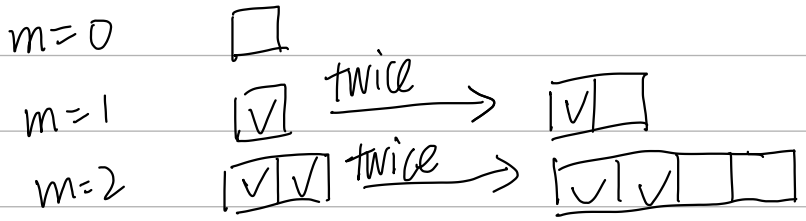
ADS 2.

amortise analysis (均摊分析)

Dynamic Array (vector)

- A[i]
- Insertion
- O(n) space



Insertion: $T = O(N)$ (copy)

but the frequency of the worst case is small

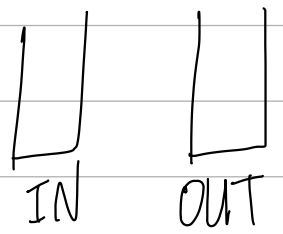
$T(m) = cm + 3c + 6c + 12c + \cdots + 2^k \cdot 3c$

(其中c是一个常数 $k = \log m$)

$\leq cm + 3c \cdot (2^{\log_2 m + 1} - 1) \leq cm + 6cm = 7cm$

则 $\frac{T(m)}{m} \leq 7c = O(1)$ ⇒ 均摊到成本为O(1)

Two-Stack Queue



enque(x): IN.push(x)

deque(): If OUT not empty:
    OUT.pop()
Else:
    While IN not empty:
        OUT.push(IN.pop())
    OUT.pop()

enque: O(1) in the worst case
deque: O(n)

Accounting method

amortized cost	actual cost	credit
10	2	10-2=8
10	100	10-100=-90
		.....

$\sum$ amortized cost = $\sum$ actual cost + $\sum$ credit

$\sum$ credit ≥ 0

$\Rightarrow \sum$ amortized cost $\geq \sum$ actual cost

	actual cost	credit ~	amortized ~
enque()	c	2c	3c
deque()	2c. #elements moved from IN to OUT +c = 3c	-2c. #ele.	c

證明其 ≥ 0.

⇓

均攤時間是 O(1) 的

potential function

$\Phi(i)$ = total # credit remaining

$\Delta\Phi = \Phi(i) - \Phi(i-1)$

amortized cost of operation O_i = actual cost of operation O_i + $\Phi(i) - \Phi(i-1)$

$\forall i: \Phi(i) \geq \Phi(0)$

Splay Tree

:($\Theta(n)$ in worst case

:) $O(\log n)$ amortized cost

easy to implement

no extra space

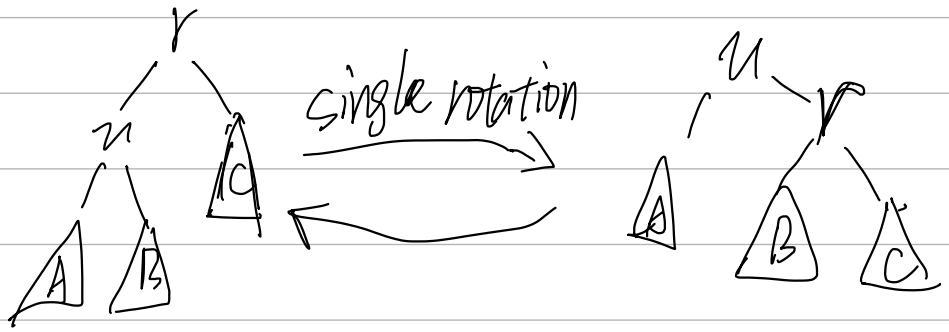
adaptive

BST with operation "Splay"

splay(u): rotate node u to the root

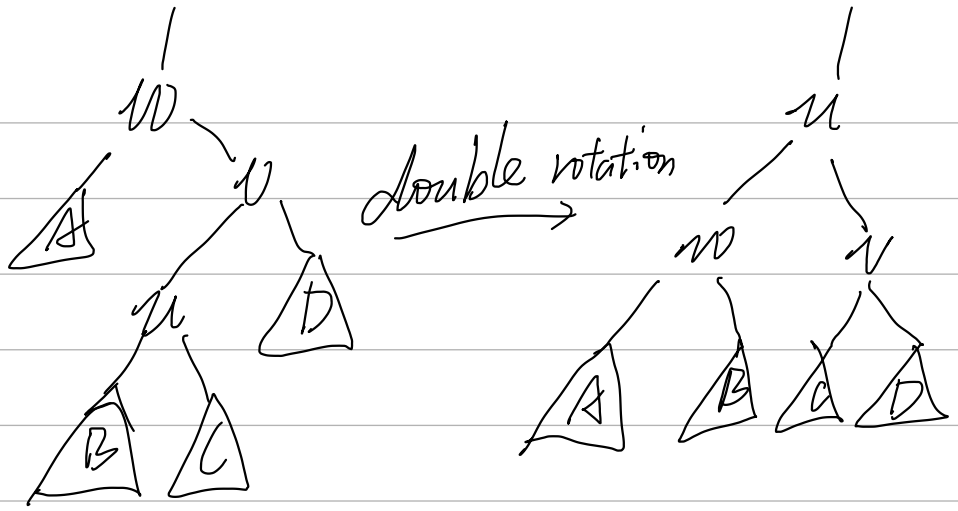
repeat the followings until u becomes the root

case 1: u is a child of the root

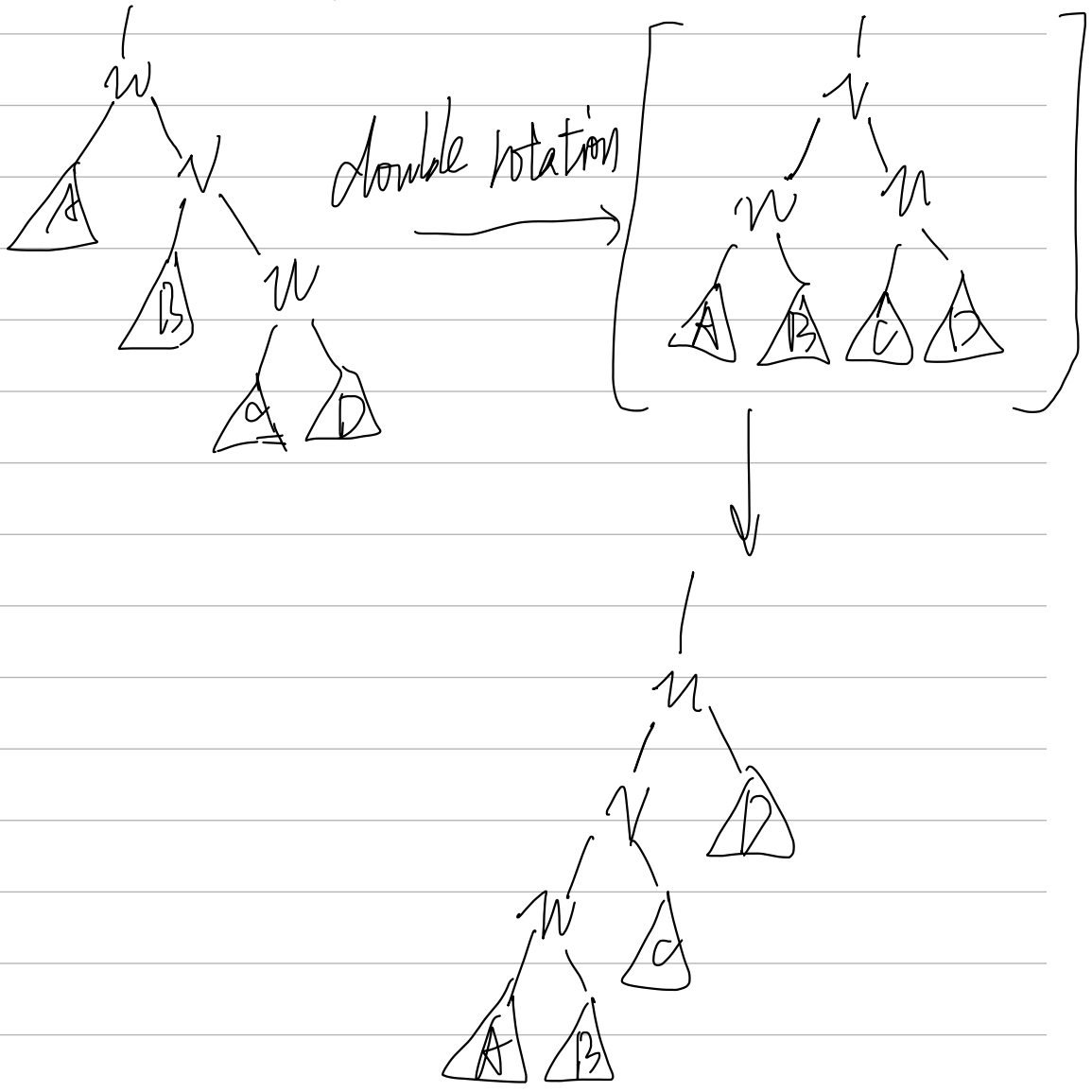


case 2: u has a grandparent

Case 2.1 Zig-Zag (지그재그)



Case 2.2. Zig-Zig



(a) findkey

1. find as in BST h.

2. splay the node to the root #rotation = h

O(h) = O(#rotation)

(b) Insertion

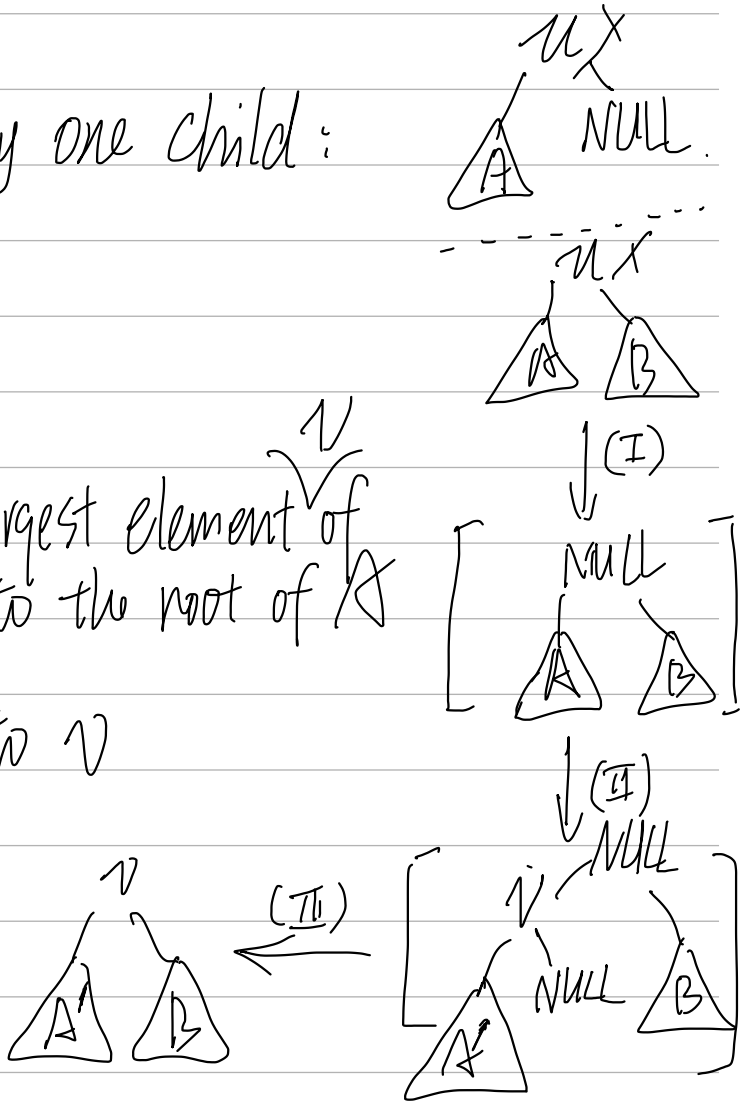
1. insert as in BST h

2. splay the new leaf node to the root

(c) deletion

1. splay(u)

2. If u has only one child:
   delete(u)
   Else:
   (I) delete(u)
   (II) splay the largest element of subtree A to the root of A
   (III) attach B to v



Splay Tree 不一定是平衡的，但是我们可以证明其均摊后是 $O(\log N)$ 的。

actual cost = $c \cdot$ #rotations