Data Structures

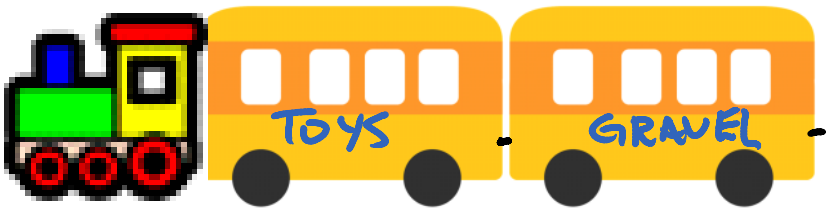
Wednesday, March 8, 2017
11:27 PM

LC101 JS: Class 3.4

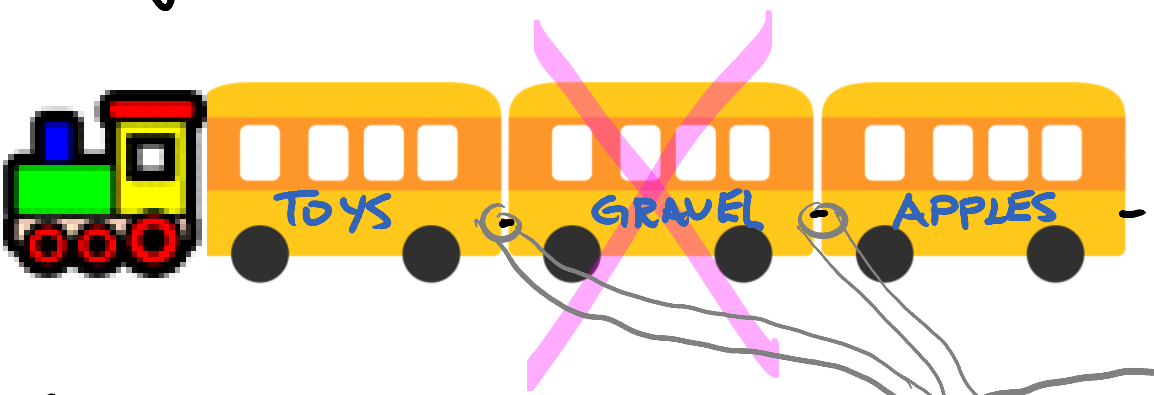
For more information on the complexity analysis and implementation details, check out:
http://blog.benoitvallon.com/data-structures-in-javascript/data-structures-in-javascript/

Linked Lists

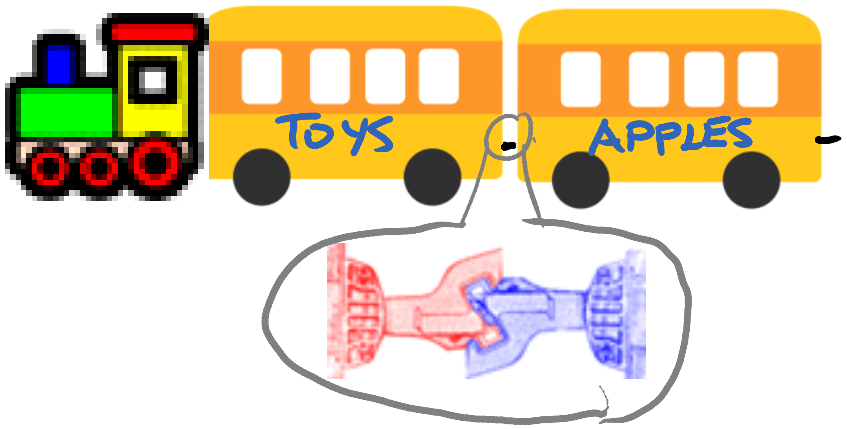
Imagine we have a line of freight cars storing stuff.



Let's say we want to add something,
Simply hook it up to the back.



If we want to remove something,
just detach the connections,
remove the cart, and then reattach the two segments!

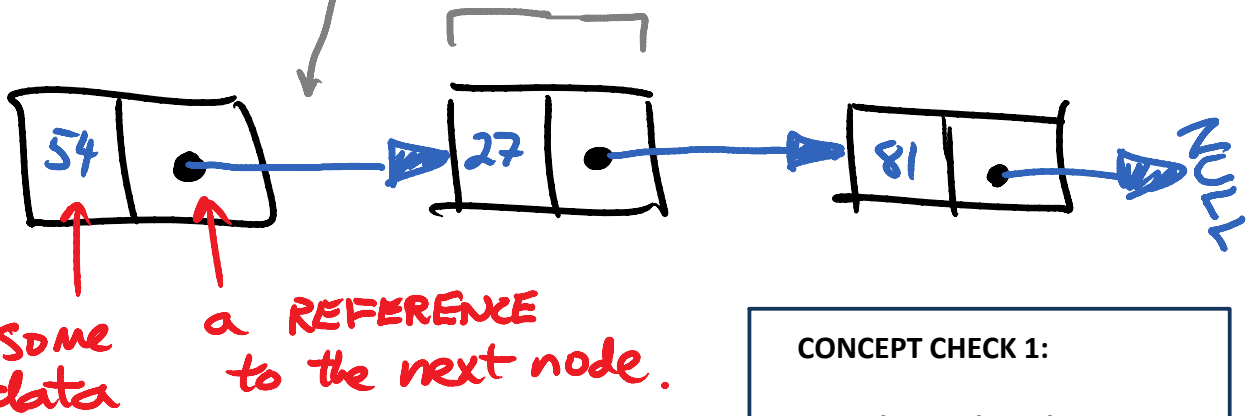


That's the idea. Now for terminology.

TRAIN = LINKED LIST

CAR = NODE

CONNECTOR = LINK



CONCEPT CHECK 1:

Describe an algorithm to reverse a singly linked list.

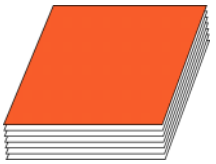
Advantages (+):

- Insertion/deletion node operations are easily implemented and can be efficiently performed in the middle of the list.
- Unlike an array, there's no need for expensive resizing operations or wasteful memory allocation.
- There is no need to define an initial size for a linked list.

Disadvantages (-):

- They use more memory than arrays because of the storage used by their pointers.
- Nodes in a linked list must be read in order from the beginning (so, bad for random access!).
- Nodes aren't guaranteed to be stored contiguously, thus increasing the time required to access individual elements.

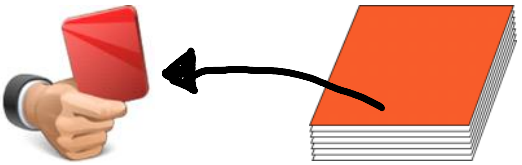
Stacks

Here's a stack of cards... 

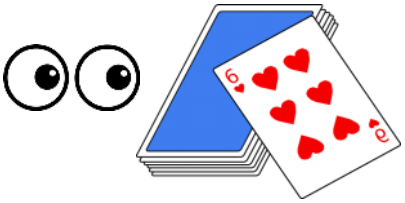
We can perform several "operations":

(1) "PUSH": place a card on top.

(2) "POP": take a card off the top.



(3) "PEEK": look at top card, but DO NOT remove.

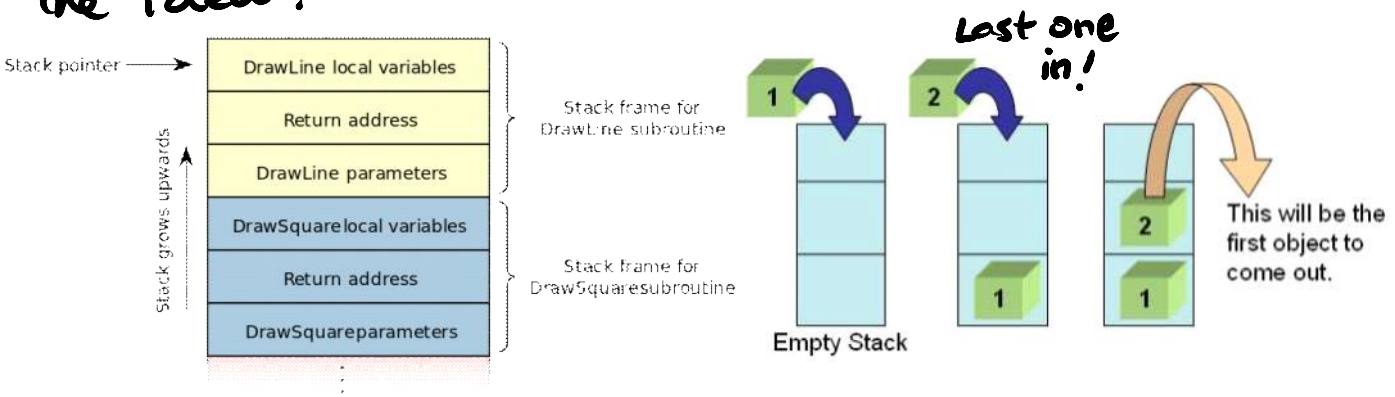


Fun Fact: When you call a function, it goes on the "call stack". Let's pretend you're calling a function recursively & forget to write a base case...

You'll likely get a stackoverflow error.

This is a Last-in-first-out (LIFO) structure!

Here's 2 additional diagrams to help illustrate the idea:



CONCEPT CHECK 2:

Review the following JAVA ~~script~~ snippet. What is printed to the console? Draw a diagram of the stack after the following code executes.

```
1.  import java.util.*;
2.
3.  class StackDemo {
4.    public static void main(String[]args) {
5.      Stack<String> stack = new Stack<String>();
6.      stack.push("A");
7.      stack.push("B");
8.      stack.push("C");
9.      stack.push("D");
10.     System.out.println(stack.peek());
11.     stack.pop();
12.     stack.pop();
13.   }
14. }
```

Queues

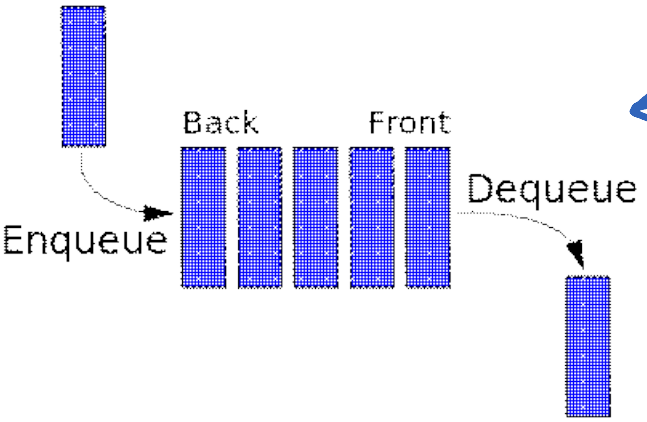
This is a queue:

or

(a "line" if you're not from the UK).

Enqueue - to get in line

Dequeue - the equivalent of someone yelling "Next!" at the DMV.



← Here's the structure.

NOTE: it's first in, first out! (FIFO)

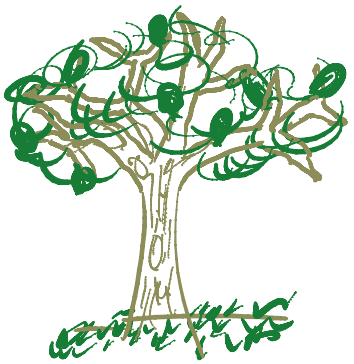
CONCEPT CHECK 3:

Assume that a Queue class has been written in JS. What will be printed to the console? How would you implement a queue if you could only use two stacks?

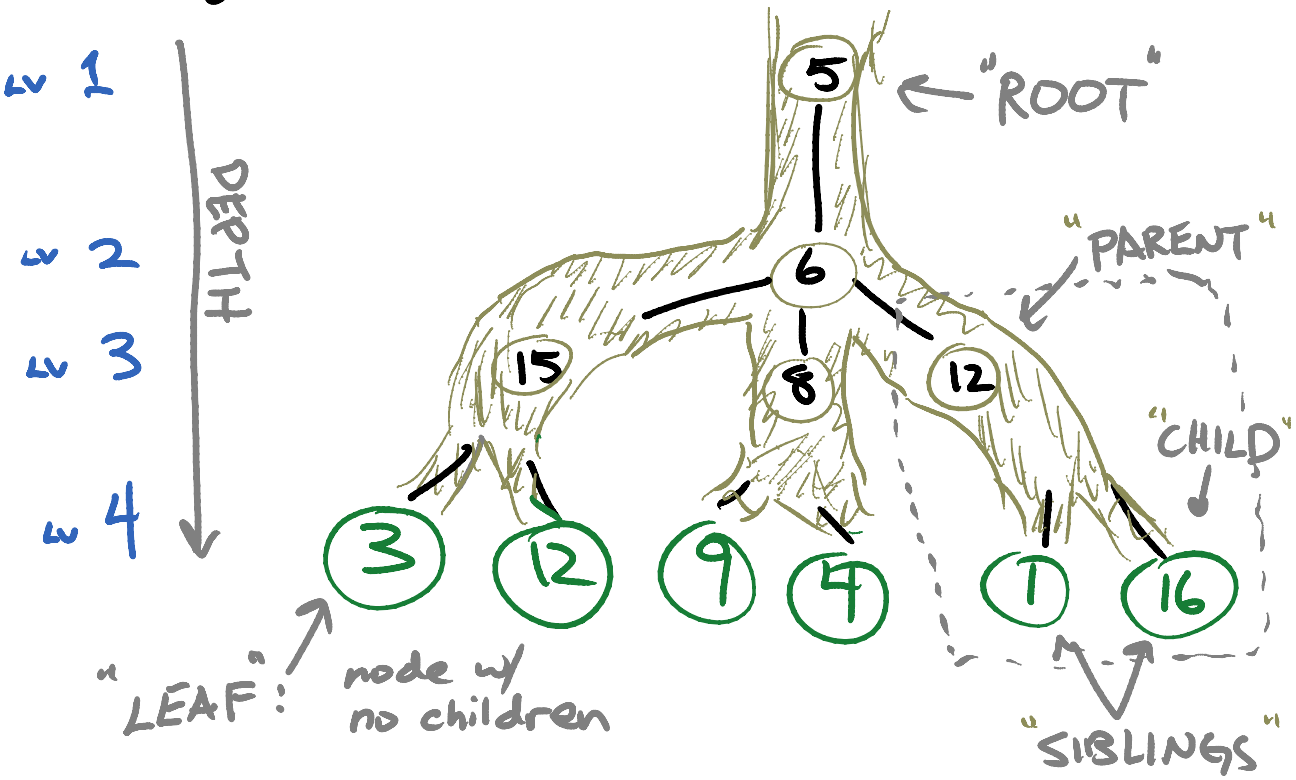
```
1. var queue = new Queue();
2. queue.enqueue("⏸");
3. queue.enqueue("⏵");
4. queue.enqueue("⏺");
5.
6. queue.dequeue();
7. queue.dequeue();
8. console.log(queue.peek());
```

Trees

Think of a tree like this ...



but now upside down —
& with numbers.



- No cycles allowed!

⇒ • Each node only has 1 parent!

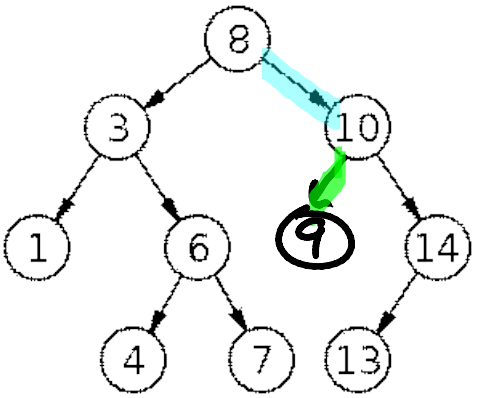
A very useful type of tree is a BINARY SEARCH TREE.

INSERTION

Let's add 9.

9 > 8 RIGHT

9 < 10 LEFT



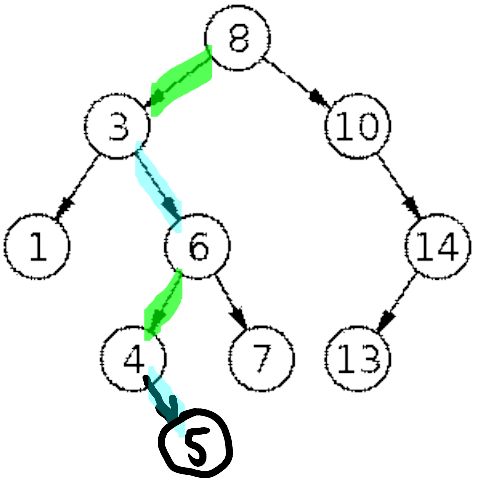
Let's add 5

5 < 8 LEFT

5 > 3 RIGHT

5 < 6 LEFT

5 > 4 RIGHT



DELETION

This is slightly more complicated. For a great explanation, go to:
http://www.algolist.net/Data_structures/Binary_search_tree/Removal

CONCEPT CHECK 4

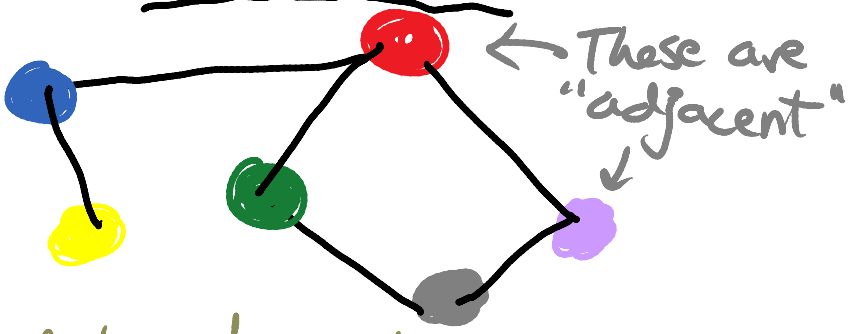
Insert 12 into the tree above and draw the resulting tree.
What is the printout of the following code? (Warning: This is a bit tricky!)

```
1. def traverse_binary_tree(node, callback):
2.     if node is None:
3.         return
4.     traverse_binary_tree(node.leftChild, callback)
5.     callback(node.value)
6.     traverse_binary_tree(node.rightChild, callback)
7.
8. traverse_binary_tree(root, console.log)
```

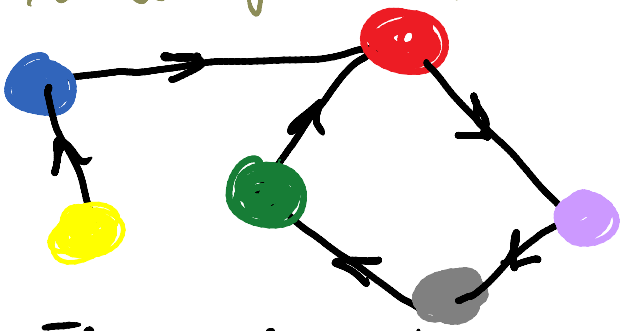
Graphs

Think of it as a network.

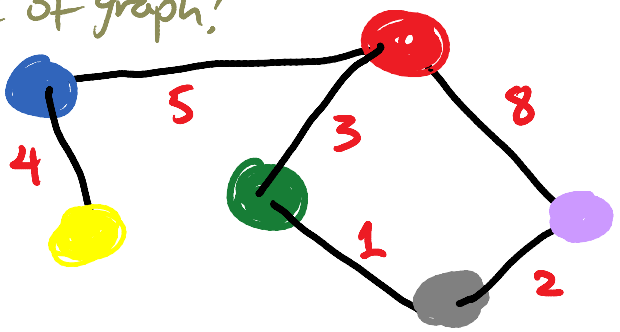
A graph has edges & nodes/vertices



NOTE:
A tree is just a special type of graph!



If we add direction, it's a DIRECTED GRAPH.

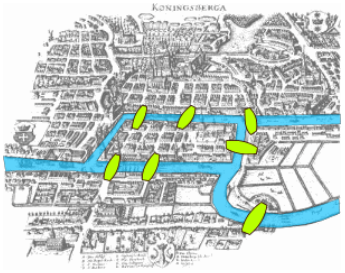


If we associate "weights" to edges, it's a WEIGHTED GRAPH.

TRAVERSAL 2 WAYS
- Depth-First – uses a stack
- Breadth-First – uses a queue

CONCEPT CHECK 5: This is better explained using a visualization: https://visualgo.net/dfsbfs

Start at the RED node above and outline the process for DFS and BFS.
The Seven Bridges of Königsberg: simplify the map below by representing it as a graph.



Can you take a walk through town, visiting each part of the town, and cross each bridge only once?