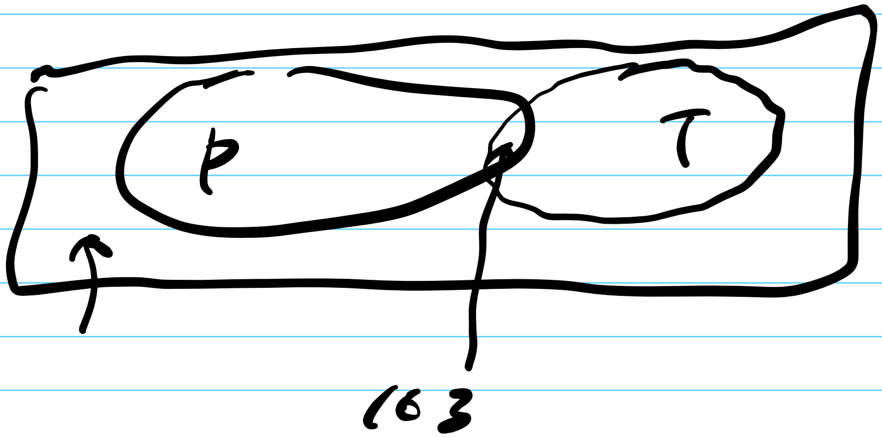
04/09 163 Lec1

0. Intro

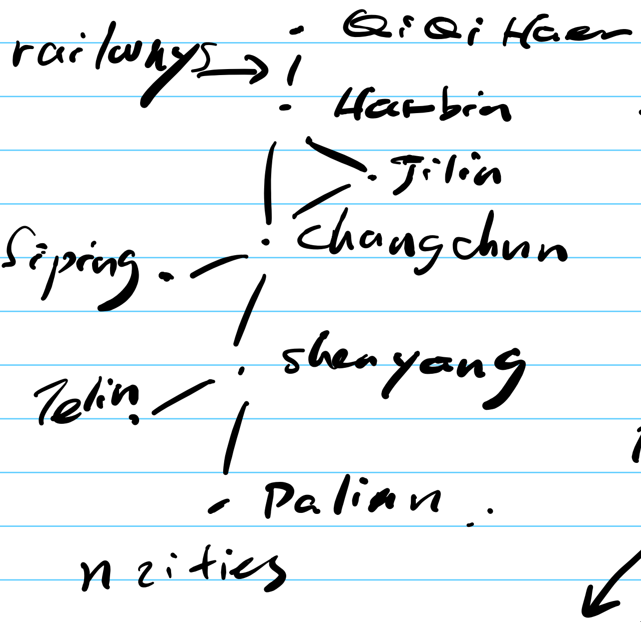
p: like coding
T: Theory



My Goal:

A starting/ending example.

Given: major cities in Dong Bei



→ I claim I can!
How to convince you?

→ A sequence of cities.
$[Q \to H \to J \to \cdots]$

proof size: $O(n)$

PCP theorem
(Probabilistic checkable pf)

new proof size: poly(n) : checking takes $O(\log n)$ bits

→ proof size : O(logn)

(comp. sound pf : Micali)

{ Hash function
Merkle tree

1. Intro Data Structures.

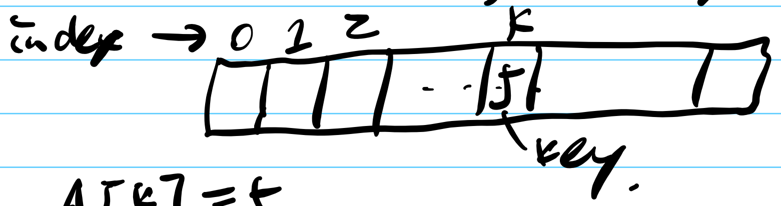
a. DS recap / warm up.

Purpose: organize data so that it can be accessed quickly & usefully

Examples:

- array (数组)

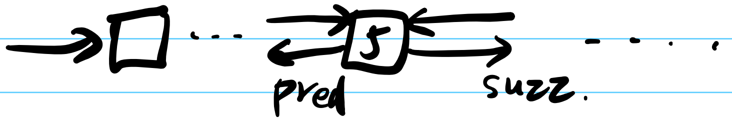
A : integer array



A[k] = 5

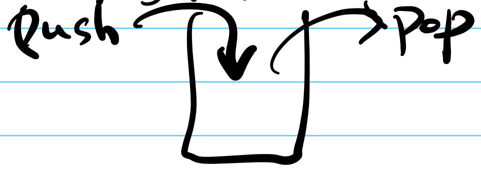
(random access)

- list (链表):
(linked)

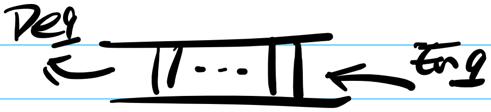


- stack / queue
(栈) (队列)

stack: FILO



Queue : FIFO

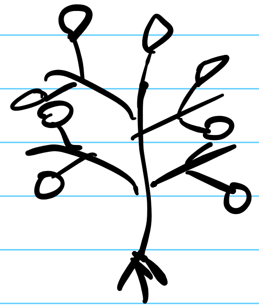


why so many? different DS support different set of op's ⇒ suitable for different types of tasks.

How to choose?

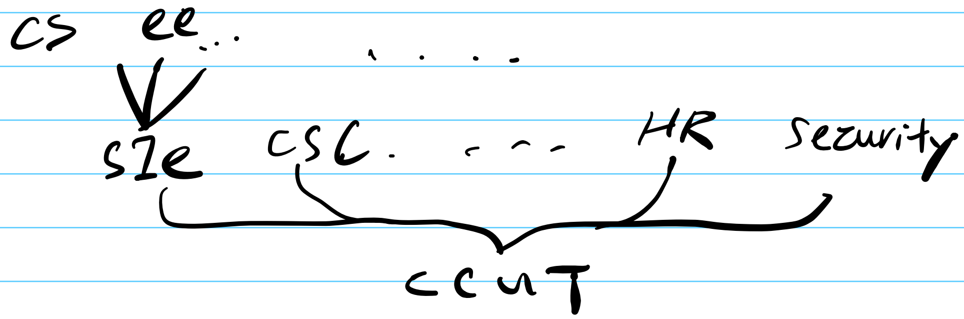
> Principle of Parsimony
>
> choose the simplest D.S. that supports all the op's required by your task.

b. Trees:

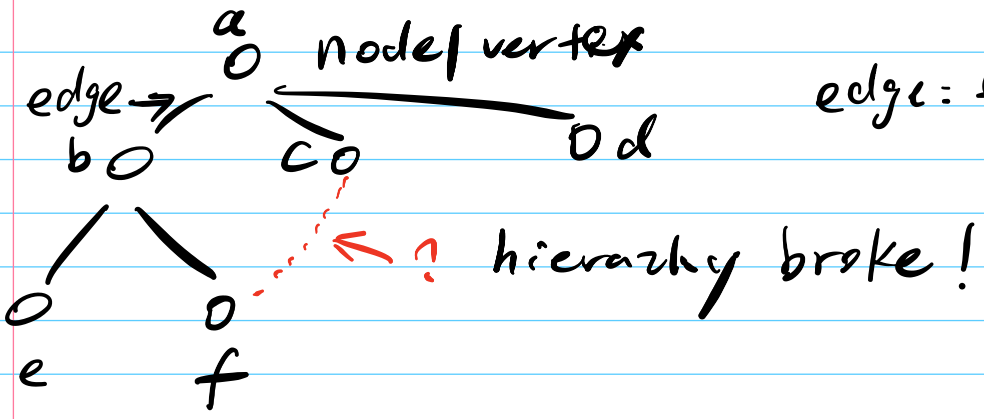


main characteristic: Hierarchical Structure

Example:



- Tree: what is it?



edge: {a, b}

DEF: A set of nodes & edges is a tree if it is connected & has no cycle

$\forall u, v, \exists$ PATH, $u - v$

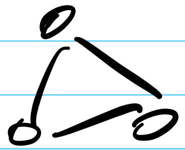
A sequence of nodes connected by edges:

$a - b - e$

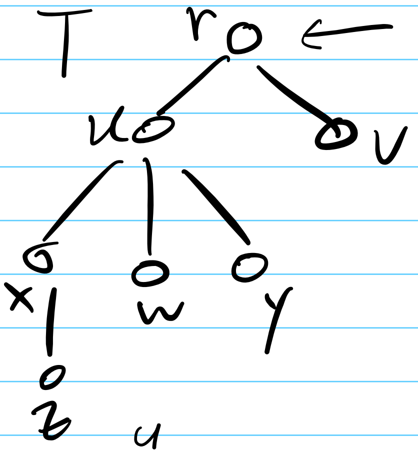
cycle: A path w/ same first/last node.

Ex: Tree w/ n nodes:

# edges = $n - 1$



- Glossory of a tree.



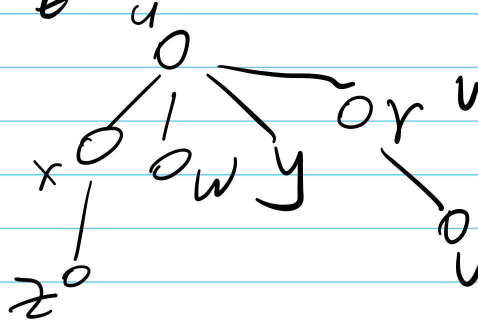
r : root ← only node w.o. parent

parent of u, v

u : child of r

u, v : siblings

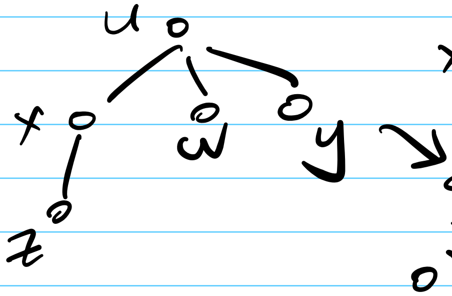
(chidren of same parent)



w : a leaf (no children)

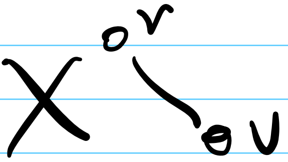
ancesters : u, r

Ex: can there be multiple paths between 2 nodes ? X



x, w, y, z = descendents of u.

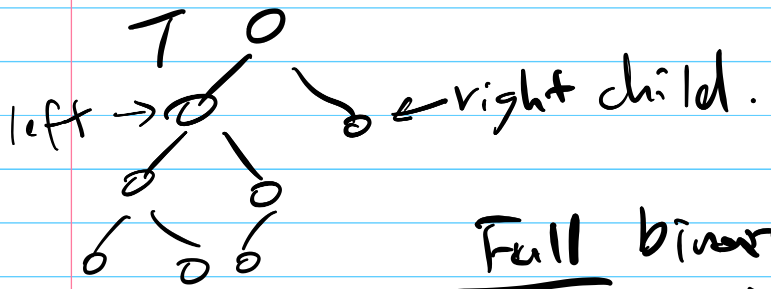
subtree of T :



set of nodes that includes a parent & all its descendents.

- Binary tree

DEF: T is binary tree if every node has ≤ 2 children.



Full binary tree:

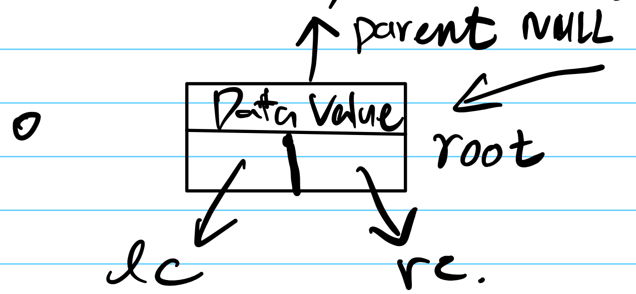
every node is either a leaf.
OR has 2 children.

c. Supported op's & Imp

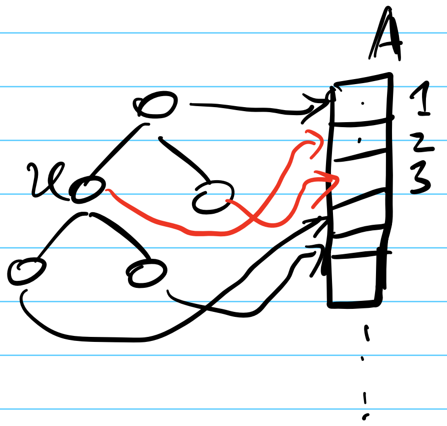
T: binary tree Time

- root(T)
- parent(u)
- lc(·), rc(·)
- insert( )
- remove( )

· Implementation by pointers:



· By a. Array



node i
$l_c = 2i$
$r_c = 2i+1$

2. Binary search Tree (BST)

a. Sorted Array.

A:

3	5	7	11	17	23	35	41	67

$|A| = n$
$(n=9)$.

· Supported op's.

opr's:	Running time
Search $(15 \overset{?}{\in} A)$	$O(\log n)$. (Binary Search)
Select (i)	$O(1)$
min/max:	$O(1)$
Rank (value): # of keys $\le$ input value	$O(\log n)$ [Ex.]
rank(33) = ~~5~~ 6	
output in sorted order	$O(n)$

☺ FAST ✓

☹ Insert/Delete ?

$\Theta(n)$

BST := sorted array +

fast ins/del $\underline{(\log n)}$