# Chapter 1: Stack & Quene

## Stack

1. First in last out(FILO)
2. Basic operations
   * 1. Pop():Take the data out from the top of the stack
     2. Push():Put the data to the top of the stack
     3. IsEmpty():Check the stack is empty or not

DIsFull():Check the stack is full or not(especially if the stack is made by array)

1. application(Pre,In,Postfix)
2. Infix

Format:operand1 operator operand2(ex:a+b)

Disadvantage: Calculating with infix notation in the compiler is quite challenging. This is because one has to account for(考慮) the priority and associativity(關聯性) of operators, which may necessitate(需要) multiple scans to compute the final result.

1. Postfix

Format:operand1 operand2 operator(ab+)

Advantage: The compiler only needs to scan from left to right once to produce the result, eliminating the need to worry about operator priority and associativity.

1. Prefix

Format:operator operand1 operand2(+ab)

Advantage: Same as postfix.

1. Infix to postfix conversion algorithm
2. Scan the expression from left to right.
3. If an operand is encountered, print it out.
4. If an operator is encountered, consider the following situation

If the stack is empty.Push()

If the stack contains other operators, compare their priorities with the current operator. If the priority of the operator at the top of the stack is lower, push() the current operator. Otherwise, pop() the stack until the priority at the top of the stack is lower than the current operator's. Do this until the stack is empty or the priority conditions are met.

If a “)” is encountered, pop() the current data in stack, until encounterint ”(“.

## Queue

1. First in first out(FIFO)
2. Basic operations
   1. Dequeue():Take the data out from the front of the queue
   2. Enqueue():Put the data to the rear of the queue
   3. IsEmpty():Check the queue is empty or not
   4. IsFull():Check the queue is full or not(especially if the queue is made by array)

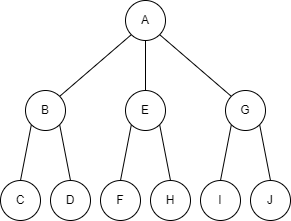
# Chapter 2: Tree & Binary Tree

1. Tree
   * + - 1. Definition: A tree is a of multiple nodes, and it cannot be empty.

At least has one root

The remaining nodes can be divided into mutually exclusive sets, denoted as subtrees T1 to Tm, which are referred to as the subtrees of the root.

* + - * 1. terminology:



Level: root is level 0

Degree of nodes:The number of children that a node has

Degree of A = 3

Degree of B = 2

Leaf:The node that degree is equal to zero

None-leaf:The node that degree is not equal to zero

Child

Parent

Sibling:Node sharing the same parent are considered siblings.

Ancestors:All the direct or indirect predecessors of a node.

Descendants:All the direct or indirect successors of a node.

Level of nodes

Degree of a Tree

Height of a Node:The longest path to leaf node

1. Binary Tree
   * + - 1. Definition:

It can be empty(node = 0), if it’s not empty it must have root and subtrees on the left and right

The subtrees on the left and right must be Binary trees.

The left and right subtrees have directional distinction.

* + - * 1. Basic theorem and proof

In Binary tree, the i-th level will have at most 2i-1 nodes.

proof by mathematical induction:

1. For level = 1, 21-1 = 1
2. If level = n - 1 holds, show that level = n also holds
3. For level = n, nodes of

For a binary tree with height of k. It will have maximum 2k-1 nodes and a minimum of k nodes.

In a non-empty binary tree, if the number of nodes with a degree of 0 (leaf nodes) is n0, and the number of nodes with a degree of 2 is n2, then n0 = n2 + 1

Proof:

n is the number of nodes n = n0 + n1 + n2

n1 is the number of nodes with degree equal to 1

B is the number of branches B = 0n0 + 1n1 + 2n2

* + - * 1. Type

Skewed Binary Tree

Full Binary Tree

Complete Binary Tree

Strict Binary Tree

1. Binary Tree Traversal
2. Binary Search Tree
3. Heap
4. Thread Binary Tree
5. Tree and Binary Tree Conversion
6. Advanced Tree