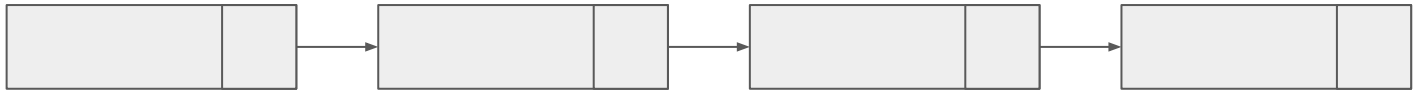
Cooperative Problem Solving & Competitive Programming

Linked Data Structures

List

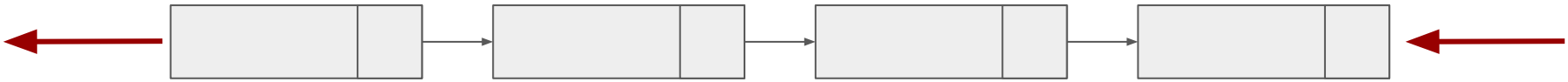
- Elements linked to each other via pointers



- Fast insertion, very slow random access.
- Not many use cases.
  - i.e. there is always a better choice.

Queue

- First In - First Out



- Only allows insertion from one end and deletion from the other. No random access.

Stack

- Last In - First Out

- Only allows insertion and deletion from one end. No random access.

Bracket Matching

- **Problem**: given a series of braces, find if the sequence is correct.

Bracket Matching

- **Problem**: given a series of braces, find if the sequence is correct.

- **Solution:**
  - if bracket is open: add to stack.
  - if bracket is closed: check the top of the stack for corresponding bracket.
  - if at the end the stack is empty, correct.

Bracket Matching

- **Example:** ( ) { [ ] }

( ) { [ ] }	(
( ) { [ ] }	
( ) { [ ] }	{
( ) { [ ] }	{[
( ) { [ ] }	{
( ) { [ ] }	

- **Example:** ( ) { ( ] }

( ) { ( ] }	(
( ) { ( ] }	
( ) { ( ] }	{
( ) { ( ] }	{(
( ) { ( ] }	ERROR

Postfix Calculator

Different ways to write mathematical expression:

- **Infix**: operator is between operands

  4 * (1 + 2 * (9 / 3) - 5)

- **Prefix**: operator is before operands

  * 4 - + 1 * 2 / 9 3 5

- **Postfix**: operator is after operands

  4 1 2 9 3 / * + 5 - *

Postfix Calculator

- Postfix notation is more computationally efficient than Infix notation.
  - e.g. No parenthesis.
- Given an expression in postfix notation, write an algorithm that computes its value.

4 1 2 9 3 / * + 5 - *

Postfix Calculator – Solution

Spoiler prevention slide

Postfix Calculator – Solution

- Keep a stack of operands.
- When an operator is read, pop the 2 top items and push the result.
- The result is the only element left in the stack.

4 1 2 9 3 / * + 5 - *	**4 1 2 9 3**
4 1 2 9 3 **/** * + 5 - *	4 1 2 **3**
4 1 2 9 3 / ***** + 5 - *	4 1 **6**
4 1 2 9 3 / * **+** 5 - *	4 **7**
4 1 2 9 3 / * + **5** - *	4 7 **5**
4 1 2 9 3 / * + 5 **-** *	4 **2**
4 1 2 9 3 / * + 5 - *****	**8**

Infix to Postfix Conversion

- Many compilers convert Infix expressions into Postfix conversions.
- This can be done in O(n) time, with n = the number of operands/operators.
- You should also consider operator precedence!

$$1 + 2 * 3 \rightarrow 1\ 2\ 3\ *\ +$$

$$1 * 2 + 3 \rightarrow 1\ 2\ *\ 3\ +$$

- Start without brackets, then find a solution that manages them.