Core Data Structures

Session 2

Overview

1. Evaluate RPN
2. Break out session
3. Recap of breakout sessions
4. Recap of the week

Evaluate Reverse Polish Notation

Evaluate Reverse Polish Notation

Evaluate the value of an arithmetic expression in Reverse Polish Notation.

What is “Reverse Polish Notation”?

What is “Reverse Polish Notation”?

How we write arithmetic equations: **infix notation**

What is “Reverse Polish Notation”?

How we write arithmetic equations: **infix notation**

3	+	4
operand	operator	operand

What is “Reverse Polish Notation”?

How we write arithmetic equations: **infix notation**

3	+	4
operand	operator	operand

...there are other ways to represent the same equation!

What is “Reverse Polish Notation”?

Operands go before the operator (also known as **postfix notation**)

What is “Reverse Polish Notation”?

Operands go before the operator (also known as **postfix notation**)

3	4	+
operand	operand	operator

More complex equations

RPN

12 2 /

Infix Notation

More complex equations

RPN

12 2 /

Infix Notation

12 / 2

More complex equations

RPN

12 2 /

3 12 2 / *

Infix Notation

12 / 2

More complex equations

RPN

12 2 /

3 12 2 / *

Infix Notation

12 / 2

12 / 2 * 3

More complex equations

RPN

12 2 /

3 12 2 / *

2 3 *

Infix Notation

12 / 2

12 / 2 * 3

More complex equations

RPN

12 2 /

3 12 2 / *

2 3 *

Infix Notation

12 / 2

12 / 2 * 3

2 * 3

More complex equations

RPN	Infix Notation
12 2 /	12 / 2
3 12 2 / *	12 / 2 * 3
2 3 *	2 * 3
12 2 3 * /	

More complex equations

RPN	Infix Notation
12 2 /	12 / 2
3 12 2 / *	12 / 2 * 3
2 3 *	2 * 3
12 2 3 * /	12 / (2 * 3)

What would this evaluate to?

10 2 3 + 1 * / 6 +

What would this evaluate to?

10 2 3 + 1 * / 6 +

What would this evaluate to?

10 2 3 + 1 * / 6 +

10 5 1 * / 6 +

What would this evaluate to?

10 2 3 + 1 * / 6 +

10 5 1 * / 6 +

What would this evaluate to?

10 2 3 + 1 * / 6 +

10 5 1 * / 6 +

10 5 / 6 +

What would this evaluate to?

10 2 3 + 1 * / 6 +

10 5 1 * / 6 +

10 5 / 6 +

What would this evaluate to?

10 2 3 + 1 * / 6 +

10 5 1 * / 6 +

10 5 / 6 +

2 6 +

Evaluate Reverse Polish Notation

Evaluate the value of an arithmetic expression in Reverse Polish Notation.

Valid operators are +, -, *, /. Each operand may be an number or another expression.

Evaluate Reverse Polish Notation

Evaluate the value of an arithmetic expression in Reverse Polish Notation.

Valid operators are +, -, *, /. Each operand may be an number or another expression.

Example #1

```
Input: ["2", "1", "+", "3", "*"]
Output: 9
Explanation: ((2 + 1) * 3) = 9
```

Evaluate Reverse Polish Notation

Evaluate the value of an arithmetic expression in Reverse Polish Notation.

Valid operators are +, -, *, /. Each operand may be an number or another expression.

Example #1

```
Input: ["2", "1", "+", "3", "*"]
Output: 9
Explanation: ((2 + 1) * 3) = 9
```

Example #2

```
Input: ["4", "15", "5", "/", "+"]
Output: 7
Explanation: (4 + (15 / 5)) = 7
```

Understand

- Make sure you understand RPN
  - Ask for a few walkthroughs
  - Come up with your own complex examples
  - Verify the order of operands with ‘/’ and ‘-’
- Will the numbers be integers or floats?
  - For sake of simplicity, let’s assume floats
- Do we need to handle the case in which the expression is not valid?
  - No

Match

- Need a data structure to store values in as you go through this expression
  - Queues or stacks?
- Need a data structure to map ‘*/+-’ operators to their corresponding functions
  - Hash Tables

Plan / Pseudocode

1. As we go through the list, we want to record the operands
    a. Need a data structure (stacks or queue?)
2. Every time you see an operand, you add it to the data structure
3. When you see an operator, you get the two most recent operands from the data structure and evaluate.
    a. Stack, not queue!
4. Add the result to the stack as a new operand
5. You keep repeating this as you go through the list until the end you hit the end

Plan: Stack based approach

10 2 3 + 1 * /

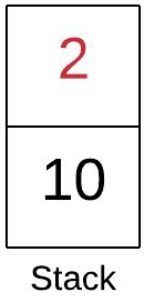
Plan: Stack based approach

10 2 3 + 1 * /

Stack

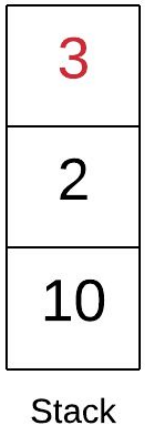
Plan: Stack based approach

10 2 3 + 1 * /



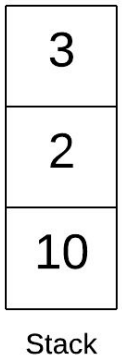
Plan: Stack based approach

10 2 3 + 1 * /



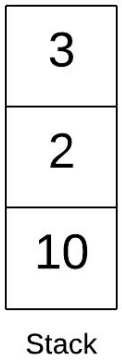
Plan: Stack based approach

10 2 3 + 1 * /

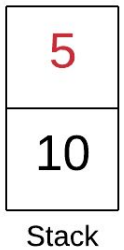


Plan: Stack based approach

10 2 3 + 1 * /



10 2 3 + 1 * /



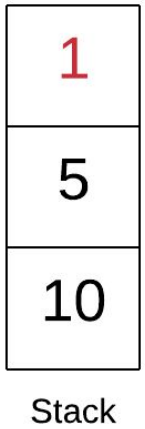
Remove 3 from stack

Remove 2 from stack

Add 2 + 3 to stack

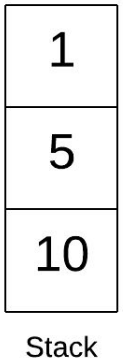
Plan: Stack based approach

10 2 3 + 1 * /



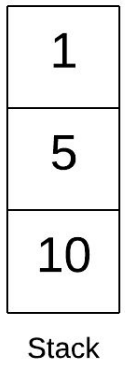
Plan: Stack based approach

10 2 3 + 1 * /

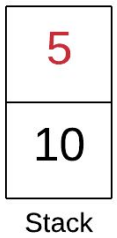


Plan: Stack based approach

10 2 3 + 1 * /



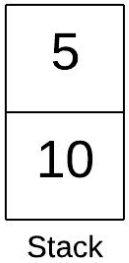
10 2 3 + 1 * /



Remove 5 from stack
Remove 1 from stack
Add 1 * 5 to stack

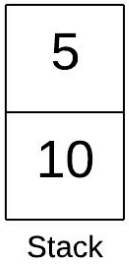
Plan: Stack based approach

10 2 3 + 1 * /



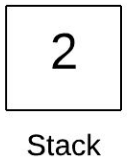
Plan: Stack based approach

10 2 3 + 1 * /



10 2 3 + 1 * /

Remove 5 from stack
Remove 10 from stack
Add 10/ 5 to stack



Pseudocode

Create a mapping of operator to functions

Pseudocode

```
Create a mapping of operator to functions

For each item in the list:
```

Pseudocode

```
Create a mapping of operator to functions
For each item in the list:
    If the item is an operator:
        Pop two off the stack, evaluate, add to stack
```

Plan / Pseudocode

```
Create a mapping of operator to functions
For each item in the list:
    If the item is an operator:
        Pop two off the stack, evaluate, add to stack
    Else:
        Add to stack
```

Break out session

- 60 minutes, 2 questions
- Share a repl.it or any similar site
- Use UMPIRE for each problem
- Rotate the leader for each step
- Slack me or the **#se103-jun20** chat for questions

GOAL: Have a working solution for Brick Wall and try to at least have the pseudocode for Celebrity

Recap

- Start even without TA
- Run your code
- Test your code before running it through leet code
  - **DON’T** rely on leetcode
- Plan stage should be pseudocode
- Slack @Elena for any issues

Interview Recap: Brick Wall

- Really helps to visualize this problem
- A few different edge cases to test
  - All bricks must be crossed
  - No bricks need to be crossed
  - No bricks

Interview Recap: Celebrity

- Tricky problem to understand

Interview Recap: Celebrity

- Tricky problem to understand
- Cases to test
  - 0 people, 1 person, no celebrity, celebrity

Implement “Evaluate RPN”

Time to code!

Evaluate

Time complexity: O(n)

- N = length of input

Evaluate

Time complexity: O(n)

- N = length of input

Space complexity: O(n)

- N = length of input

Week 2 Recap

- Heaps, Stacks, Queues, Hash Tables
- Problems
  - Warm up: Top K frequent words
  - Session #1: Design a min stack
  - Session #2: Evaluate RPN
  - Post session: Implement queue using stacks
  - Post session: Find median from a data stream

Reminders

- Solutions for warm up problems & post assessment questions are up
- Solutions for HackerRank Week #1 is up
- Recommendation
  - Try a hard level difficulty question (Minimum Number of Refueling Stops)
  - Review walkthrough video
  - Review the guides, all the problems
- HackerRank is due on Monday
- Review week next week!