previous: worst-case bound for a single operation.

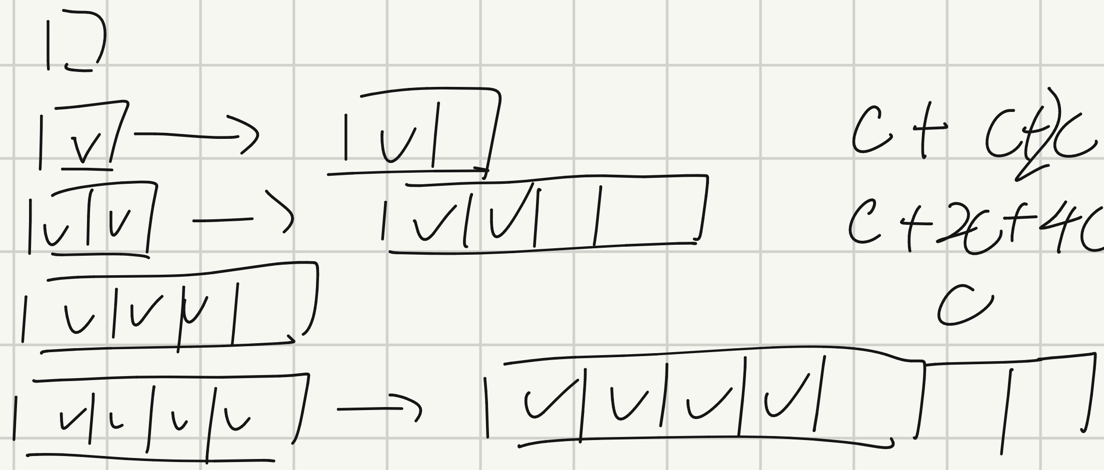
amortized: worst-case bound a sequence of operations

1. Dynamic array

A[i].

Insertion.

O(n) space.



$m=0$

$m=1$ $\quad c + c + 2c \Rightarrow c + 3c$

$m=2$ $\quad c + 2c + 4c \Rightarrow c + 6c$

$m=3$ $\quad c$

$m=4$ $\quad c + 4c + 8c \Rightarrow c + 12c$

Insertion: $O(n)$ worst case, when full.

$T(m)$: cost of the worst sequence of insertion

$$= cm + 2^0 \cdot 3c + 2^1 \cdot 3c + \cdots + 2^{\log_2 m} \cdot 3c$$

$$= cm + 3c\,(2^{\lfloor \log_2 m \rfloor + 1} - 1)$$ $\because \log_2 m$ expansion

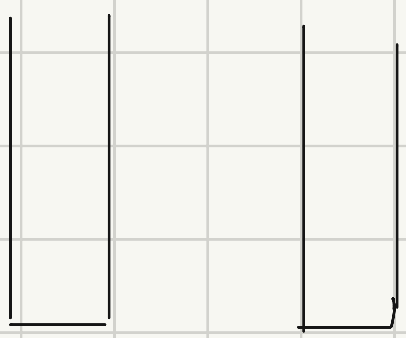
$$\leq cm + 6cm$$ aggregation method

$$= 7cm.$$

$$\frac{T(m)}{m} = 7(\leftarrow O(1)$$

↑ amortized cost

2. Two-stack Queue



In OUT

enqueue (x):
    IN.push (x) $O(1)$

dequeue ( )
    If OUT not empty
        OUT. pop ( )
    else if IN not empty
        While IN not empty.
            x = IN. pop ( )

OUT.push (x)

OUT.pop ( ).

Accounting method

amortized cost = actual cost + credit

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

$\sum$ credit $\geqslant 0$.

$\sum$ amortized $\geqslant$ $\sum$ actual cost

	actual cost	credit	amortized
enqueue	c.	2c.	3c
dequeue:	2c · # element moved + c	−2c · # element moved	c.

description:

$\Phi(i)$ = total # credits in the bank after the ith operation

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

$\Phi(i) \geqslant \Phi(0) = 0$.

$\Rightarrow$ potential function

essence: relation between push and pop stack

$\Phi(Q) = (\#\text{ elements in Stack IN}) \cdot 2c$

for any sequence of operation: $o_1 \quad o_2, \quad \cdots \quad o_m$

$Q_0 \quad Q_1 \quad Q_2 \quad Q_m$

empty.

$\Phi(Q_0) = 0 \qquad \Phi(Q_i) \geq 0.$

if O_i = enqueue

$\hat{d_i} = d_i + 2c = c + 2c = 3c$

if O_i = deque

$\hat{d_i} = c + 2c \cdot \#\text{ ele moved} - \#\text{ ele-moved} \cdot 2c = c$

Define

Given k types of operations $1, \cdots, k$ with actual cost $T_1(D) \cdots T_k(D)$ (insertion of a BST $T(D)$ = height of D)

We say they have amortized cost $A_1(D) \cdots A_k(D)$ if for any $m > 0$ and for any sequence of m operations $O_1, \cdots O_m$

$D_0 \quad D_1 \quad D_m$

$$\sum_{i=1}^{m} A_{type(O_i)}(D_{i-1}) \geq \sum_{i=1}^{m} T_{type(O_i)}(D_{i-1})$$

Potential Function

$\Phi: \mathcal{D} \to \mathbb{Z} \qquad \Phi(D) \geq \Phi(\text{empty})$ for any $D \in \mathcal{D}$

$$A_t(D) = T_t(D) + \Phi(D') - \Phi(D)$$

D': after operation t

Splay Tree:

$O(n)$ in worst case

$O(\log n)$ amortized cost

1> easy to implement

2> no extra space (RBT need colors, BBST need height,

3> adaptive (eg: m find (4) operation

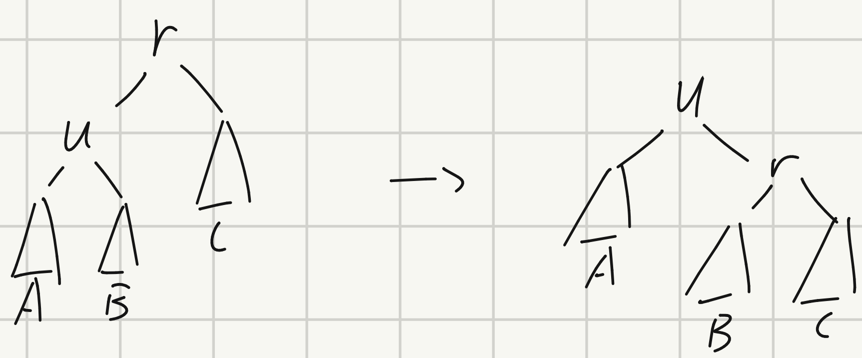
$O(m + \log n)$

BST

splay(u) :

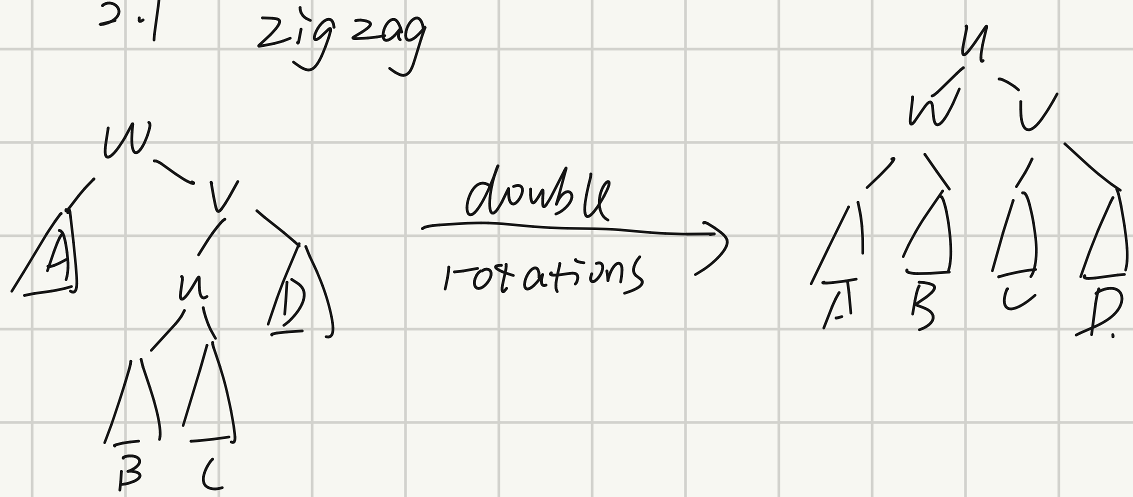
repeat the follows until u is the root

case 1. u is a child of the root

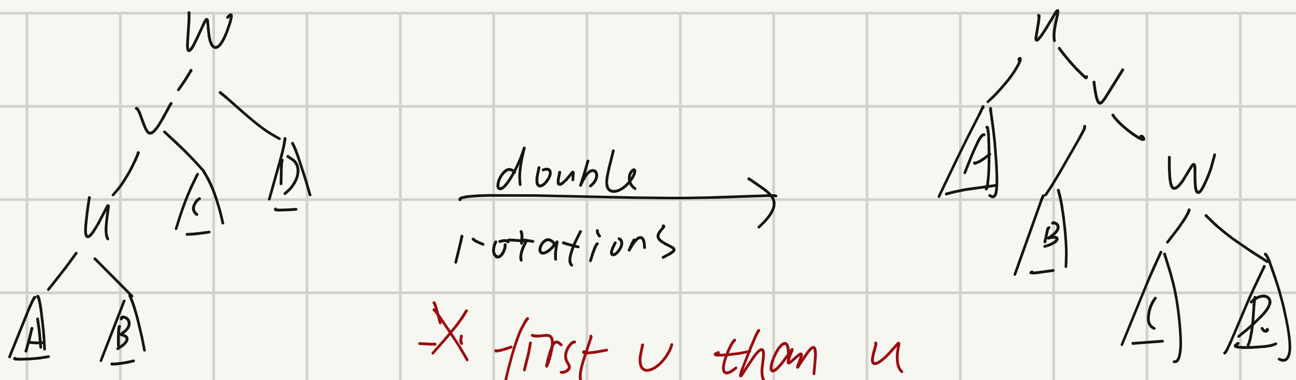


case 2. u has a grandparent

case 2.1 zigzag



case 2.2 zig zig



× first v than u
different from AVL Tree

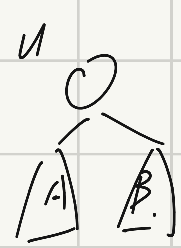
find Key

1. find as in BST
2. splay the node you found

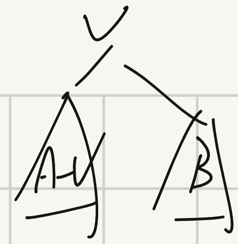
Insert
1. insert as in BST
2. splay the new node

delete (u)
splay (u)
if u has only one child:
delete u.

else if u has 2 children:
delete u
splay the largest element v in A



attach B to V



observation:

actual cost of each operation is $C \cdot \#$ rotations
$\Downarrow$ constant

amortized cost $= C \cdot \lg n \leftarrow$ goal. (常数已经最小了 还用得化简)

$\Delta\Phi = c \cdot \lg n - c \; \#$ rotations (From potential function)

Given a BST T

for each $u \in T$

$\text{size}(u) = \#$ nodes in T_u.

rank $r(u) = \lg(\text{size}(u))$

define $\Phi(T) = c \cdot \sum_{u \in T} r(u)$ $\qquad \Phi(\text{empty}) = 0$

$\Phi(T) \geq 0$

Lemma:

Let T be a splay tree. Let $u \in T$

Let T' be the tree obtain. from T by performing splay (u)

$$\Phi(T') - \Phi(T) \leq 3c\,[\,r'(u) - r(u)\,] - 2c\,(\#\text{rotations} - 1)$$

$r'(u)$: rank of u in T'

Then: Find key: actual cost $= c \cdot \#$ rotations

$\Delta\Phi = 3c\,[\,\underbrace{r'(u)}_{\lg n \text{ after spray}} - r(u)\,] - 2c\,(\#\text{rotations} - 1)$

$\leq 3c \lg n - 2c \; \#\text{rotations} + 2c$

amortized cost $\leq 3c\lg n - 2c\#\text{rotations} + 2c + c\#\text{rotations}$

$\leq 3c\lg n + 2c$

$\leq 5c\lg n$

Insertion: 1. insert as in BST

2. splay the new node

actual cost = $c\#$ rotations

amortized cost = $\#\text{rotations} + 3c[r'(u) - r(u)] - 2c(\#\text{rotations} - 1)$

$\leq 3c\lg n - (2c-1)\#\text{rotations} + 2c$

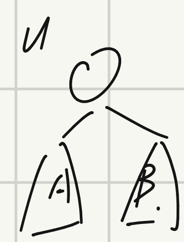
$\leq 3c\cdot\lg n + 2c \leq 5c\lg n$

delete:

D1 1. splay (u)

2. if u has only one child:

3 delete u.

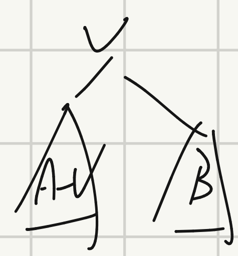


4 else if u has 2 children:

D2 5 delete u

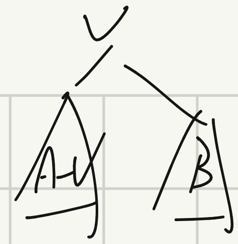
D3 6 splay the largest element v in A

D4 7 attach B to v



(O(1)) ?

7 attach B to v



$$\Delta_2 = -\lg(|A| + |B| + 1)$$

$$\Delta_3 = \lg(|A| + |B|) - \lg(|A| - 1)$$

$$\Delta_2 + \Delta_4 \le 0$$

$$\Delta\Phi \le \Delta_1 + \Delta_3$$

amortized cost $\le$ actual cost +

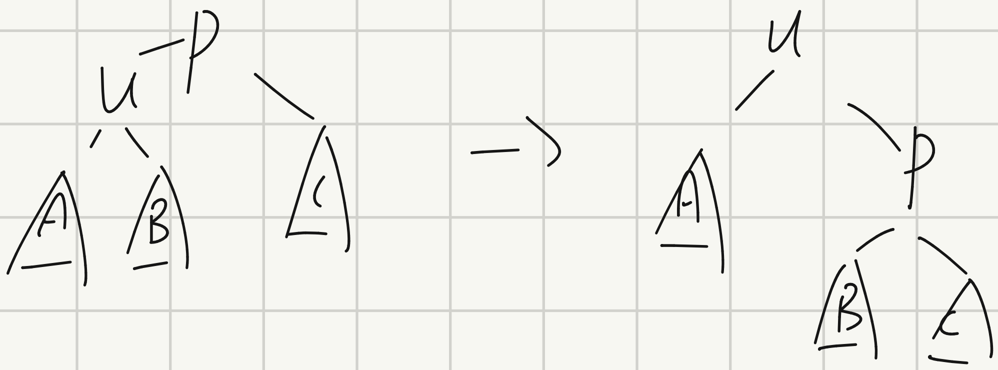
$\le$ # rotations in step 1 + Δ_1 + # rotations in step 6 + Δ_3

$$\le 3c\lg n + 2c + 3c\lg n + 2c$$

$$\le 10c\lg n$$

demonstrate Lemma

case 1: u is a child of root

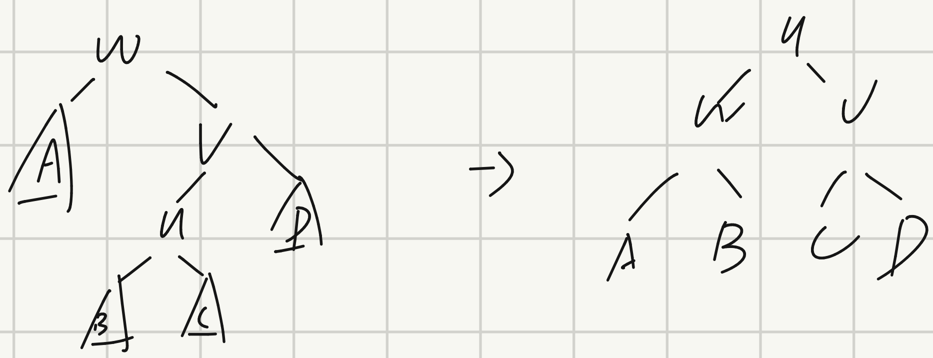


$$\frac{\Delta\Phi}{c} = r'(u) - r(u) + r'(p) - r(p)$$

$$\le r'(u) - r(u)$$

$$\le 3[r'(u) - r(u)] - 2(\#\ \text{rotation} - 1)$$

case 2.1 zig-zag



$$\frac{\Delta\Phi}{c} = r'(u) - r(u) + r'(w) - r(w) + r'(v) - r(v)$$

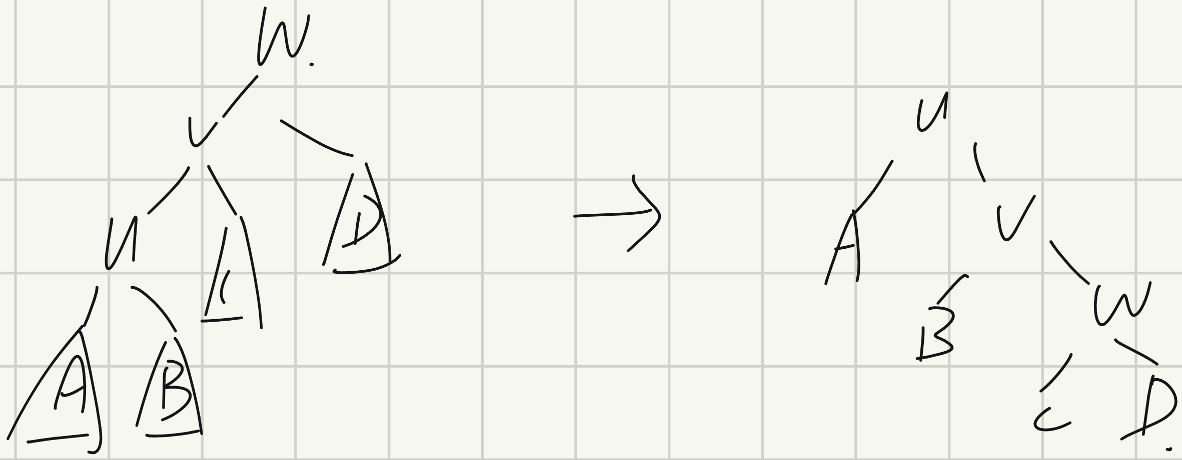
$$= r'(v) + r'(w) - r(u) - r(v)$$

$$\le r'(v) + r'(w) - 2r(u)$$

$$\le 2r'(u) - 2r(u) - 2 \le 3(r'(u) - r(u)) - 2(\#\ \text{rotation} - 1)$$

$$\left(\begin{array}{c} size'(v) + size'(u) \ \cancel{+} = size'(u) \\ \Downarrow \\ r'(v) + r'(w) \le 2r'(u) - 2 \end{array} \right)$$

case 3.1 zigzig



$$\frac{\Delta\Phi}{c} = r'(u) - r(u) + r'(v) - r(v) + r'(w) - r(w)$$

$$\leq r'(w) - r(u) + r'(v) - r(v)$$

$$\leq r'(w) + r'(u) - 2r(u)$$

$$= r'(w) + r(u) + r'(u) - 3r(u)$$

$$\text{size}'(w) = \text{size}'(C) + \text{size}'(D) + 1$$

$$\left(\begin{array}{l} \text{size}(u) = \text{size}(A) + \text{size}(B) + 1 \\ \text{size}'(w) + \text{size}(u) + 1 = \text{size}'(u) \end{array} \right)$$

$$\Rightarrow r'(w) + r(u) \leq 2r'(u) - 2.$$

$$\leq 3[r'(u) - r(u)] - 2 \leq 3[r'(u) - r(u)] - 2(\#\text{rotations} - 1)$$

12:8讲 平衡树.