CS240

Nathan Sprague

September 26, 2014

Stack ADT

Stack ADT

- `s.push(e)` - Adds e to the top of the stack.
- `s.pop()` - Removes and returns the top item.
- `s.is_empty()` - True if `s` does not contain any items.
- `len(s)` - Returns number of items in the stack.
- `s.top()` - Returns the top item without removing it.

Exercise

```python
s = Stack()
s.push("A")
s.push("B")
s.push("C")
s.pop()
s.push("D")

while not s.is_empty():
    print(s.pop())
```

1: A B C D
2: D B A
3: A B D

Exercise

```python
s1 = Stack()
s2 = Stack()
s1.push("A")
s1.push("B")
s1.push("C")

while not s1.is_empty():
      s2.push(s1.pop())

while not s2.is_empty():
      print(s2.pop())
```

1: A B C
2: C B A
3: CRASH!

Implementation...

Dynamic Array Stack Performance

`s.is_empty()`	O(1)
len(s)	O(1)
`s.pop()`	O(1)*
`s.peek()`	O(1)
`s.push(item)`	O(1)*

* Amortized

Uses for Stacks

- Reversing elements in a sequence

Uses for Stacks

- Reversing elements in a sequence
- Matching Parentheses

Uses for Stacks

- Reversing elements in a sequence
- Matching Parentheses
- Evaluating postfix expressions...

Infix and Postfix (RPN)

- Infix version:
  - $3 + 5 * 2 - 6$
  - $= ((3 + (5 * 2)) - 6)$
- Postfix version:
  - 3 5 2 * $+$ 6 -

Converting (Parenthesized) Infix to Postfix

- We can use the same idea as the parenthesis matching algorithm from the book.

```
procedure InfixToPostfix($t$: a sequence of tokens)
    for Each token in $t$
        if The token is an operand then
            Copy it to the result
        else if The token is an operator then
            Push it on the stack
        else if The token is “)” then
            Pop from the stack, and append to the result
        else // The token is “(”
            Ignore it
    return The resulting expression
```

Infix, Postfix and Stacks

Evaluating Postfix using a stack:

```
procedure EvaluatePostfix($t$: a sequence of tokens)
    for Each token in $t$
        if The current token is an operand then
            Push the token on the stack
        else    // Token must be an operator
            Pop the top two items
            Perform the operation
            Push the result on the stack
    return The top of the stack
```

Exercise

- Consider this infix expression:
  $2 * 4/8 + 7 * 3$
- Convert to postfix
- Hand evaluate using a stack.