Quicksort Quick sort is another divide and conquer algorithm. It has an average complexity of $O(N \log(N))$, and best?? case complexity is $O(N^2)$.

- An array of size 0 or 1 is already sorted.
- Pick an element from the array called the pivot.
- Arrange the array so that the elements smaller than the pivot are below it, and the ones larger are above it.

We won't worry about the implementation of quick sort in this unit. It will be part of COMP225.

Stable Sort What happens when you've got two equal values? Say you're sorting people by first name and you've got two people with the same name. In a stable sort, any two equal items stay in the same order they were to start with. An unstable sort might re-order them.

Mixed Sort Something you can do is mix sorting types based on the array length.

Abstract Data Types

- ADT is a way of thinking about a data structure. A list is an example of this.
- They can be implemented in different ways.
- They might have different performance properties. ArrayList is faster at selecting elements than LinkedList.

Another aspect of ADT provides an alternate interface to the same kind of storage. We normally treat List as a sequence of items, but there are cases where it's useful to view it in a different way.

- Two ADTs that are really lists with more restricted interfaces.
- Stack: add things to the front, remove them from the front.
- Queue: add things to the back, remove from the front.

Why would you want to have a type that is more restricted? It stops you from doing things you don't want to do, limiting errors from occurring. The main advantage is that it changes how you think about the data. Because you're restricted, it makes you think of the data as a stack.

Stack

- A stack is a data structure where elements are added and removed from the top
- Called a LIFO data structure.
- Basic operations:

- push adds a new item to the top of the stack
- pop removes the top item and returns it
- peek returns the top item without removing it.
- You can also check if the stack is empty.

Stack Examples Stacks are at the core of computing. Many algorithms make use of stacks to store data as they run.

Undo The undo in your editor uses a stack. Each operation that changes the text is pushed on the stack. To undo we pop the operation from the stack and reverse it. The stack is perfect for this because the last thing added is always the first thing you want to undo.

Balanced Parentheses

- To check whether parentheses are balanced in an expression or program.
- Reading through the expression left-to-right
- When I see an open parenthesis '(' I push it to the stack.
- When I see a close parenthesis I pop the stack.
- At the end, the stack should be empty.
- If there's something there, or you try to pop an empty stack, there's something wrong.

Reverse Polish Notation

- Arithmetic expressions where the operator comes after the operands.
- 3 4 + adds two numbers.
- 3 4 + 6 * then multiplies by 6.
- 3 4 6 + * adds 4 and 6, then multiplies by 3.

This is stack based maths. See the related class for a rough implementation of this.

Stack Overflow Java keeps a stack to keep track of where it is inside recursive methods. When Java calls a method, the current state is pushed on a stack. If you have an unbounded recursion method, Java continues to add to the stack until it runs out of memory space, causing a a stack overflow.

Forth Forth is a stack based programming language. It uses the stack to pass arguments and generally doesn't use variables. See example.forth for an example of what it looks like.

Queues

- A stack is a LIFO structure.
- A queue is a FIFO structure
- Add stuff to the back (tail), remove stuff from the front (head).
- enqueue - add something to the back
- peek - return front
- dequeue - return and remove front

Queues are widely used in computing. Printers have them, video rendering, ticketing systems, simulations of real life queues.

Queue Implementation In Java, Queues are not quite like Stacks or the other classes we've seen. Queue is an Interface. For now, think of this as a way of using another type. To make a queue: $Queue < Integer > myq = newLinkedList < Integer > ();$

The built in queue method names are different. They include add, remove, and peek.

Car Wash Simulator

- Let's simulate a car wash
- On average, one car arrives every 6 minutes
- Full wash takes 4 minutes
- When a car arrives and the car wash is free it gets washed immediately.
- Otherwise it waits in the queue.