

\downarrow finite sequence of cells of same bit, ascending address.

CPU: contains regs

① Initialization ② Arithmetic

③ Comparison ④ memory access

Big-O notation (bb1快)

$\exists C_1, C_2 > 0, \forall n \geq C_2, f(n) \leq C_1 g(n)$

存在大于 C_1, C_2 , 使所有大于 C_2 的

(实数都 $C_1 g(n)$ 更大 ($C_1, C_2 \rightarrow \infty$))

Big-O (bb1慢) 上面慢 (=)

$f(n) = O(g(n)) \rightarrow f(n)$ 更慢

$f(n) = \Omega(g(n)) \rightarrow f(n)$ 快; ④ 相同

Binary search

left=1, right=n; repeat

mid \leftarrow (left+right)/2

if $c[\text{mid}] = t$ return true

else if $t < c[\text{mid}]$ right=mid-1

$\rightarrow [\text{mid}]$ left=mid+1

until left > right / return false

$g(n) = 2 + q(1 + \log_2 n) = \Theta(\log n)$

worst time $\log n$ best $O(1)$

$1 < \log n < n < n \log n < n^2 < n^3 < 2^n <$

$n! < n^n$ arr worse best space

selection n^2 n^2 n^2 n^2 n^2 n^2 n^2 n^2 n^2 x

insertion n^2 n^2 n^2 n^2 n^2 n^2 n^2 n^2 n^2 v

bubble n^2 n^2 n^2 n^2 n^2 n^2 n^2 n^2 n^2 v

merge $n \log n$ v

quick $n \log n$ n^2 n^2 n^2 n^2 n^2 n^2 n^2 n^2 x

selection: ① 找到最小的元素, 支换至第1位 ② 找到 $[2, n]$ 最小的元素, 前一个提严谨正确结果.

for $i[1, n-1] = k \leftarrow i$ For $j[i+1, n]$:

if $A[k] > A[j]$, $k \leftarrow j$; swap $A[i][j]$

insertion: 对第*i*个元素, 像插牌一样

插入前 $[1, n-1]$, (前 $[1, n-1]$ 调)

for $i[1, n]$ for $j[0, n-i-1] =$ bubble

if $arr[j] > arr[j+1]$ swap swap

if no swap in a loop, break

每次最大的冒泡冒上去

Mergesort($A[]$, n)

if $n > 1$ $p \leftarrow \lfloor n/2 \rfloor$ else return

$B \leftarrow [1, p] \leftarrow A[1-p]$

$C \leftarrow [1, n-p] \leftarrow A[p+1, n]$

$MSC(B, p), MSC(C, n-p)$

$A[1, n] \leftarrow \text{Merge}(B, p, C, n-p)$

Merge(L, R, nR)

$n \leftarrow nL + nR$

$A \leftarrow \text{new } M[n]$

$i \leftarrow 1; j \leftarrow 1$

for $k \leftarrow 1$ to n :

if ($i \leq nL \& j \leq nR$) $L[i] < R[j]$)

$ACK \leftarrow L[i]; i++$

else $ACK \leftarrow R[j]; j++$ return A

Quicksort($A[lo, hi]$, $lo=1, hi=n$)

$P \leftarrow \text{partition}(A, lo, hi)$

Quicksort($A, lo, p-1$)

Quicksort($A, p+1, hi$)

$p \leftarrow \text{random}(lo, hi)$

$\text{pivot} \leftarrow A[p]$

$L \leftarrow lo, R \leftarrow hi, A \leftarrow \text{new } A$

for $i[lo, hi]$

if ($A[i] < \text{pivot}$) $A'[l++]$ $\leftarrow A[i]$

else $A'[R--] \leftarrow A[i]$

$A'[l] \leftarrow \text{pivot}$

$A[lo, hi] \leftarrow A'$ return L

$T(n) = aC T(n/b) + f(n)$

$T(n) = aT\left(\frac{n}{b}\right) + O(n^\alpha), n \geq 2$

① $\log_b n < r, T(n) = O(n^\alpha)$

② $\log_b a = r, T(n) = O(n^r \log n)$

③ $\log_b a > r, T(n) = O(n^{\log_b a})$

Binary $T(n) \leq T\left(\frac{n}{2}\right) + C$

Merge $T(n) = 2T\left(\frac{n}{2}\right) + O(n)$

数组优 ① (1) 询问 ② 随机读写

③ 内存紧凑. 例 ① 申请初始大小 ② 很难/无法调整大小 ④ 很难扩缩 (API)

链表 ① 无限增长 ② 易插/删

① 无法随机访问 ② 要 next 操作

③ 容易内存泄漏 ④ 不紧凑

循环链表

traverse(A)

if ($A = \text{null}$) return

else print $A.\text{val}$, traverse($A.\text{next}$)

要倒叙则先 traverse print

stack \rightarrow 先进后出, queue 先进先出

priority queue \rightarrow 先出.

Ring queue

满 $\text{front} = (\text{rear}+1) \% \text{max}$

空 $\text{front} = \text{rear}$

入队 $\text{rear} = (\text{rear}+1) \% \text{max}$

String \rightarrow tt

Brute Force 暴力 $O(mn)$

Karbin-Karp 哈希值, $O(n)$, worse (mn)

Finite State Automata.

aba abaca \rightarrow next(c);

next[D] null 0

[1] "b" 0

[2] "ba" 1

[3] "baa" 2 ... 3, 0, 1,

ababacaba {b, 0, 1, 2, 3, 0, 1, 2}

☆ 在表达算法复杂度时注意不同符号
例：其中 $O(n \log N)$

生成 next =

$\text{next} = \text{int}[length], l \leq 0, r \leq 1$

while $r < length$

if ($\text{string}.c[r] = \text{string}.c[l]$) {

$\text{next}[r] = l + 1$

$l++$

} else if ($l == 0$)

$\text{next}[r] = 0, r++$

else $l = \text{next}[l - 1]$

ababacaba transition function.

0 1 2 3 4 5 6 7 8

a 1 3 1 5 1 7 1 9

b 0 2 0 4 0 4 0 8 0

c 0 0 0 0 0 6 0 0 0

① 先把对应的 1~9 填上

② 对应 $\text{next}[n-1]$

把 $\text{next}[n-1]$ 行复制过来

树: ancestor 包括本身.

用 dfs 跑一圈, 记录出入栈时间

则向以 $O(1)$ 决定父子关系

中 + (前/后) 可以确定一个树

前: GDAFEMHZ

中: ADEFGHMZ

前 ① 为根节点

同理 ② 为左树根

A 在中的第一, 没有子结点

FE: F 先.

① D E M Z

② G H

③ B C

④ A F

⑤ E H

⑥ G M

⑦ B C

⑧ D E

⑨ F H

⑩ A G

⑪ B C

⑫ D E

⑬ F H

⑭ A G

⑮ B C

⑯ D E

⑰ F H

⑱ A G

⑲ B C

⑳ D E

⑳ F H

⑳ A G

⑳ B C

⑳ D E

⑳ F H

⑳ A G

⑳ B C

⑳ D E

⑳ F H

Binary Search Tree:

$O(n)$ Space

$O(h)$ per search/query

$O(h)$ per insertion

$O(h)$ per deletion

☆注意，一切没有
平衡 balanced 的
BST 最坏情况为 $O(h)$
此时树退化为链表

Predecessor Query

找到比 x 小的最大数：

① 根节点开始

根 $> x$, 向左,

根 $< x$, 向右, 两次为
相同时直接中断
如果向左, 右后不存在此节
点, 则返回上一节点 $O(h)$

Successor Query

找到比 x 大的最小数：

若 x 有右子数, 则对它是
右子数上的最小节点

BST Deletion:

Case 1: 叶子节点直接删

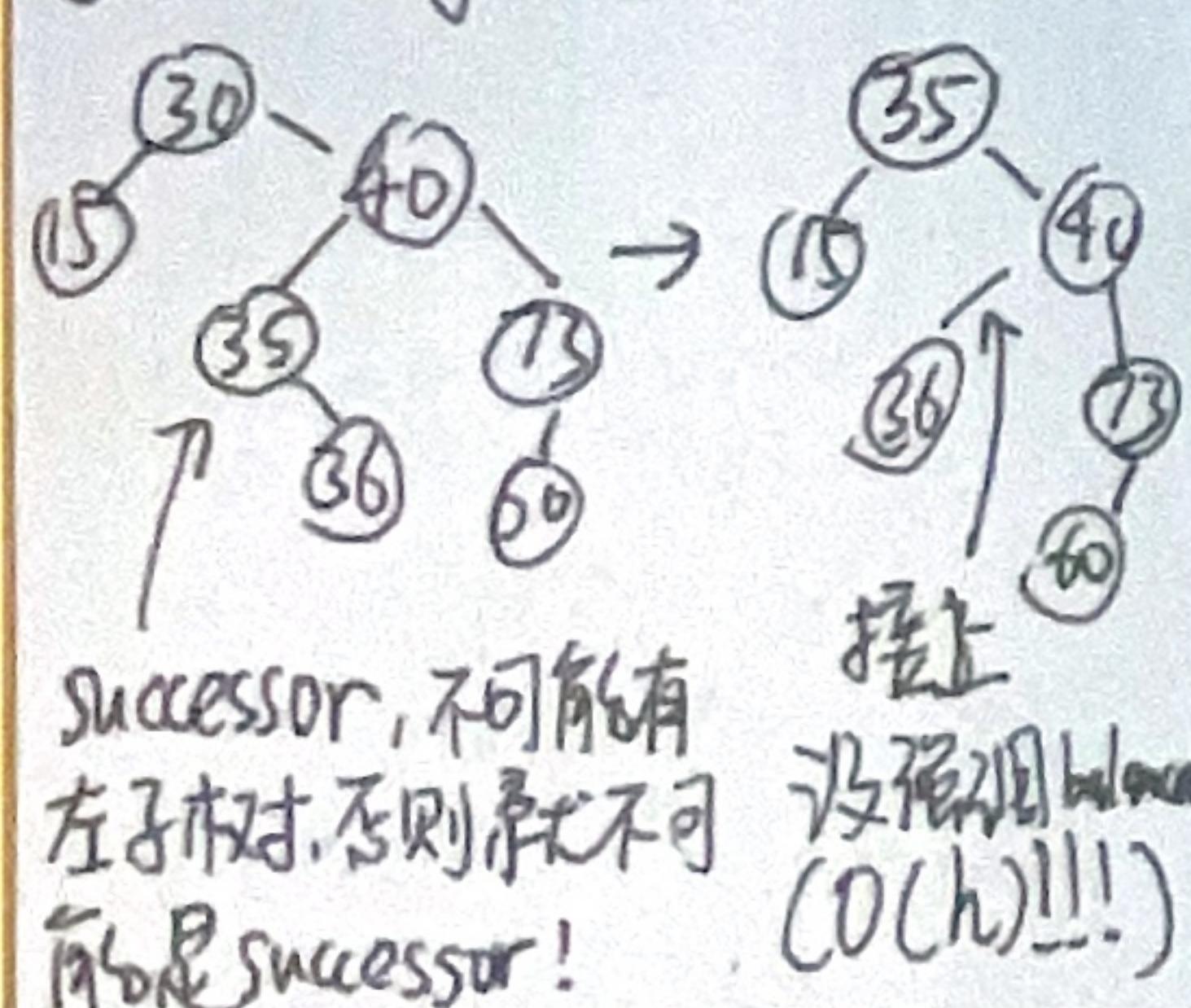
Case 2: U 有右子树:

找到 U 的 Successor, 换了后

Case 3: U 只有左子树

直接删了接上左子树

Case 2 e.g.: delete 30



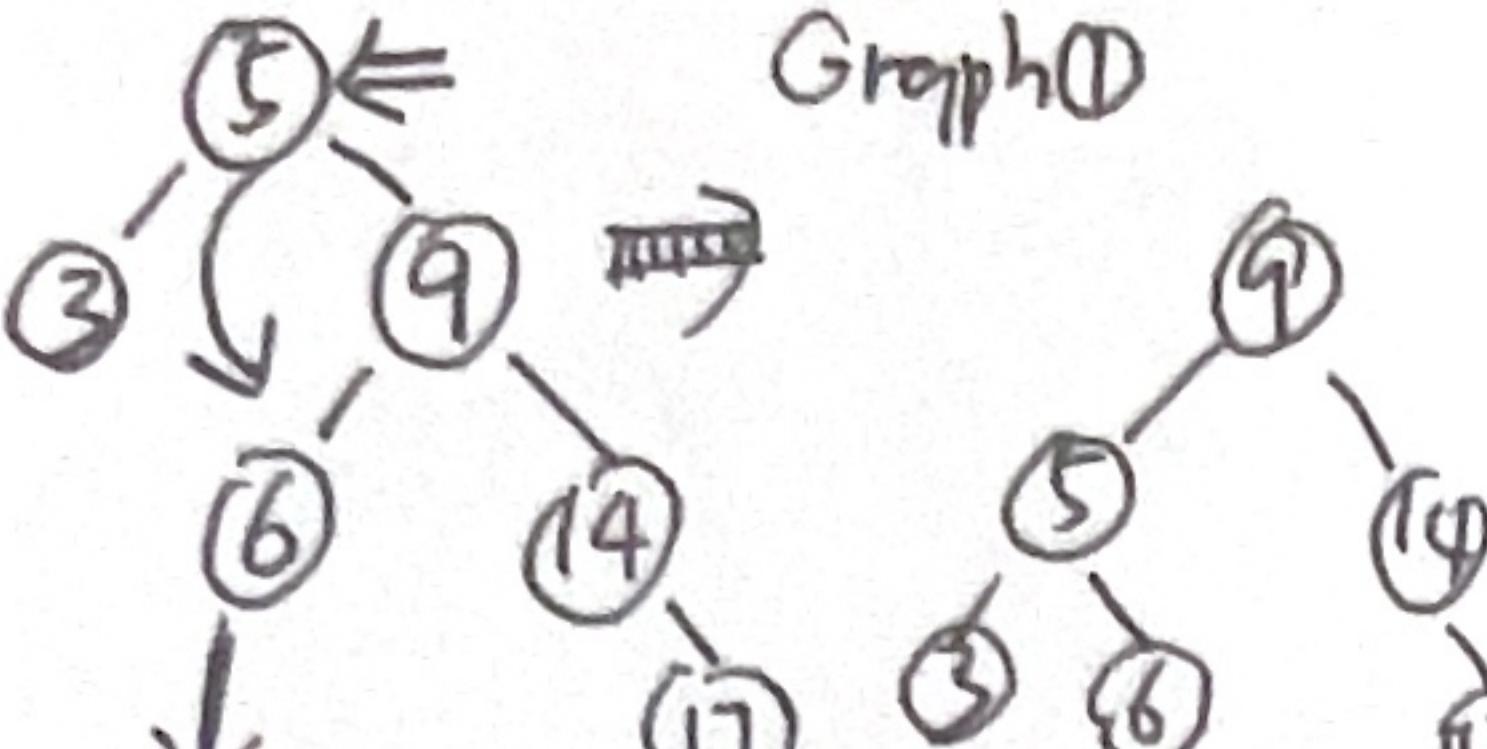
Balanced! BST

$O(\log n)$ 证明: $n_h = 1 + n_{h-1} + \dots$

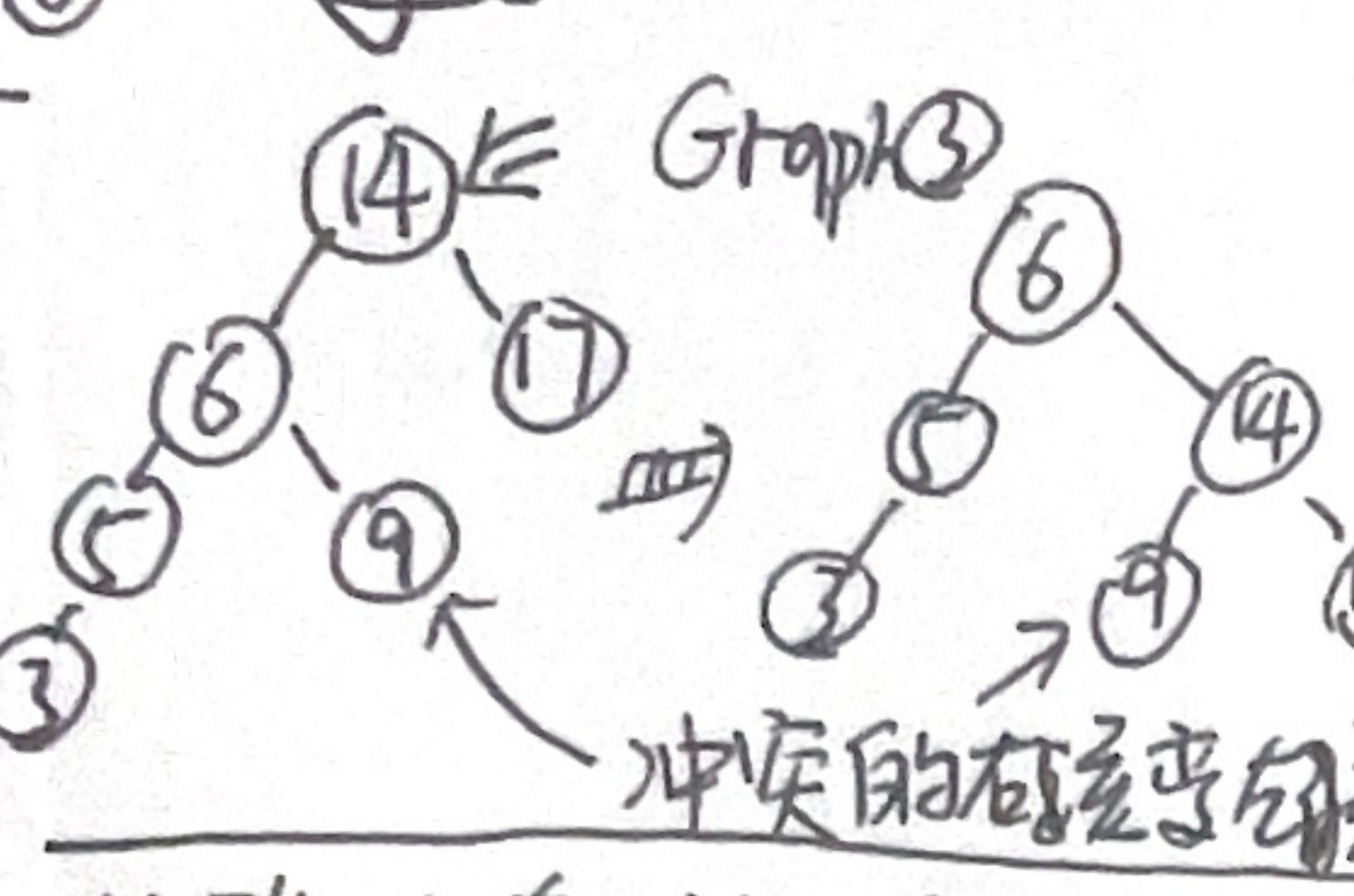
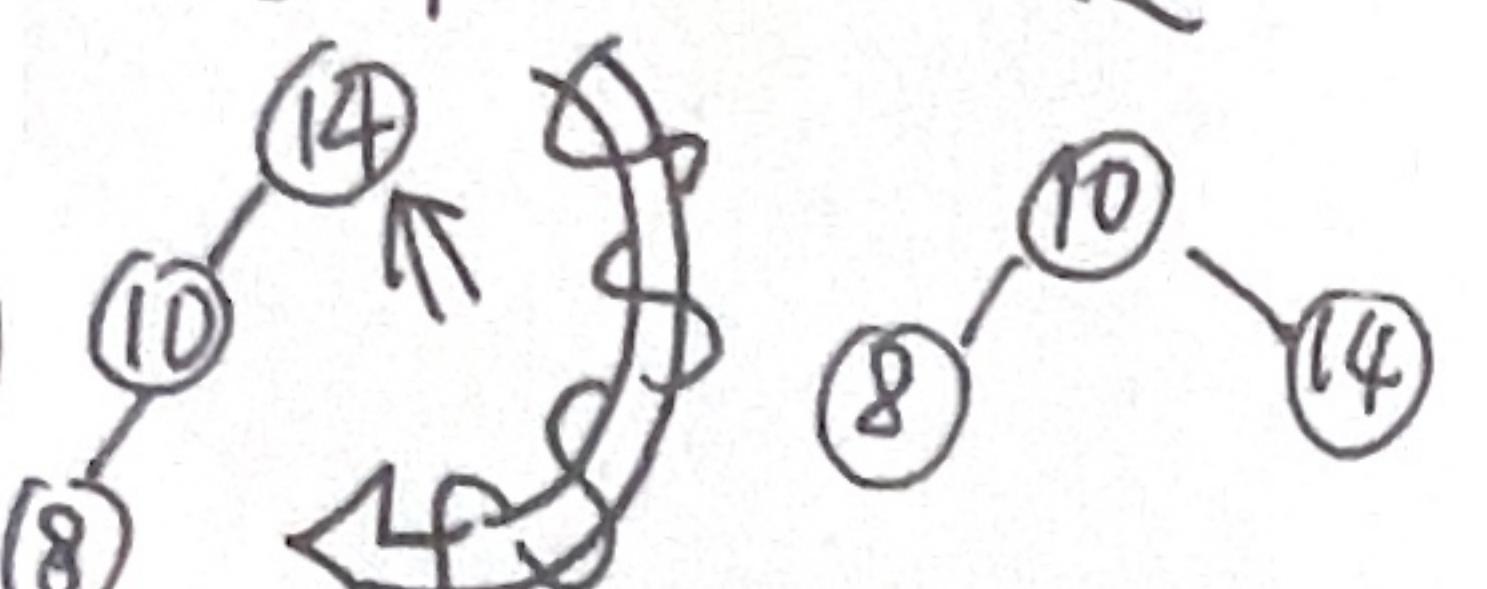
$n_h \sim (1+\frac{1}{2})^h$

$\therefore O(2^h) = n = O(2^h), h = O(\log n)$

定义: 左旋: 节点逆时针转



右旋: 节点顺时针转



- 如 Graph ③, 做 -LR 变
- RR 型, 失-2, 右孩子-1
- LR 型, 失-2, 左孩子-1

- 右旋左孩子, 然后右旋 ④
- 左旋右孩子, 然后左旋 ⑤

- LR 型, 失-2, 左孩子-1
- 左旋左孩子, 然后右旋 ④
- RL 型, 失-2, 右孩子-1
- 右旋右孩子, 然后左旋 ⑤

- 左旋右孩子, 然后左旋 ④
- 右旋左孩子, 然后右旋 ⑤

- 右旋右孩子, 然后左旋 ⑤

- 左旋左孩子, 然后右旋 ④

- 右旋左孩子, 然后右旋 ④

- 左旋右孩子, 然后左旋 ⑤

- 右旋右孩子, 然后左旋 ⑤

- 左旋左孩子, 然后右旋 ④

- 右旋右孩子, 然后左旋 ⑤

- 左旋右孩子, 然后左旋 ⑤

- 右旋左孩子, 然后右旋 ④

- 左旋左孩子, 然后右旋 ④

- 右旋右孩子, 然后左旋 ⑤

- 左旋右孩子, 然后左旋 ⑤

- 右旋左孩子, 然后右旋 ④

- 左旋左孩子, 然后右旋 ④

- 右旋右孩子, 然后左旋 ⑤

- 左旋右孩子, 然后左旋 ⑤

- 右旋左孩子, 然后右旋 ④

- 左旋左孩子, 然后右旋 ④

- 右旋右孩子, 然后左旋 ⑤

- 左旋右孩子, 然后左旋 ⑤

- 右旋左孩子, 然后右旋 ④

- 左旋左孩子, 然后右旋 ④

- 右旋右孩子, 然后左旋 ⑤

- 左旋右孩子, 然后左旋 ⑤

- 右旋左孩子, 然后右旋 ④

- 左旋左孩子, 然后右旋 ④

- 右旋右孩子, 然后左旋 ⑤

- 左旋右孩子, 然后左旋 ⑤

- 右旋左孩子, 然后右旋 ④

- 左旋左孩子, 然后右旋 ④

- 右旋右孩子, 然后左旋 ⑤

- 左旋右孩子, 然后左旋 ⑤

- 右旋左孩子, 然后右旋 ④

- 左旋左孩子, 然后右旋 ④

- 右旋右孩子, 然后左旋 ⑤

- 左旋右孩子, 然后左旋 ⑤

- 右旋左孩子, 然后右旋 ④

- 左旋左孩子, 然后右旋 ④

- 右旋右孩子, 然后左旋 ⑤

- 左旋右孩子, 然后左旋 ⑤

- 右旋左孩子, 然后右旋 ④

- 左旋左孩子, 然后右旋 ④

- 右旋右孩子, 然后左旋 ⑤

- 左旋右孩子, 然后左旋 ⑤

- 右旋左孩子, 然后右旋 ④

- 左旋左孩子, 然后右旋 ④

- 右旋右孩子, 然后左旋 ⑤

- 左旋右孩子, 然后左旋 ⑤

- 右旋左孩子, 然后右旋 ④

- 左旋左孩子, 然后右旋 ④

- 右旋右孩子, 然后左旋 ⑤

- 左旋右孩子, 然后左旋 ⑤

- 右旋左孩子, 然后右旋 ④

- 左旋左孩子, 然后右旋 ④

- 右旋右孩子, 然后左旋 ⑤

- 左旋右孩子, 然后左旋 ⑤

- 右旋左孩子, 然后右旋 ④

- 左旋左孩子, 然后右旋 ④

- 右旋右孩子, 然后左旋 ⑤

- 左旋右孩子, 然后左旋 ⑤

- 右旋左孩子, 然后右旋 ④

- 左旋左孩子, 然后右旋 ④

- 右旋右孩子, 然后左旋 ⑤

- 左旋右孩子, 然后左旋 ⑤

- 右旋左孩子, 然后右旋 ④

- 左旋左孩子, 然后右旋 ④

- 右旋右孩子, 然后左旋 ⑤

- 左旋右孩子, 然后左旋 ⑤

- 右旋左孩子, 然后右旋 ④

- 左旋左孩子, 然后右旋 ④

- 右旋右孩子, 然后左旋 ⑤

- 左旋右孩子, 然后左旋 ⑤

- 右旋左孩子, 然后右旋 ④

- 左旋左孩子, 然后右旋 ④

- 右旋右孩子, 然后左旋 ⑤

- 左旋右孩子, 然后左旋 ⑤

- 右旋左孩子, 然后右旋 ④

- 左旋左孩子, 然后右旋 ④

- 右旋右孩子, 然后左旋 ⑤

- 左旋右孩子, 然后左旋 ⑤

- 右旋左孩子, 然后右旋 ④

- 左旋左孩子, 然后右旋 ④

- 右旋右孩子, 然后左旋 ⑤

- 左旋右孩子, 然后左旋 ⑤

- 右旋左孩子, 然后右旋 ④

- 左旋左孩子, 然后右旋 ④

- 右旋右孩子, 然后左旋 ⑤

- 左旋右孩子, 然后左旋 ⑤

- 右旋左孩子, 然后右旋 ④

- 左旋左孩子, 然后右旋 ④

- 右旋右孩子, 然后左旋 ⑤

- 左旋右孩子, 然后左旋 ⑤

- 右旋左孩子, 然后右旋 ④

- 左旋左孩子, 然后右旋 ④

- 右旋右孩子, 然后左旋 ⑤

- 左旋右孩子, 然后左旋 ⑤

- 右旋左孩子, 然后右旋 ④

- 左旋左孩子, 然后