04/18/2011
*what we've done
- linked list (chain, circular, doubly)
- sparse matrix with linked list
- equivalence class with linked list
- Reading Assignments:
    more on list operations
    more on sparse matrix

Subject : .....

No. :

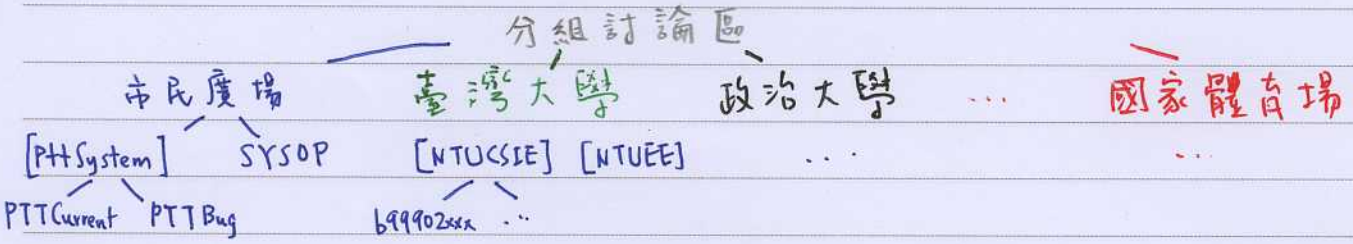
Date : ...../...../.....

(Sec 5.1) Trees.

array (indexed access) stack / queue (restricted access)

list (sequential access)

trees: hierarchical access



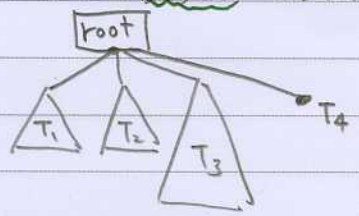
* similarly, directories / files in your filesystem

* formal definition :

$T_{tree} \equiv (root; T_1, T_2, \cdots, T_n)$

disjoint subtrees

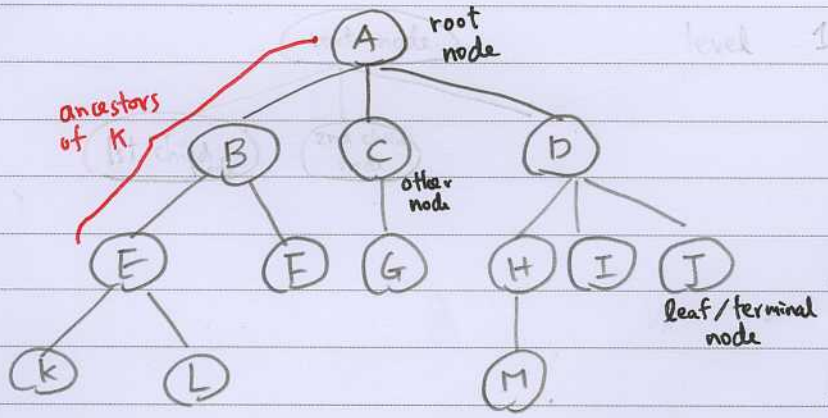
note: recursive definition



smallest tree $T^{(0)} \equiv (root;)$

disjoint : no cross links

*



level 1

level 2

level 3

level 4

(depth / height)

parent of B
grandparent of H

children of A
siblings of B

degree of node: # of children ; degree of tree: max degree of any node

Subject : .....

No. :

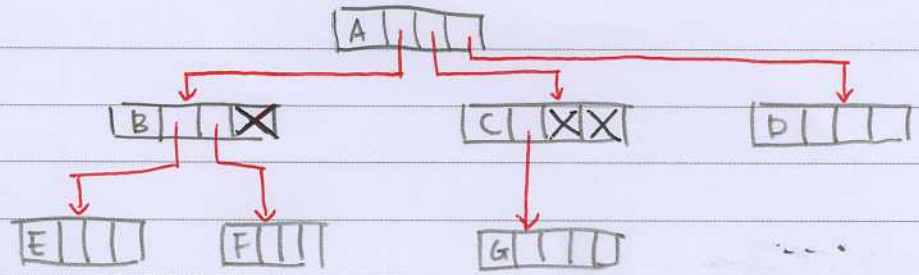
Date : ...../...../.....

* representing trees

by definition

$T \equiv (root, T_1, T_2, \cdots, T_n)$

```
struct Tree {
    Data root
    struct Tree * subtrees[MAXDEG];
};
```



. . . .

☒ : wasted

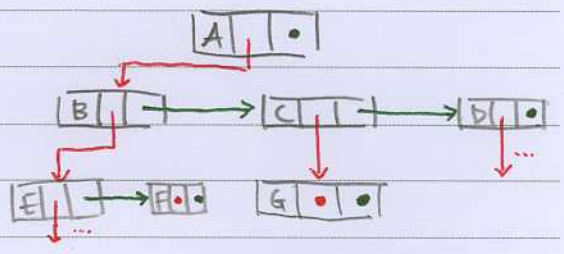
* how about dynamic array for $T_1, \cdots, T_n$

add { int n_subtree;
struct Tree * subtrees[]

- more complicated allocations
- still hard to Insert / delete

* how about linked list for $T_1, \cdots, T_n$

```
struct Tree {
    Data root;
    struct Tree * child;
    struct Tree * sibling;
};
```



called left-child right-sibling

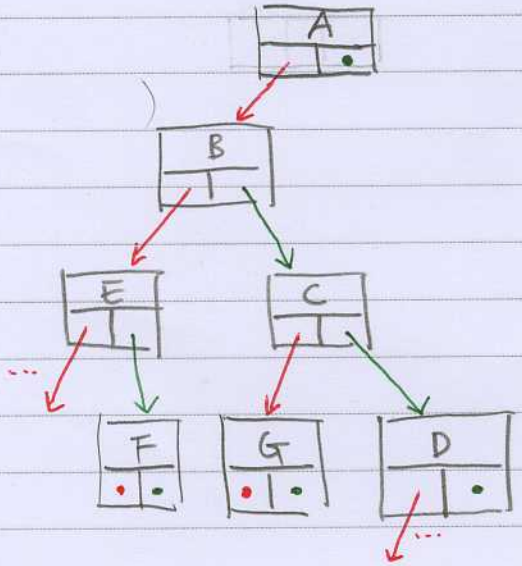
Subject : .....

No. :

Date : ...../...../.....

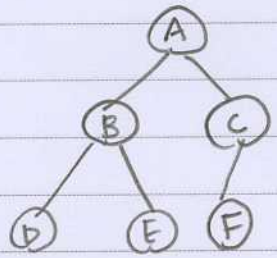
* how about call child by left

sibling by right



a binary tree that equivalently represents the original tree

* task: what's the binary tree representation of the binary tree



?

Subject : .....

No. :

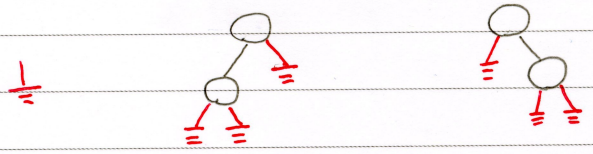
Date : ...../...../.....

* Binary Tree

tree w/ =2 binary subtrees

and ordered (left / right)

that can be empty



(usually don't draw ⏚)

* how many nodes / edges (branches)?

n_0 : 0 child (leaf)

n_1 : 1 child

n_2 : 2 child

$$\begin{cases} \#\,node = n_0 + n_1 + n_2 \\ \#\,edge = n_1 + 2n_2 = \#\,node - 1 \end{cases}$$

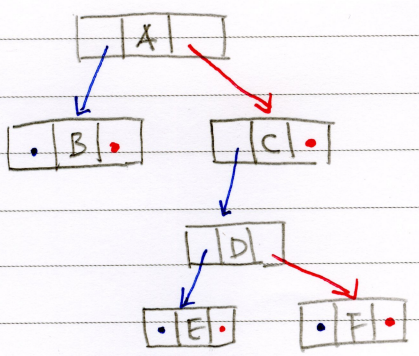
(root)

$$\Rightarrow \quad n_2 = n_0 - 1$$

* max # of nodes for depth-k tree? $2^k - 1$ (full bin. tree)

min # ? k

* binary tree representation



- can get whole tree from root
- tracing back?

  can add 'parent' link

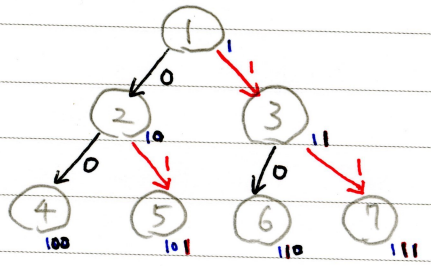
  (like doubly linked)

Subject : .....

No. :

Date : ...../...../.....

* full binary tree



node # in full binary tree in binary code :

1 [path code]

⇒ can "pack" the tree into an array

"→" "→" implicit

⇕ ⇕

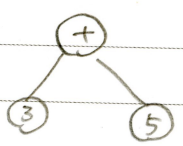
append 0 append 1

⇕ ⇕

*2 *2+1

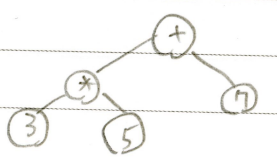
* Binary Expression Tree

3+5 ⟺



infix	postfix	prefix
3+5	35+	+35

3*5+7 ⟺



((3*5)+7)	35*7+	+*357

```
InfixPrint(T){
    InfixPrint(T->left);
    output(T->data);
    InfixPrint(T->right);
}
```

node: ① add termination condition
② add "()"

```
Postfix(T){
    Postfix(T->left);
    Postfix(T->right);
    output(T->data);
}
```

```
Prefix(T){
    output(T->data);
    Prefix(T->left);
    Prefix(T->right);
}
```

Double A

Subject : .....

No. :

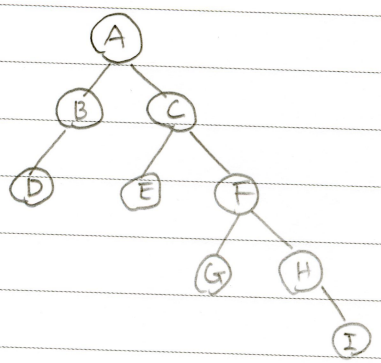
Date : ...../...../.....

* more generally (on any binary tree)

In-order Traversal

Post-order Traversal

Pre-order Traversal



in: D B A E C G F H I

post: D B E G I H F C A

pre: A B D C E F G H I