Data Structures and Algorithms
Background
Queues and Stacks

Goodrich & Tamassia, Chapter 5

As pre-requisite for Graph Search, we revise two basic data structures

- Queues (FIFO)
- Stacks (LIFO)

Queues

The goal of a **queue** data structure, is to store items in such a way that the least recent (oldest) item is found first.

It only provides access only to the front of the queue, retrieving the oldes element, while always adding to the rear of the queue.

Thus, items are processed in **first-in, first-out (FIFO)** order.

Examples: supermarket queue

Applications: reversing an array

Queues

A typical API for a queue data structure is:

enqueue(e)	insert element e at the *rear* of the queue
dequeue()	remove and return from the queue the element at the *front*
size()	return the number of elements in the queue
isEmpty()	return a boolean indicating if the queue is empty
front()	return the front element in the queue, without removing it

Stacks

The goal of a **stack** data structure, is to store items in such a way that the most recent item is found first.

It only provides access to the top element in the stack (the most recent element).

Thus, items are processed in **last-in, first-out (LIFO)** order.

Examples: matching parentheses

Applications: reversing an array

Stack

A typically API for a stack data structure is

push(e)	insert element e, to the *top* of the stack
pop()	remove from the stack and return the *top* element on the stack
size()	return the number of elements on the stack
isEmpty()	return a boolean indicating if the stack is empty
top()	return the top element on the stack, without removing it

Trying to remove an element from an empty stack should throw an exception.

Iteration vs Recursion

Repetition of sequences of operations can be achieved in two ways

- by *iteration*, using a loop, or
- by *recursion*, using function/method calls.

Example of Iteration

Problem: Compute the sum over all elements of an array A of length n.

IterativeSum(A,n)
Input: An integer array A and an integer $n \geq 1$,
such that A has at least n elements
Output: The sum of the first n integers in A

```
for (i=0, s=0; i<n; i++)
  s = s + A[i]
return s
```

Example of Iteration vs Recursion

Problem: Compute the sum over all elements of an array A of length n.

LinearSum(A,n)
Input: An integer array A and an integer $n \geq 1$,
such that A has at least n elements
Output: The sum of the first n integers in A

```
if n=1 then
  return A[0]
else
  return LinearSum(A,n-1) + A[n-1]
```

Recursive Algorithms over Recursive Data Structures

Some data structures are defined in a recursive fashion: e.g. a *List* is either a null pointer, or a value followed by another *List*.

For such data structures, it is natural to use recursive algorithms: e.g. the length of a List, is either 0 (if the List is null), or 1 plus the length of the remaining List.

Exercise: Solve the Huffman tree decryption exercise both using iteration and recursion.