# CSCI 204: Data Structures & Algorithms

Stacks

## Stacks

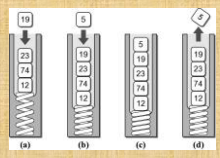
Revised based on textbook author's notes.

## Stacks

- A restricted access container that stores a linear collection.
  - Very common for solving problems in computer science.
  - Provides a **last-in first-out** (LIFO) protocol.

## The Stack

- New items are added and existing items are removed from the **top** of the stack.




## Why stacks are useful?

- Many computer science problems (and real life problems) are solved using stacks.
- We'll touch a few here.
  - Determine if a string is a palindrome
  - Find paths among a set of cities
  - Evaluate math expressions

## The palindrome problem

- A palindrome is a string that reads the same in forward and backward
- Here is a list of palindrome http://www.palindromelist.net

A Toyota's a Toyota.
Civic
Go dog.
Tell a ballet.

- We saw the solution recursion before

```
def is_palindrome(s):
    """ Check if s is a palindrome, assuming s contains only lower-case
    letters that have been preprocessed (see function preprocess()"""
    if len(s) <= 1:
        return True
    else:
        count = len(s)
        return (s[0] == s[count-1]) and is_palindrome(s[1:(count-1)])
```

## We can also solve the problem using stacks

- Given a string s, put the sting into a stack, then take the one on stack out in order (LIFO). If the two are the same, it is a palindrome!

| Original string | In-stack | Output | |
|---|---|---|---|
| civic | civic | civic | Yes |
| godog | godog | godog | Yes |
| hello | olleh | olleh | No |

## The Stack ADT

- A *stack* stores a linear collection of items with access limited to a last-in first-out order.
- Adding and removing items is restricted to the top of the stack.

- Stack()
- is_empty()
- len()
- pop()
- peek()
- push( item )

8

## Stack Example

reverse.py

```
# Extracts a collection of integer values from the user
# and prints them in reverse order.
from list_stack import Stack
#from pyliststack import Stack

PROMPT = "Enter an int value (<0 to end): "
my_stack = Stack()

# Extract the values and push them onto a stack.
value = int(input( PROMPT ))
while value >= 0 :
  my_stack.push( value )
  value = int(input( PROMPT ))

# Pop the values from the stack and print each.
while not my_stack.is_empty() :
  value = my_stack.pop()
  print( value )
```

## Stack Implementation

- Several common ways to implement a stack:
  - Python list
    - easiest to implement
  - Linked list
    - better choice when a large number of push and pop operations are performed.

## Activity

- Implement the Stack ADT in two different ways
  - Using a singly linked list
  - Using a Python list