Ch5 Tree與Binary Tree

Tree是由>0個Nodes構成之有限集合，不可以為空

Root’s level = 1

(有些版本、題目設為0開始)

Forest

>=0顆互斥樹，所構成之集合，可以為空

Tree的表示方法

法一:利用Link List直接表示Tree

如果Tree’s degree = k，故一個Node準備K個Link欄位

缺點:浪費Link Space，只有(n-1)條Link有用(n個節點)

法二:將Tree化成Binary Tree，再用B.T表示方法儲存

法三:child-sibling方法

Node Structure為Data child sibling

Child: Left-most child only

Sibling: Next-right sibling

法四:括號法

利用父(子1子2…子m)表現父與子點之組成關係，且可以巢狀表示

Binary Tree是由>=0個Nodes所構成之有限集合，可以為空

滿足:

1. 有Root及左右子樹
2. 左右子樹也是Binary Tree

Binary Tree三個基本定理(假設 root之Level = 1)

1. B.T中第i Level之最多Node數:
2. 高度k之Binary Tree，最多Node數:

最少Node數:k(一層一個點)

1. 非空的B.T

Leaf個數:N0個

Degree = 2 ，Node數:N2個

N0 = N2 + 1

Binary Tree的種類

1. 斜曲的B.T => 左斜、右斜
2. Full B.T => 高度=k , 最多Node數 =
3. Complete B.T => 高度=k, n個Nodes , 滿足 < n < ，由上而下，由左而右的Node增長順序

Complete B.T之定理

某Node之編號是i

左子點編號:2\*i

右子點編號:2\*i + 1

父點編號:i/2 (取整數)

1. Complete、Full B.T必為最小樹高
2. Complete B.T n1 = 1 or n1 = 0 => true
3. 有一個B.T，若Root的左右子樹皆為Complete B.T，則整棵B.T必為Complete B.T => False，換成Full、Binary Search Tree、HEAP、AVL Tree、Red-Black-Tree 皆False
4. 一個complete B.T任何Node之左右子樹必為Complete B.T => True，換成Full BST、AVL Tree、HEAP、皆True，Red-Black Tree => False
5. 在Complete B.T，若Node無左子點、右子點，必為Leaf => True
6. N個Nodes之Complete B.T高度必為最小化 => True (full 也是)

Strict B.T

在B.T中，Non-leaf必有2個子點，即Degree = 1, Node數 = 0個

1. Full B.T 必為 Strict B.T => True
2. Complete B.T 必為 Strict B.T => False
3. Strict B.T 必為 Full B.T => False
4. Strict B.T 必為 Complete B.T => False

B.T的表示方法

1. 利用Array表示，仿用Full B.T之Node順序，將Data填入Index中

優點:易於存取左、右子點、父點，針對Full、Complete B.T有充分利用空間

缺點:Node增刪不便、對於斜曲的B.T極度浪費空間