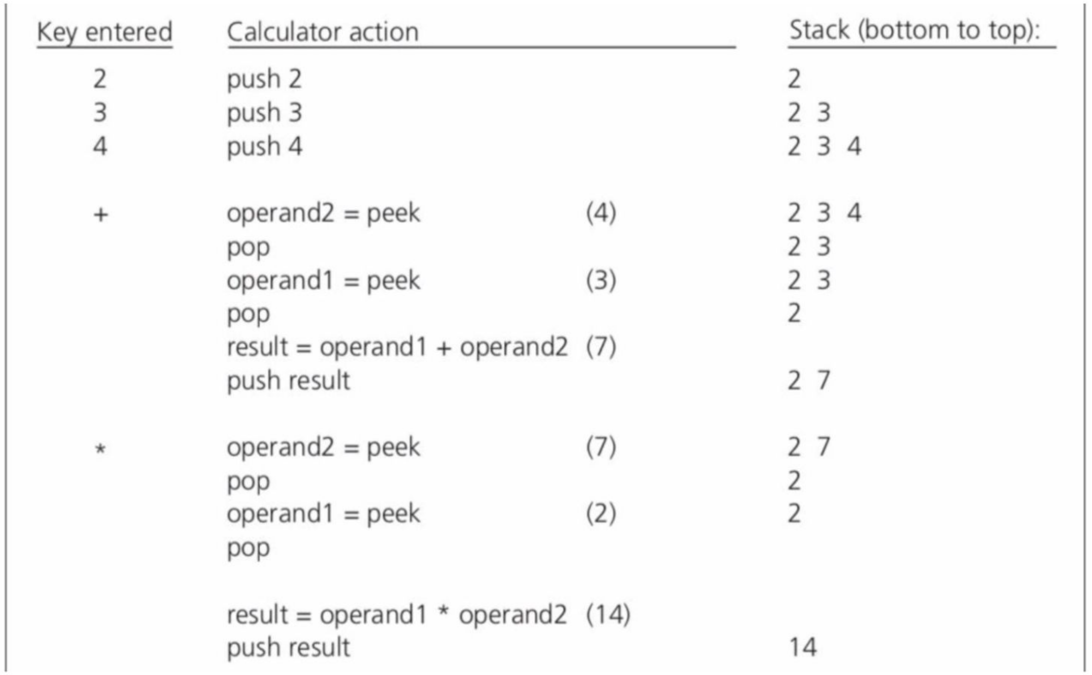


- **Infix Notation**
  - Operators are written in between their operands → X + Y
  - Ambiguous - needs extra rules built in about operator precedence and associativity and parentheses
- **Postfix Notation**
  - Operators are written after their operands → X Y +
  - Operators are evaluated left-to-right. They act on the two nearest values on the left.
- **Prefix Notation**
- **Tasks**
  - Evaluating Postfix Expressions
  - Converting Infix to Postfix Expressions

Stacks: Evaluate Postfix Expressions

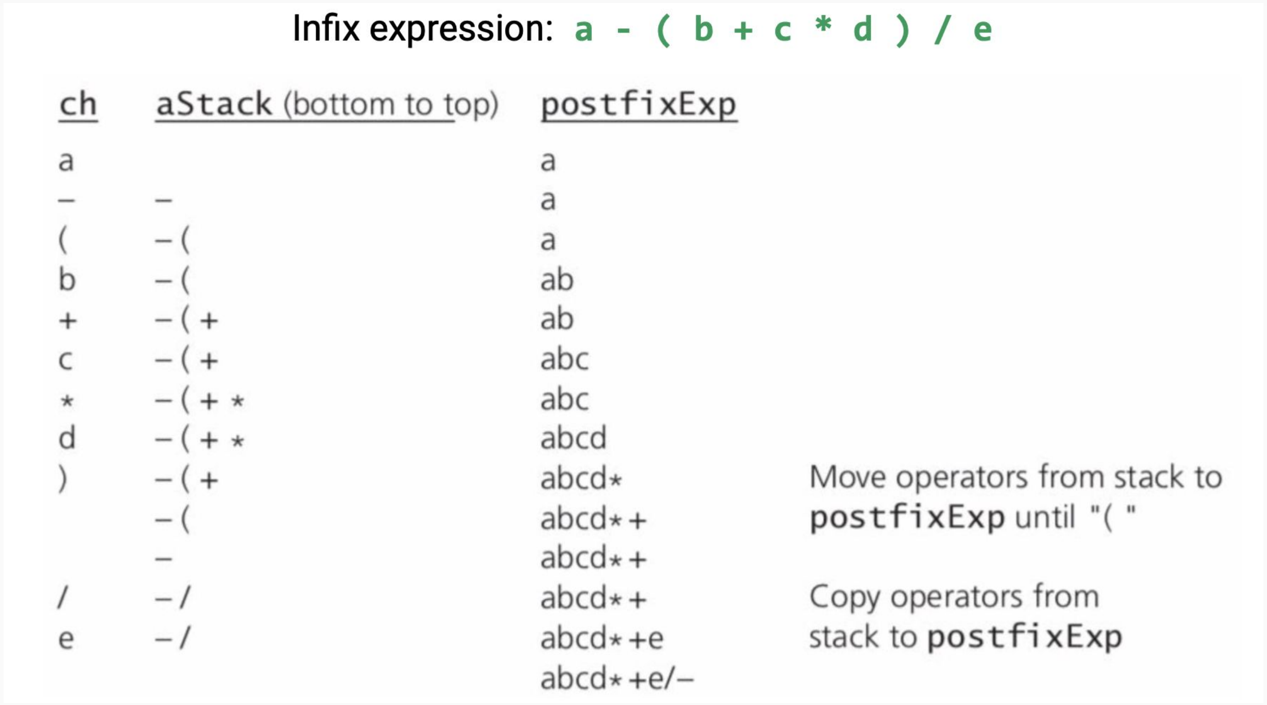
Postfix Expression
2 3 4 + *

Infix Expression
2 * (3 + 4)



Key entered	Calculator action		Stack (bottom to top):
2	push 2		2
3	push 3		2 3
4	push 4		2 3 4
+	operand2 = peek	(4)	2 3 4
	pop		2 3
	operand1 = peek	(3)	2 3
	pop		2
	result = operand1 + operand2	(7)	
	push result		2 7
*	operand2 = peek	(7)	2 7
	pop		2
	operand1 = peek	(2)	2
	pop		
	result = operand1 * operand2	(14)	
	push result		14

Stacks: Converting Infix to Postfix

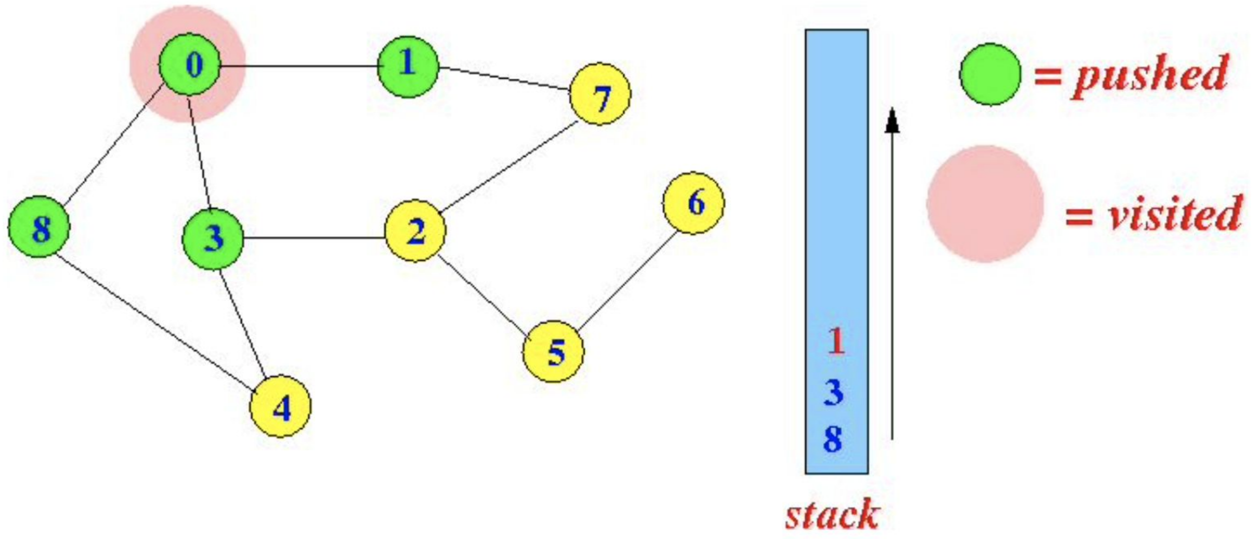


Infix expression: `a - ( b + c * d ) / e`

ch	aStack (bottom to top)	postfixExp	
a		a	
–	–	a	
(	– (	a	
b	– (	ab	
+	– (+	ab	
c	– (+	abc	
*	– (+ *	abc	
d	– (+ *	abcd	
)	– (+	abcd*	Move operators from stack to postfixExp until "("
	– (	abcd*+	
	–	abcd*+	
/	– /	abcd*+	Copy operators from stack to postfixExp
e	– /	abcd*+e	
		abcd*+e/–	

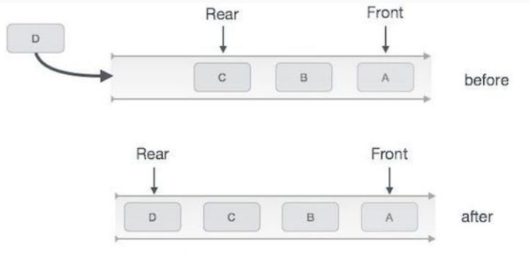
Stacks: Depth-first Search (DFS)

- Depth-first Search (DFS) on graph (will be later lectures or CS180)

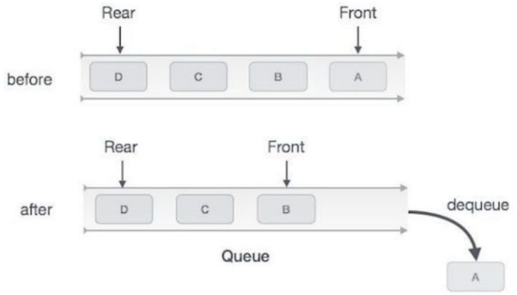


Queues

- FIFO: First In, First Out
- Basic methods:
  - `enqueue(), dequeue()`
  - `front(), back()`
  - `count()`
- Applications
  - Data streams
  - Process scheduling (DMV service request)
  - Breadth-first graph search
- How to implement queue with linked lists or dynamic arrays?

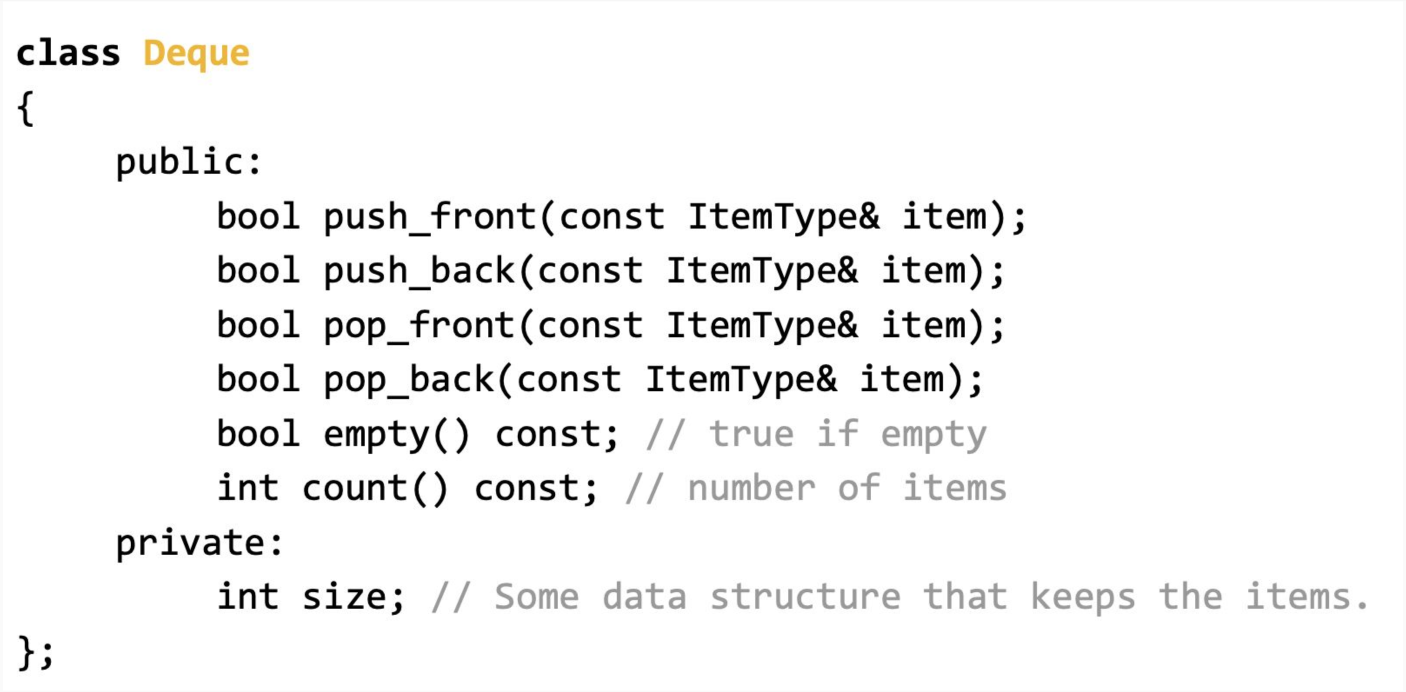


Queue Enqueue



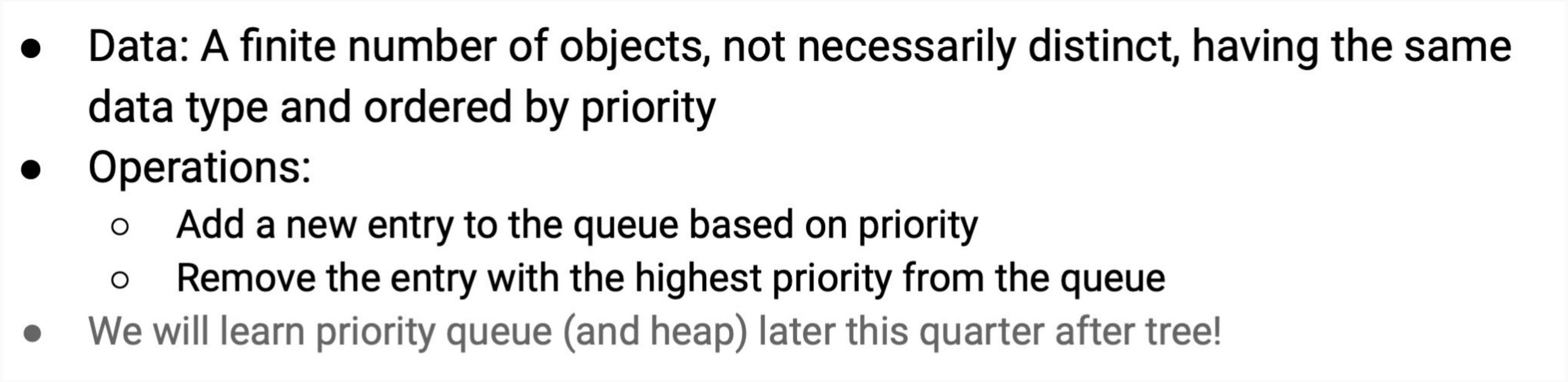
Queue Dequeue

Queues: Deque (double-ended queue)



```cpp
class Deque
{
    public:
        bool push_front(const ItemType& item);
        bool push_back(const ItemType& item);
        bool pop_front(const ItemType& item);
        bool pop_back(const ItemType& item);
        bool empty() const; // true if empty
        int count() const; // number of items
    private:
        int size; // Some data structure that keeps the items.
};
```

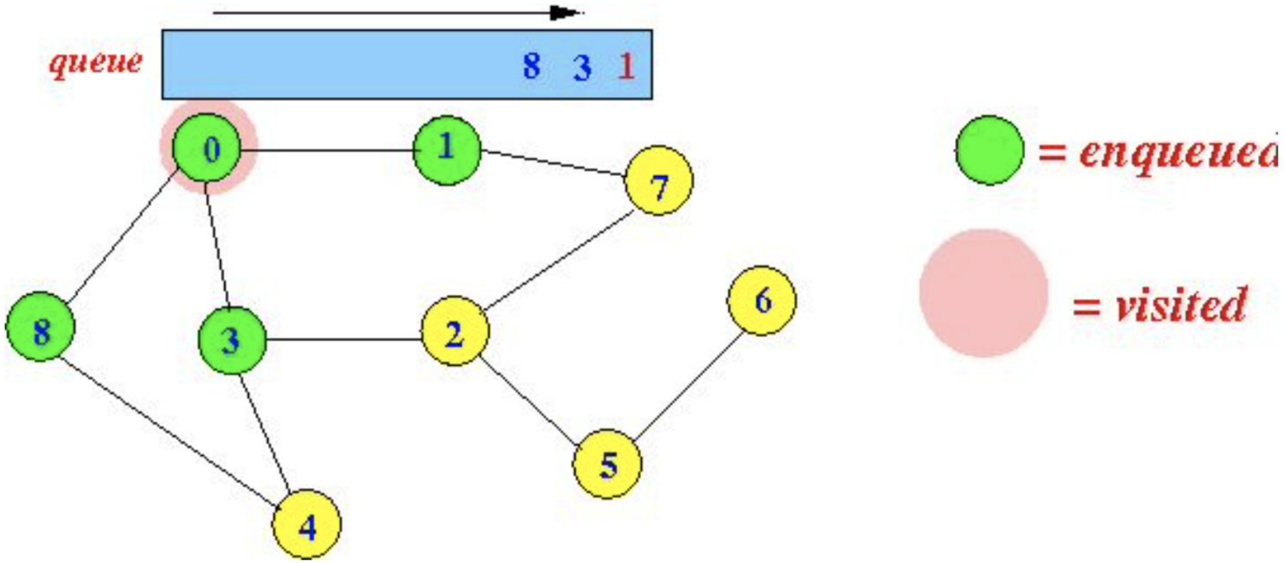
Queues: Priority Queue



- Data: A finite number of objects, not necessarily distinct, having the same data type and ordered by priority
- Operations:
  - Add a new entry to the queue based on priority
  - Remove the entry with the highest priority from the queue
- We will learn priority queue (and heap) later this quarter after tree!

Queues: Breadth-first Search (BFS)

- Breadth-first Search (BFS) on graph (will be later lectures or CS180)



Break: 5 mins

Worksheet

Codeshare

Room 1

Room 2

Room 3

Room 4

Worksheet Solution