進階資料結構

cthbst

2020/08/04

FB 粉專: 演算法海牛

競賽經歷

- 2019 ACM-ICPC World Finals 第 21 名
- 2018 交大 PCCA 社長
- 2015, 2016 TOI 2!
- 2015 APIO 銀牌

資料結構的角色

資料結構的角色

設計手法

D&C, DP, Greedy, Probabilistic Methods ...

演算法領域

Graph, Geometry, String, Game, ...

資料結構

Array, List, Tree, Hash Table, ...

LIS 問題

Input:

一個長度為 n 的正整數序列 $a = [a_1, a_2, a_3, \ldots, a_n]$

Output:

找出最長的嚴格遞增子序列

LIS 問題

DP 轉移式

$dp(i) =$ 以 a_i 為結尾的 LIS 長度 (一定要選 a_i)

$$dp(i) = \max_{\substack{j<i \\ a_j<a_i}} \{\mathrm{dp}(j) + 1\}$$

$O(n^2)$?

LIS 問題

DP 轉移式

$dp(i) =$ 以 a_i 為結尾的 LIS 長度 (一定要選 a_i)

$$dp(i) = \max_{\substack{j<i \\ a_j<a_i}} \{\mathrm{dp}(j) + 1\}$$

使用線段樹 or BIT

$O(n \log n)$

帶權 LIS 問題

Input:

一個長度為 n 的正整數序列 $a = [a_1, a_2, a_3, \ldots, a_n]$

一個長度為 n 的正整數序列 $\mathrm{w} = [w_1, w_2, w_3, \ldots, w_n]$

Output:

找出 w 總和最大的子序列，滿足在 a 中是嚴格遞增的

使用高級工具之前

區間 XOR 查詢

Input:

一個長度為 n 的正整數序列 $a = [a_1, a_2, a_3, \ldots, a_n]$

Query(L, R):

回傳 $a_L \oplus a_{L+1} \oplus \ldots \oplus a_R$，其中 $\oplus$ 表示 XOR 運算

區間 XOR 查詢

Input:

一個長度為 n 的正整數序列 $a = [a_1, a_2, a_3, \ldots, a_n]$

Query(L, R):

回傳 $a_L \oplus a_{L+1} \oplus \ldots \oplus a_R$，其中 $\oplus$ 表示 XOR 運算

線段樹 （X）

$< O(n), O(\log n) >$

區間 XOR 查詢

Input:

一個長度為 n 的正整數序列 $a = [a_1, a_2, a_3, \ldots, a_n]$

Query(L, R):

回傳 $a_L \oplus a_{L+1} \oplus \ldots \oplus a_R$，其中 $\oplus$ 表示 XOR 運算

前綴和（O）

$< O(n), O(\log n) >$

區間 XOR 查詢

Input:

一個長度為 n 的正整數序列 $a = [a_1, a_2, a_3, \ldots, a_n]$

Query(L, R):

回傳 $a_L \oplus a_{L+1} \oplus \ldots \oplus a_R$，其中 $\oplus$ 表示 XOR 運算

常見症狀：線段樹中毒

2020 YTP 決賽 Problem 9 簡化

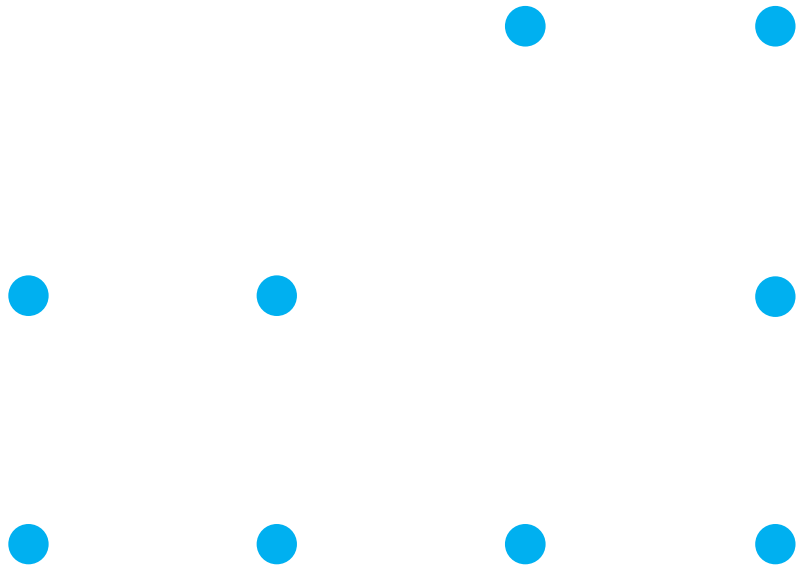
Input:

二維平面上給 n 個點座標

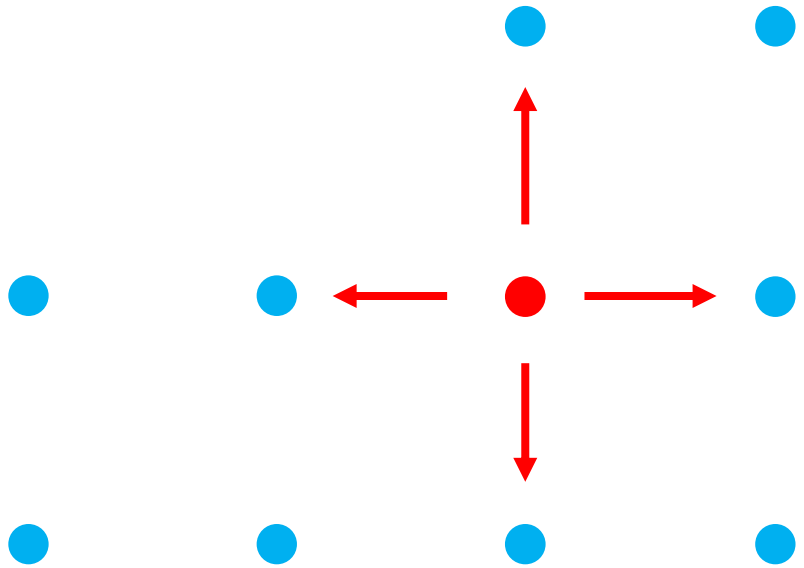
Output:

平面上存在一個點，往 +x, -x, +y, -y 方向的射線都會撞到 Input 中給的點，就輸出 “**No**”，否則輸出“**Yes**”。

2020 YTP 決賽 Problem 9 簡化

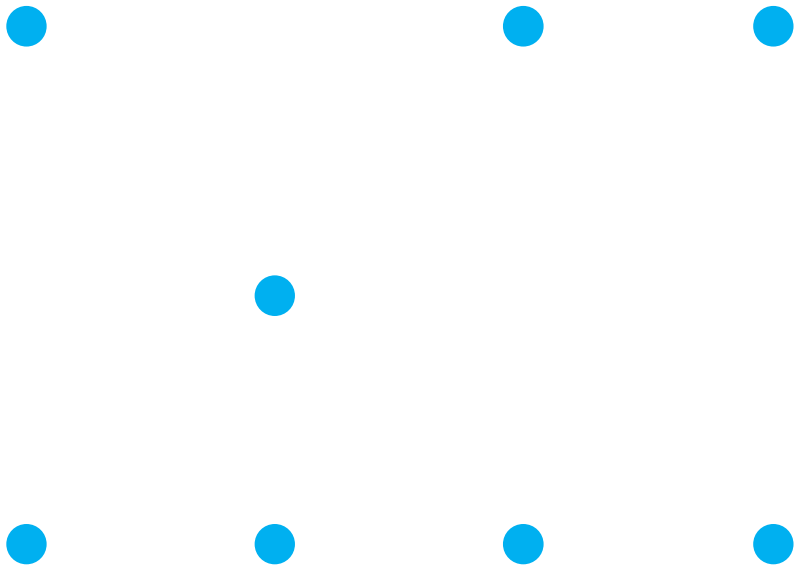


2020 YTP 決賽 Problem 9 簡化



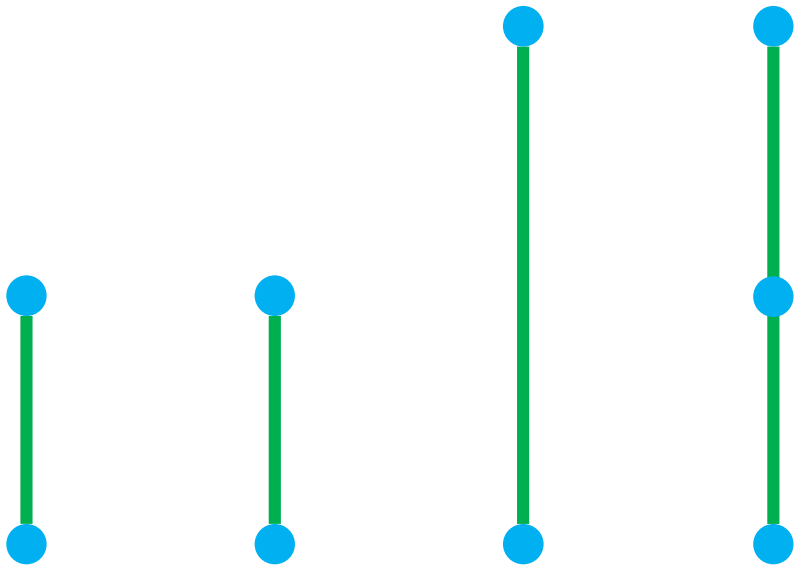
輸出 “**No**”

2020 YTP 決賽 Problem 9 簡化

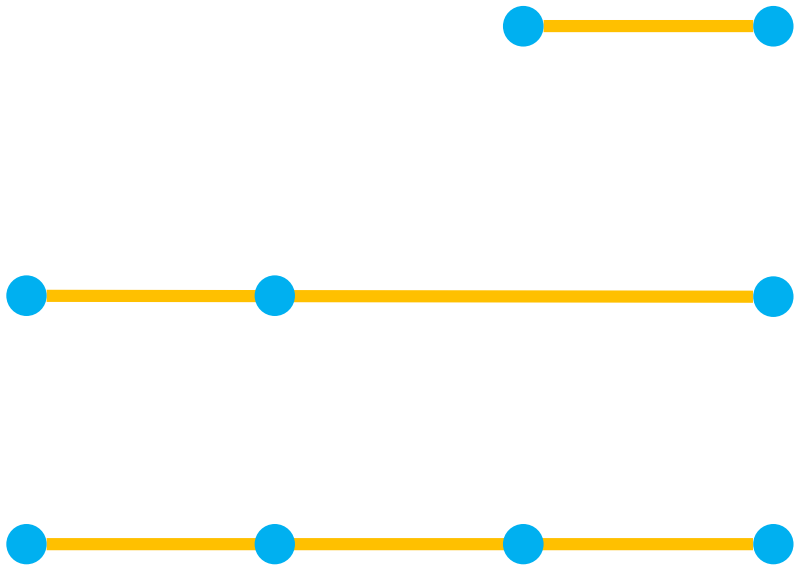


輸出 “Yes”

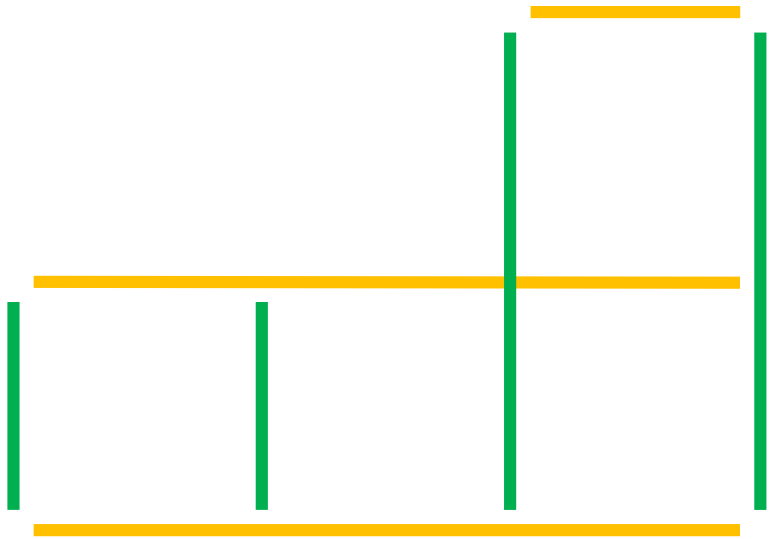
2020 YTP 決賽 Problem 9 簡化



2020 YTP 決賽 Problem 9 簡化

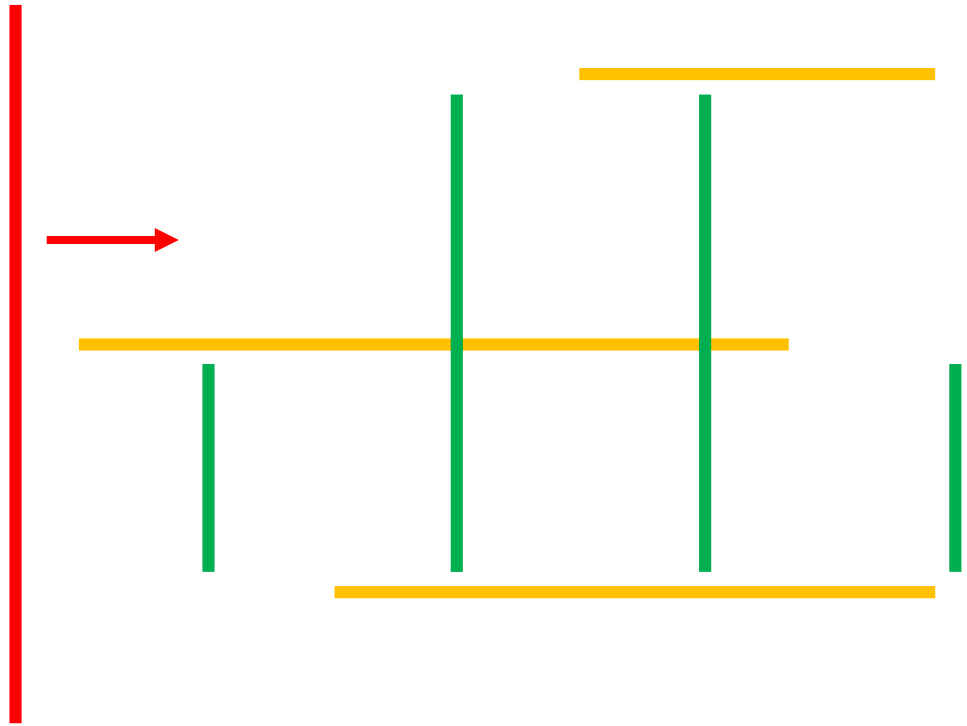


2020 YTP 決賽 Problem 9 簡化

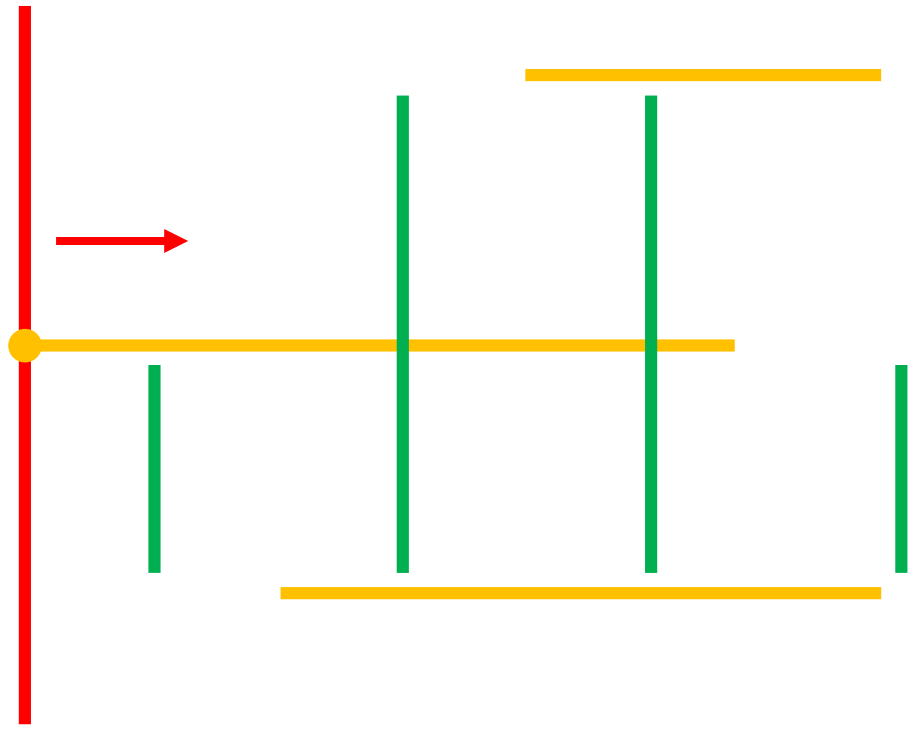


轉為線段相交問題

掃描線技巧

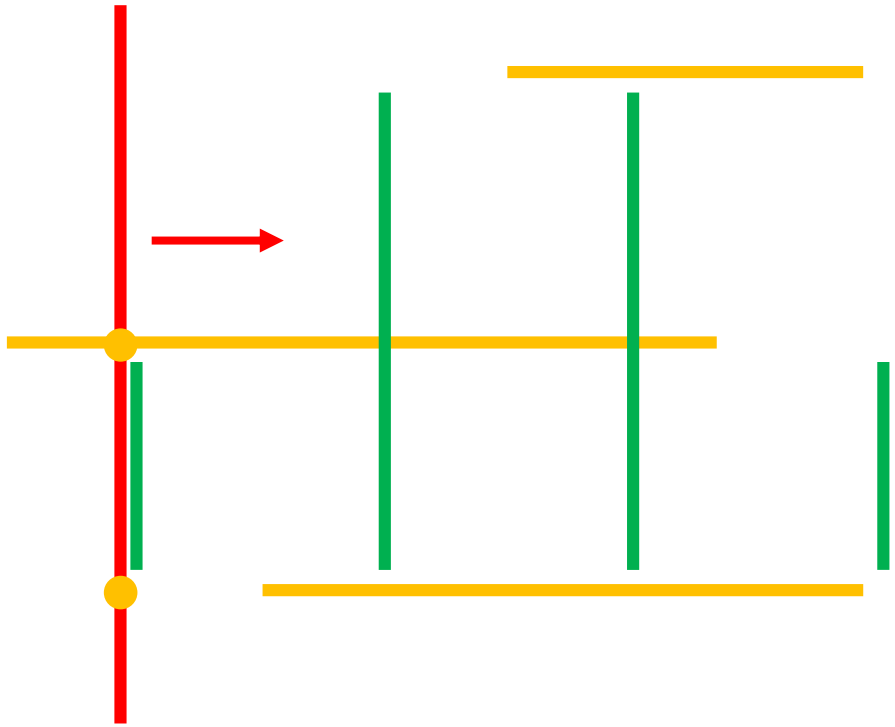


掃描線技巧



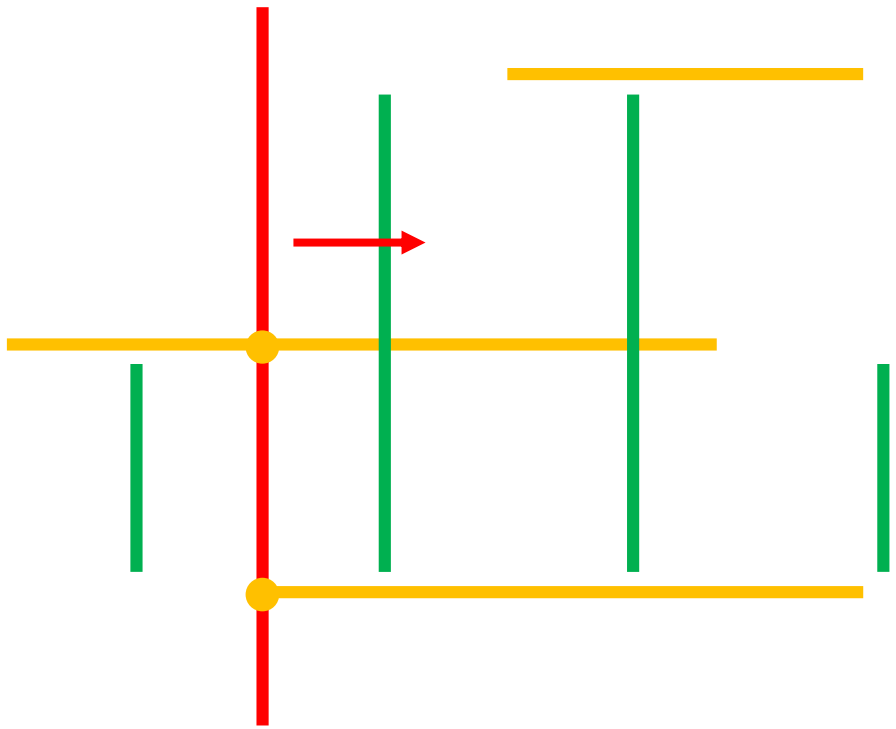
insert

掃描線技巧



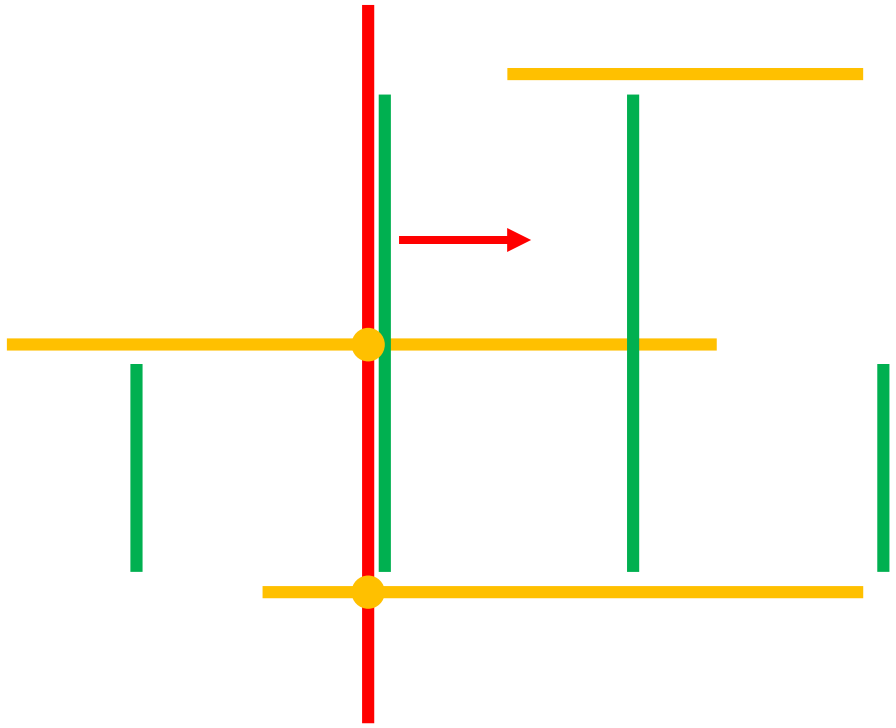
query

掃描線技巧



insert

掃描線技巧



query

問題

Update(x):

在集合中新增/刪除數字 x

Query(L, R):

判斷集合中是否有一個數字介於 (L, R)

線段樹

$< O(\log n), O(\log n) >$

問題

Update(x):

在集合中新增/刪除數字 x

Query(L, R):

判斷集合中是否有一個數字介於 (L, R)

`std::set`

$< O(\log n), O(\log n) >$

問題

Update(x):

在集合中新增/刪除數字 x

Query(L, R):

判斷集合中是否有一個數字介於 (L, R)

常見症狀：輕度毒瘤

Range Minimum Query

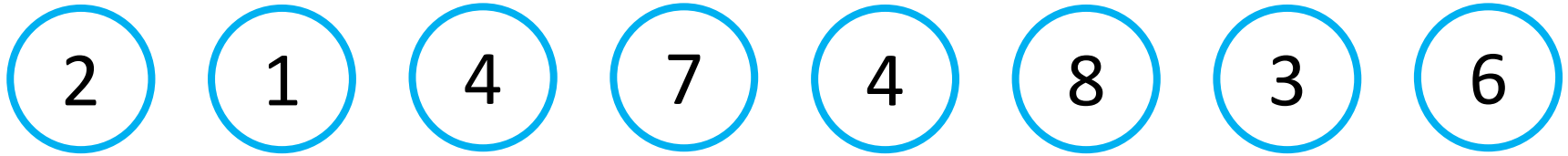
Range Minimum Query (RMQ)

Input:

一個長度為 n 的整數序列 $a = [a_1, a_2, a_3, \ldots, a_n]$

Query(L, R):

回傳 $\min\{a_L, a_{L+1}, \ldots, a_R\}$



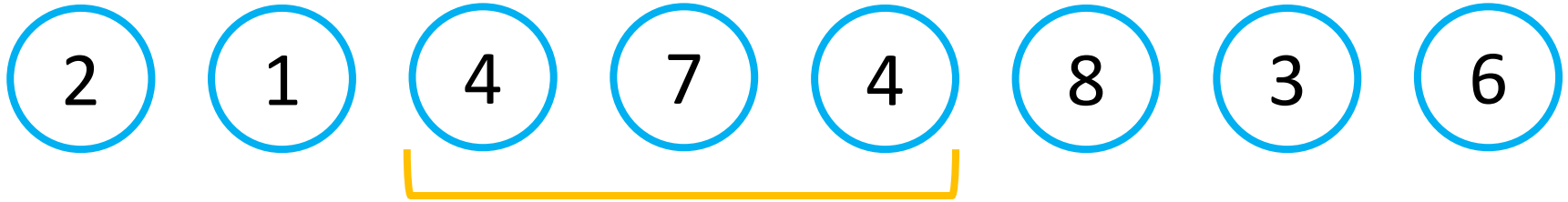
Range Minimum Query (RMQ)

Input:

一個長度為 n 的整數序列 $a = [a_1, a_2, a_3, \dots, a_n]$

Query(L, R):

回傳 $\min\{a_L, a_{L+1}, \dots, a_R\}$



$\min\{4,7,4\} = 4$

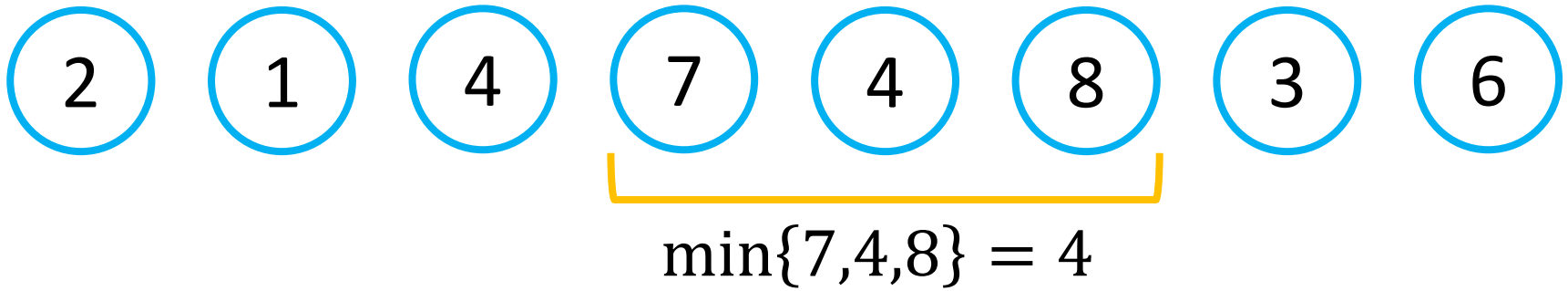
Range Minimum Query (RMQ)

Input:

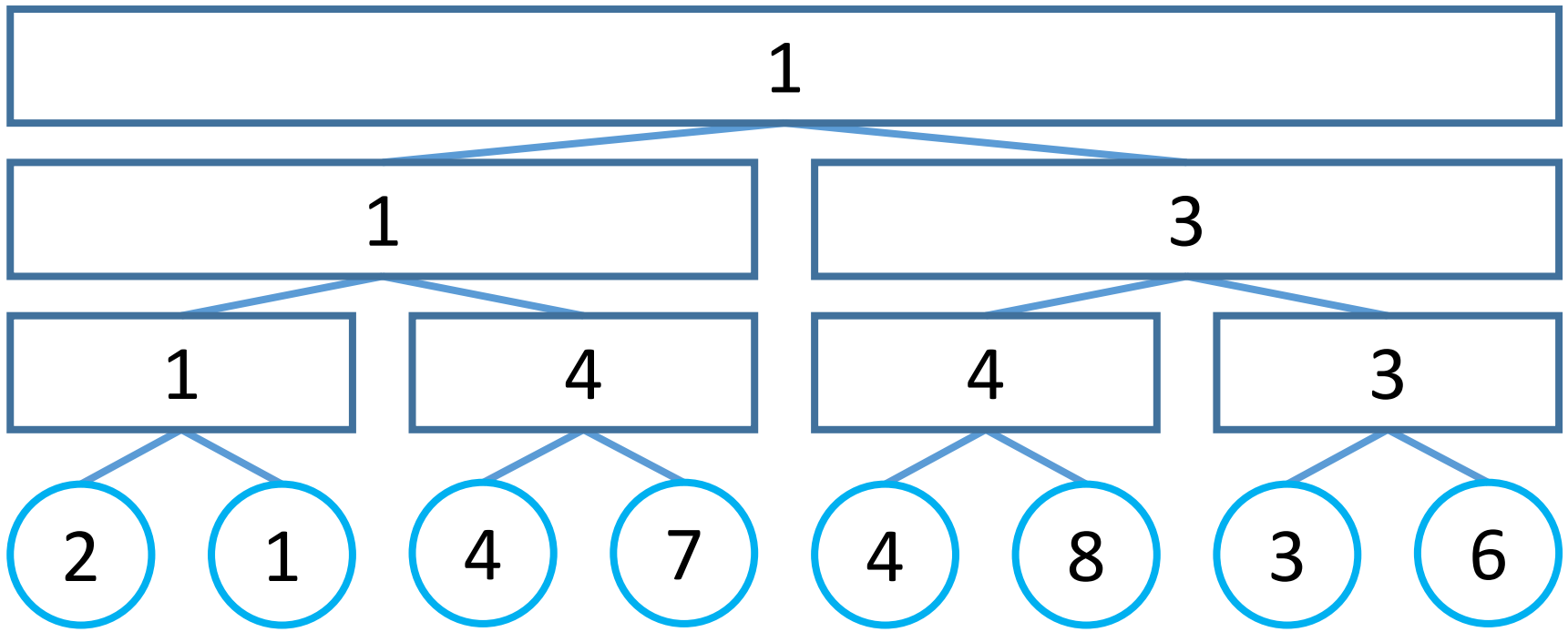
一個長度為 n 的整數序列 $a = [a_1, a_2, a_3, \ldots, a_n]$

Query(L, R):

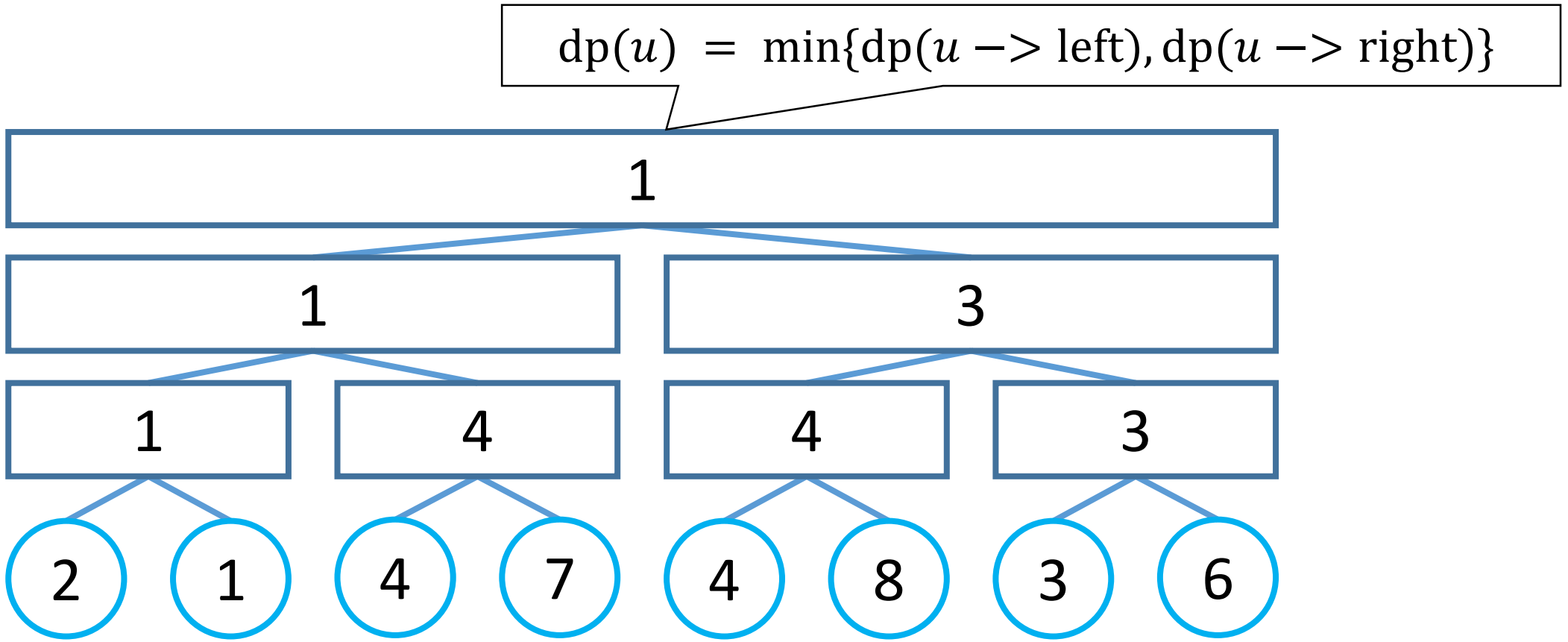
回傳 $\min\{a_L, a_{L+1}, \ldots, a_R\}$



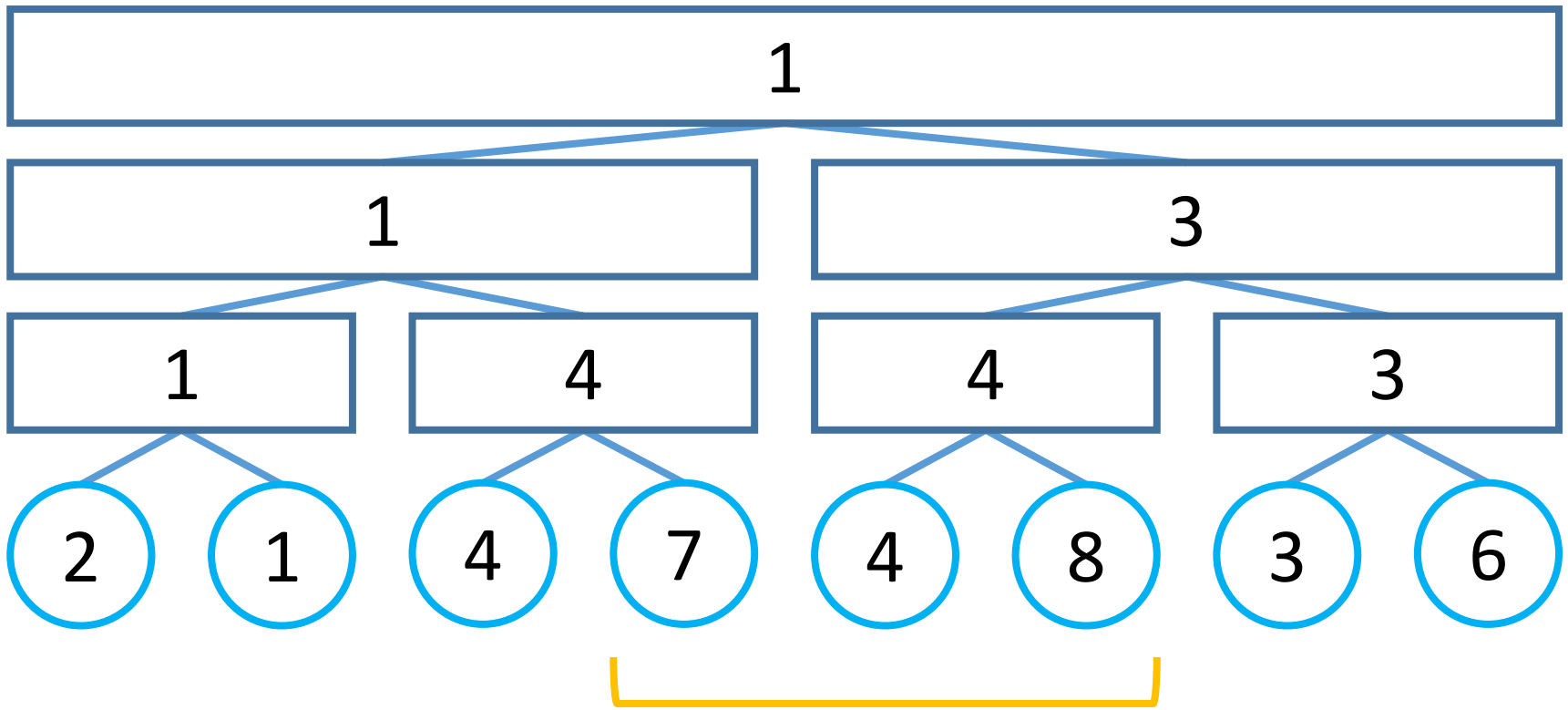
Segment Tree $< O(n), O(\log n) >$



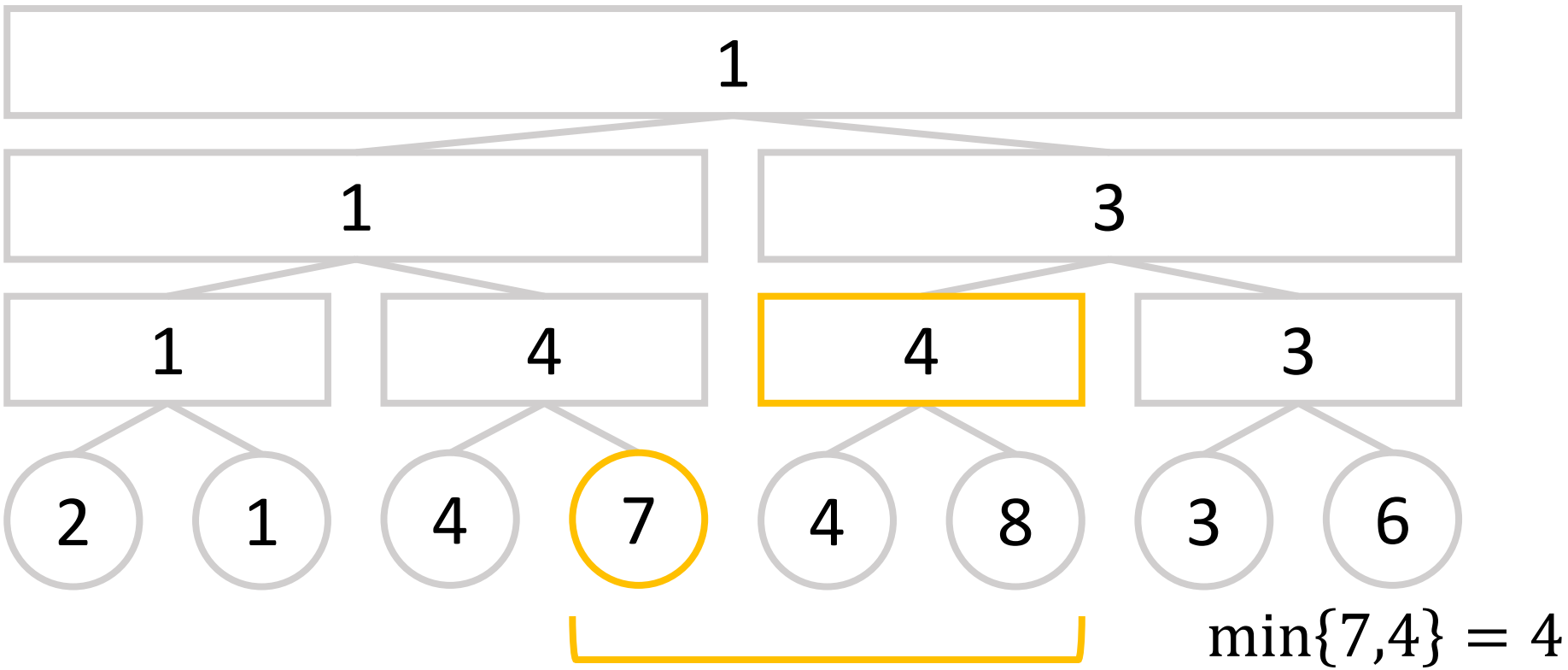
Segment Tree $< O(n), O(\log n) >$



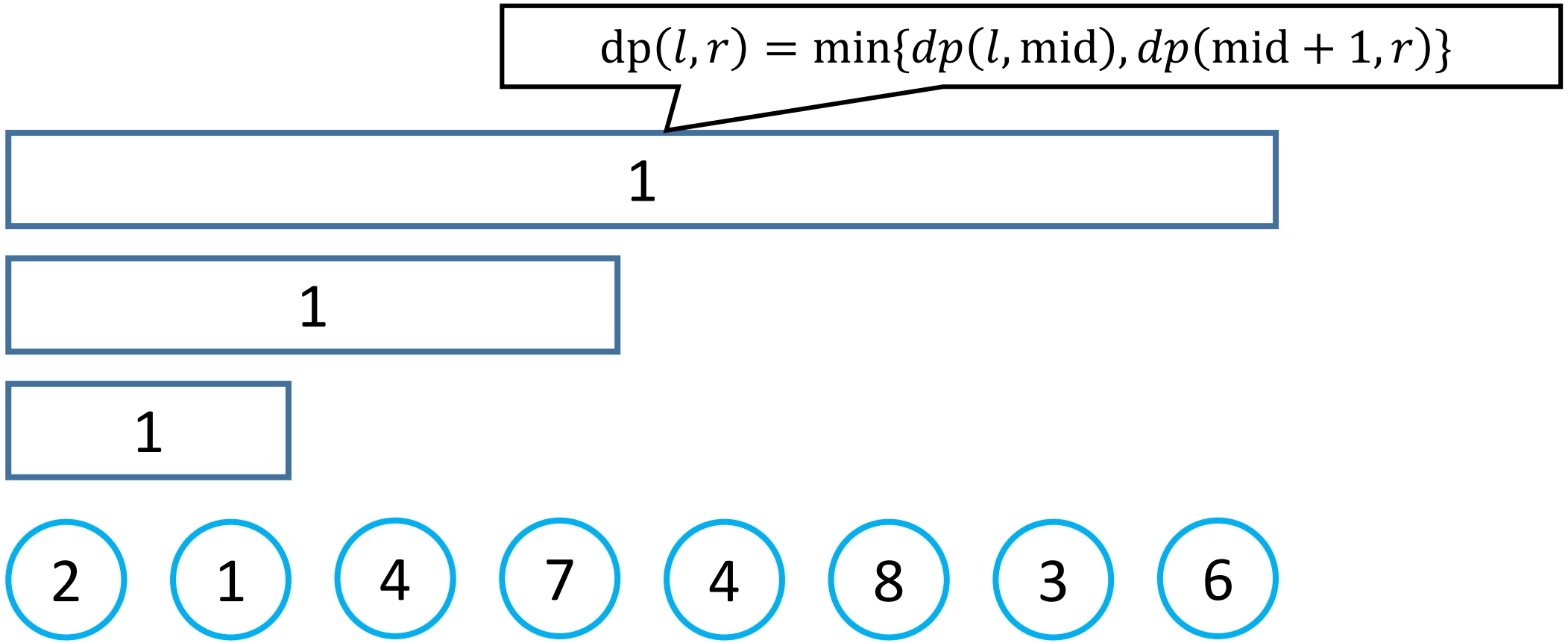
Segment Tree $< O(n), O(\log n) >$



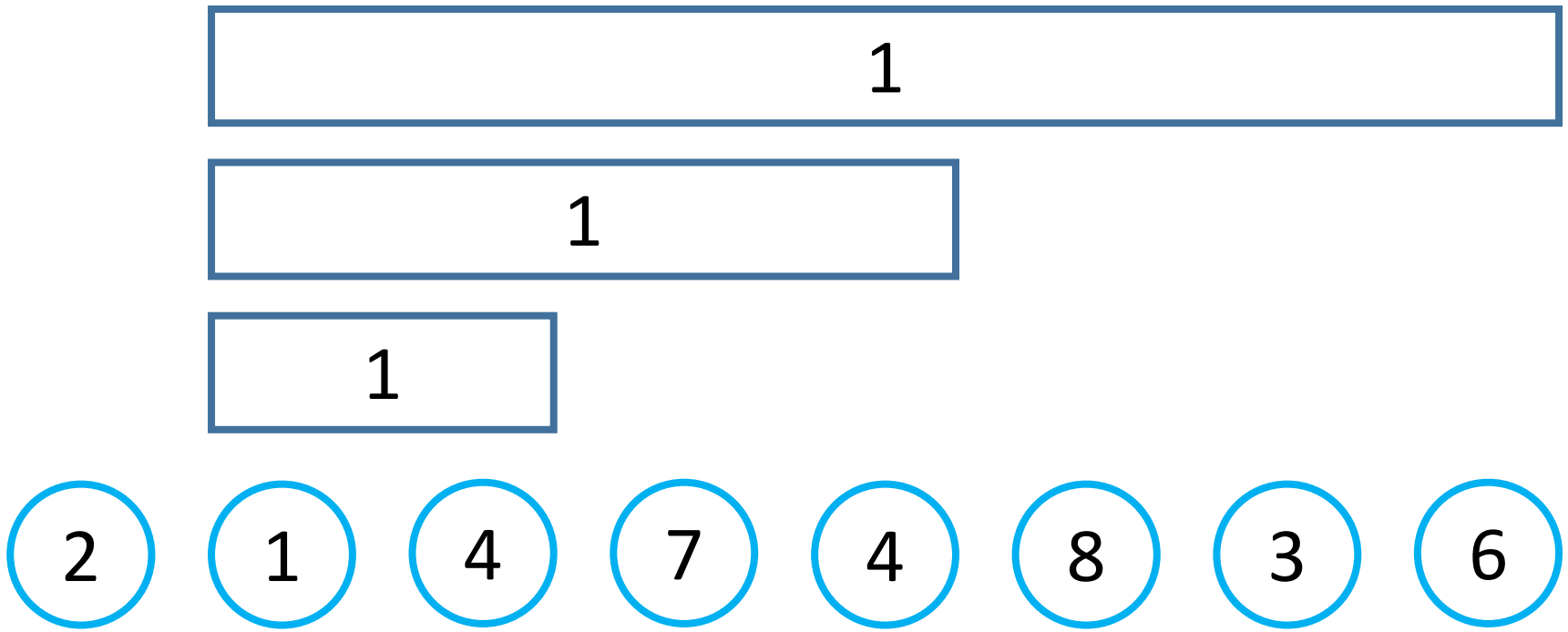
Segment Tree $< O(n), O(\log n) >$



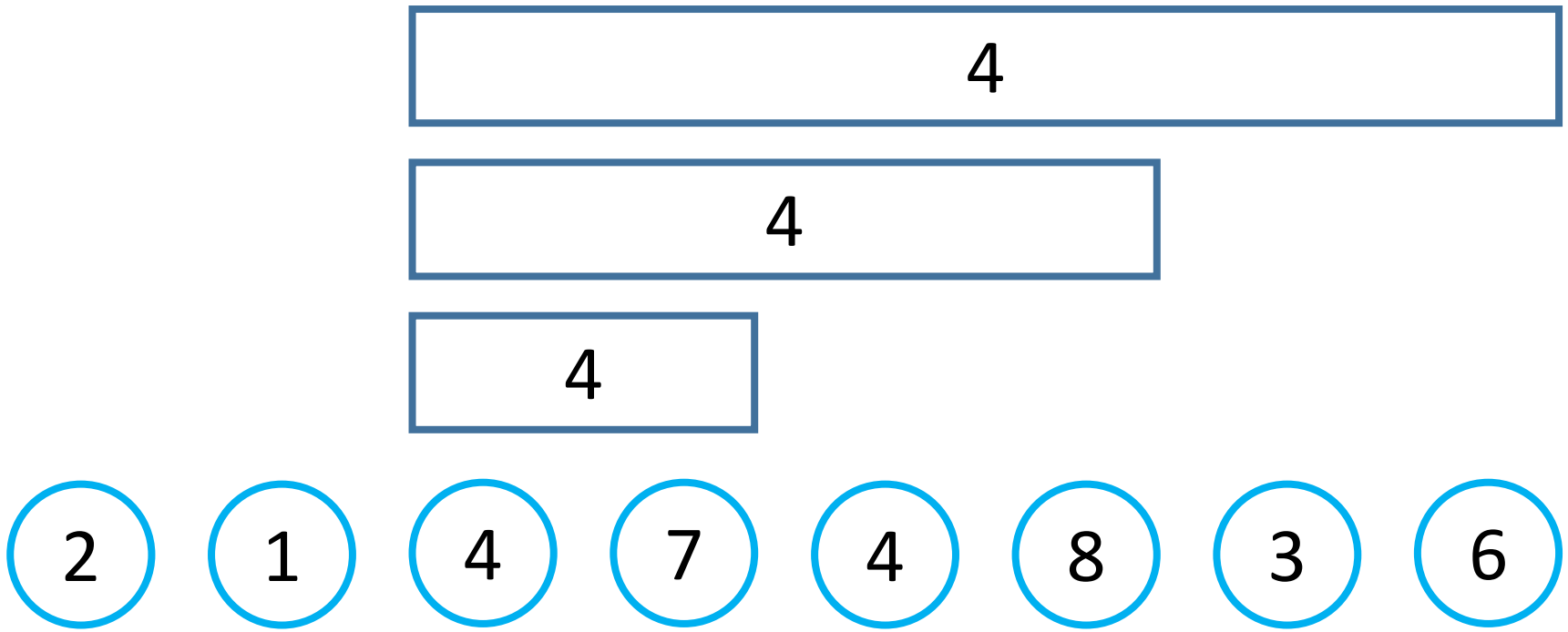
Sparse Table $< O(n \log n), O(1) >$



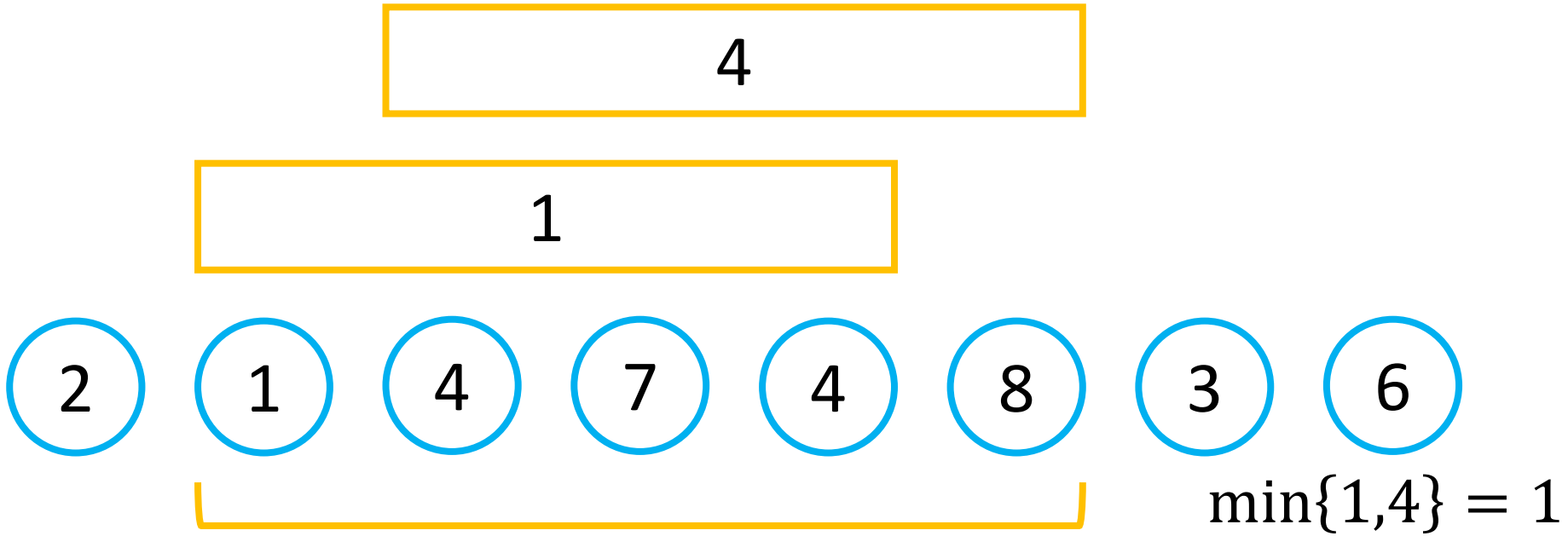
Sparse Table $< O(n \log n), O(1) >$



Sparse Table $< O(n \log n), O(1) >$



Sparse Table $< O(n \log n), O(1) >$

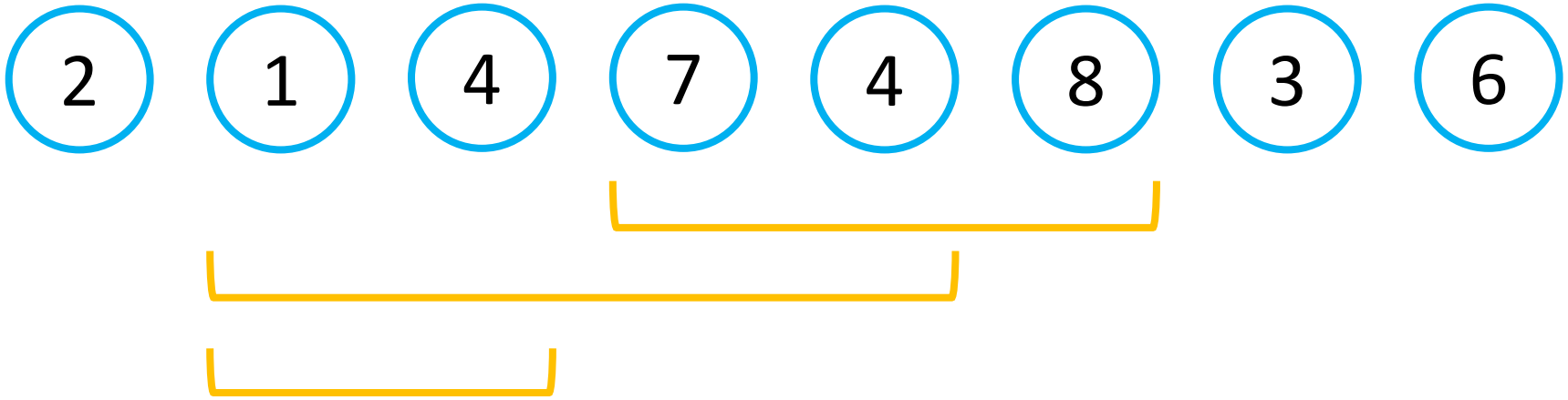


± 1 RMQ $< O(n), O(1) >$

loading...

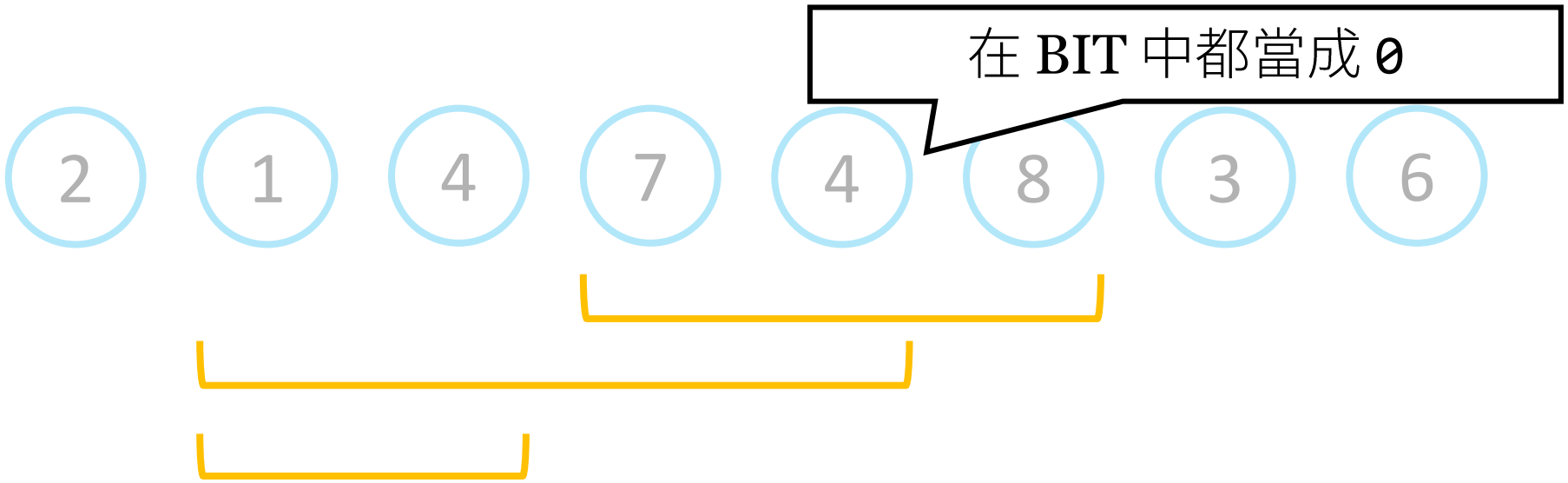
BIT Offline RMQ $< O(n), O(\log n) >$

1. 將所有查詢按照左界遞減排序
2. 準備一個 BIT 一開始全部都是 `0`
3. Query(L_i, R_i) 時，將 $a_{L_i} \sim a_{L_{i-1}}$ 加入 BIT，回傳 PrefixMin(R)



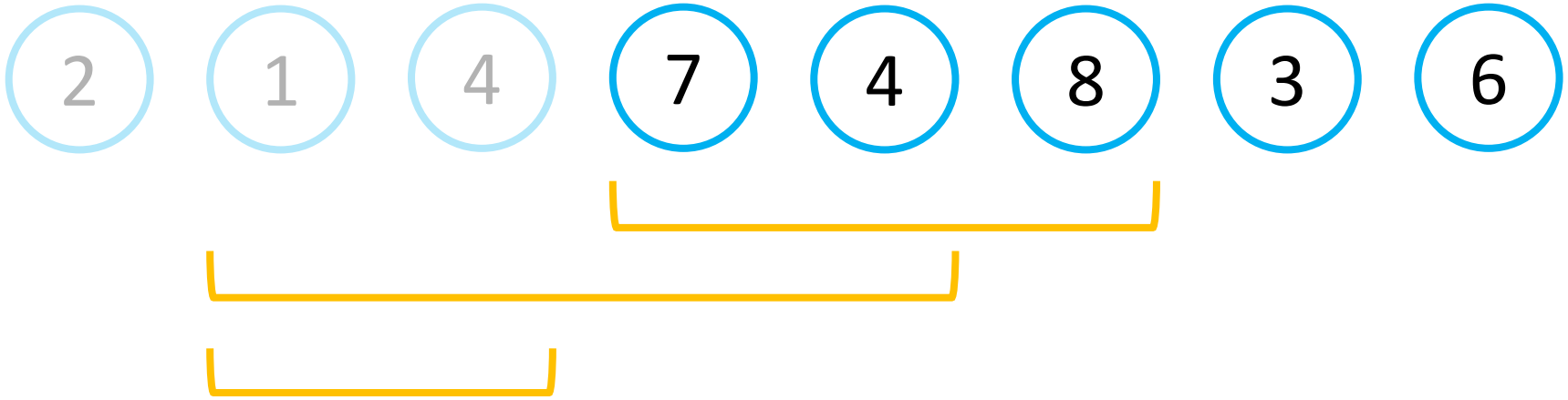
BIT Offline RMQ $< O(n), O(\log n) >$

1. 將所有查詢按照左界遞減排序
2. 準備一個 BIT 一開始全部都是 0
3. Query(L_i, R_i) 時，將 $a_{L_i} \sim a_{L_{i-1}}$ 加入 BIT，回傳 PrefixMin(R)



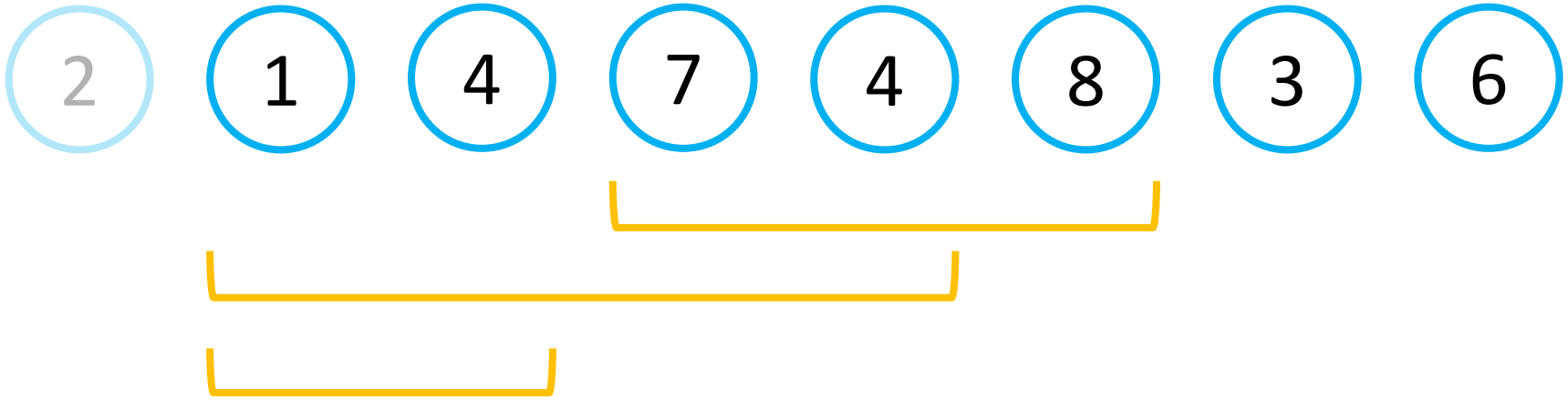
BIT Offline RMQ $< O(n), O(\log n) >$

1. 將所有查詢按照左界遞減排序
2. 準備一個 BIT 一開始全部都是 0
3. Query(L_i, R_i) 時，將 $a_{L_i} \sim a_{L_{i-1}}$ 加入 BIT，回傳 PrefixMin(R)

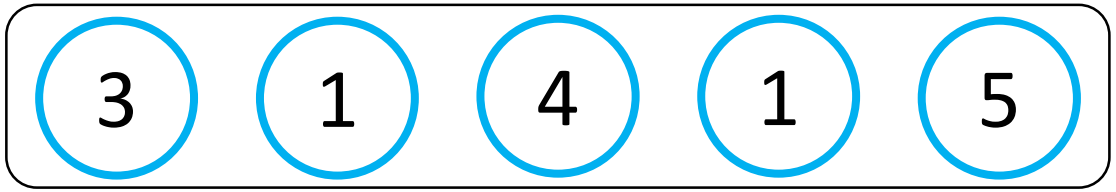


BIT Offline RMQ $< O(n), O(\log n) >$

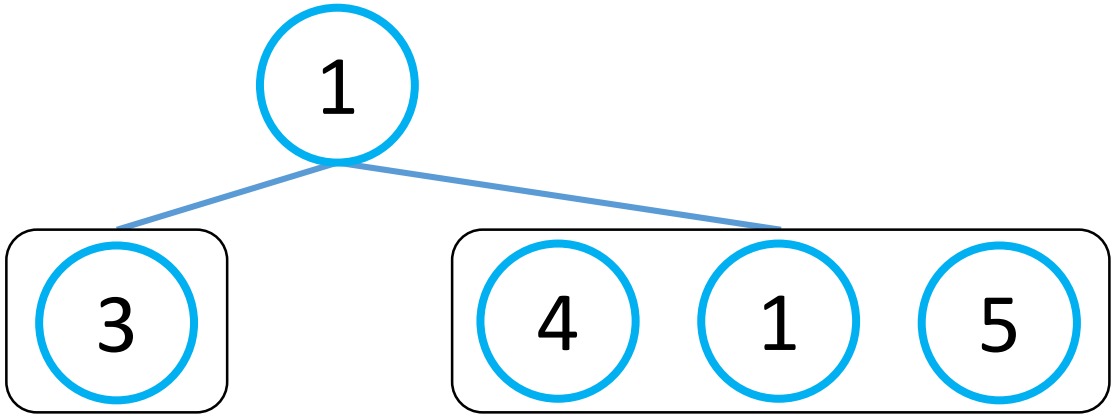
1. 將所有查詢按照左界遞減排序
2. 準備一個 BIT 一開始全部都是 0
3. Query(L_i, R_i) 時，將 $a_{L_i} \sim a_{L_{i-1}}$ 加入 BIT，回傳 PrefixMin(R)



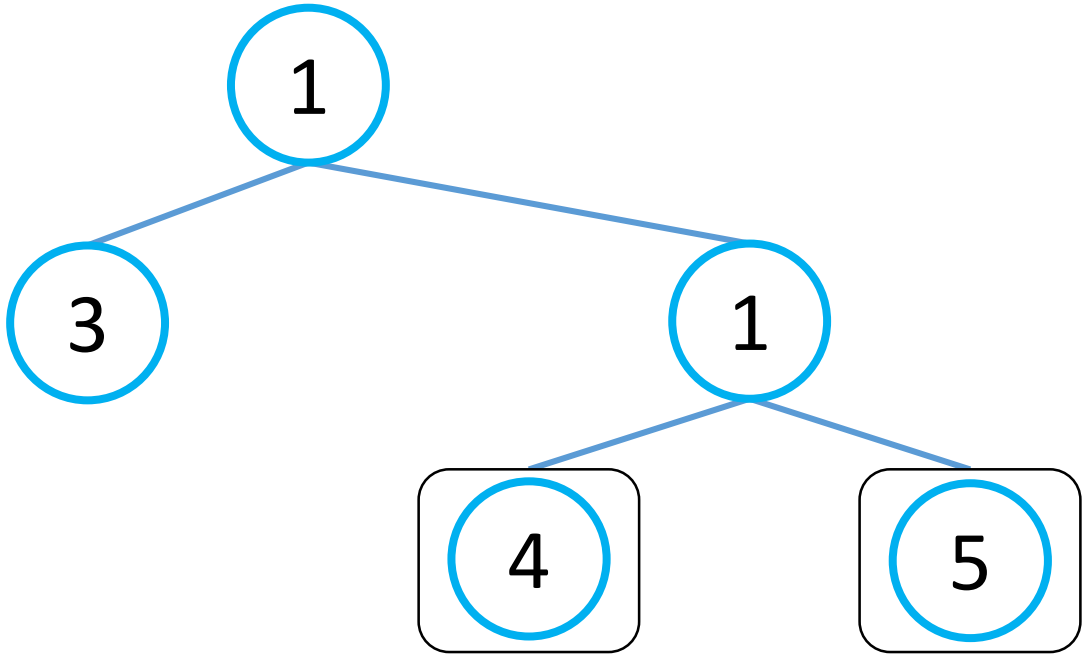
Tarjan's Offline LCA $< O(n), O(1) >$



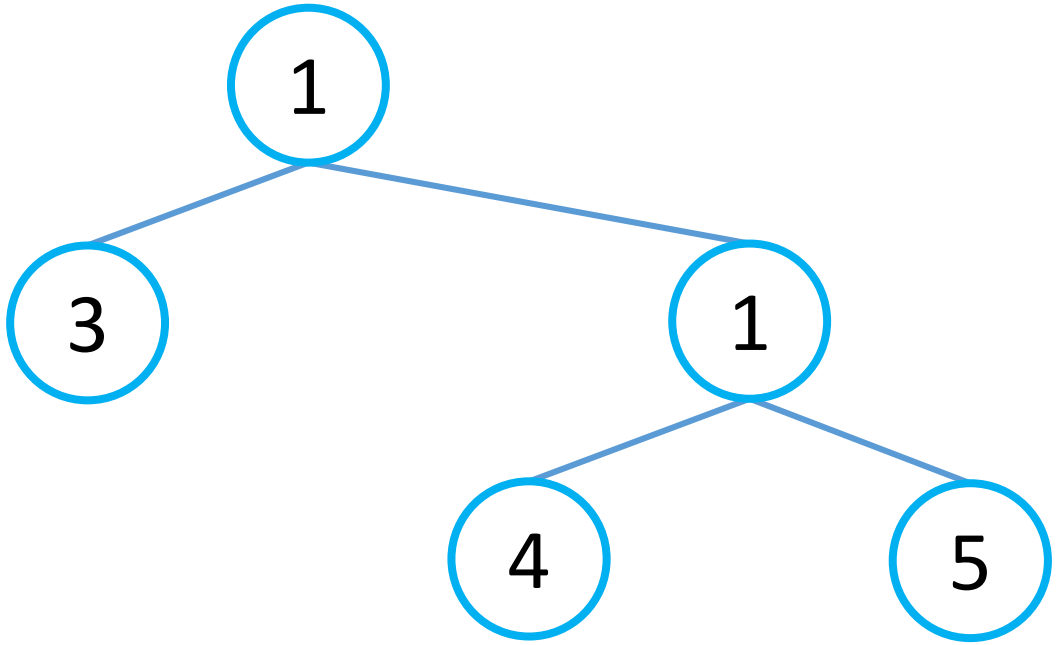
Tarjan's Offline LCA $< O(n), O(1) >$



Tarjan's Offline LCA $< O(n), O(1) >$

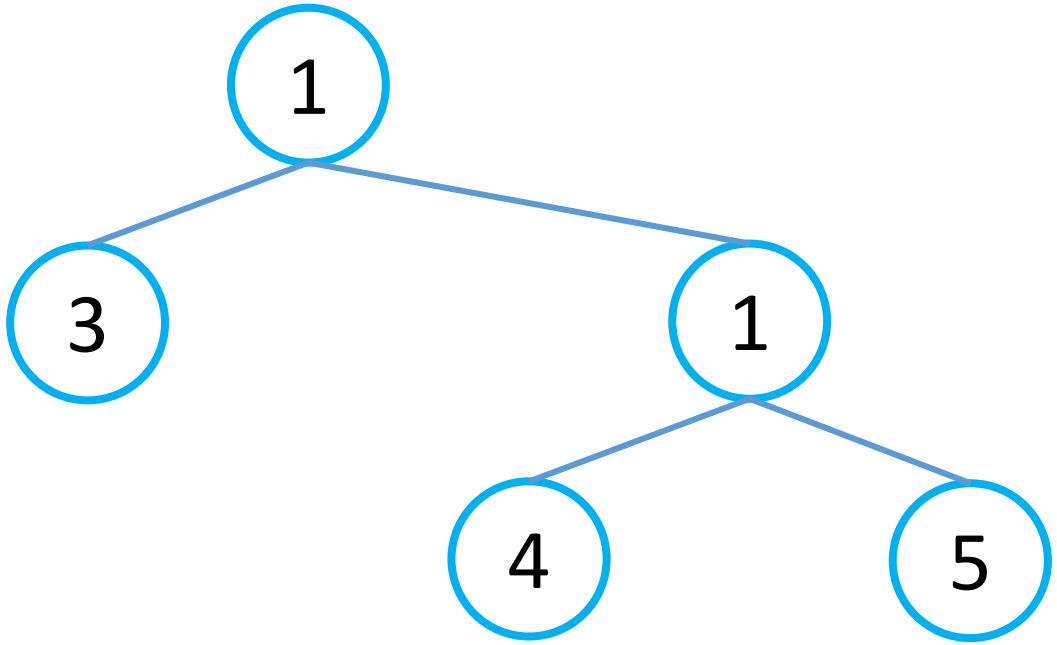


Tarjan's Offline LCA $< O(n), O(1) >$



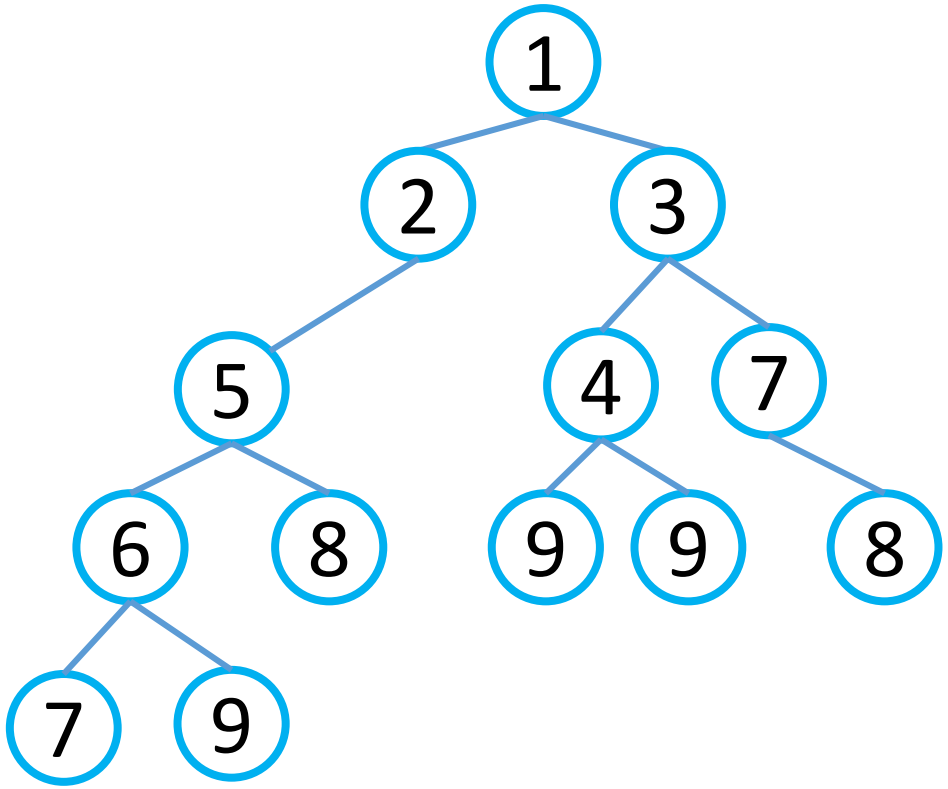
Cartesian tree

Tarjan's Offline LCA $< O(n), O(1) >$

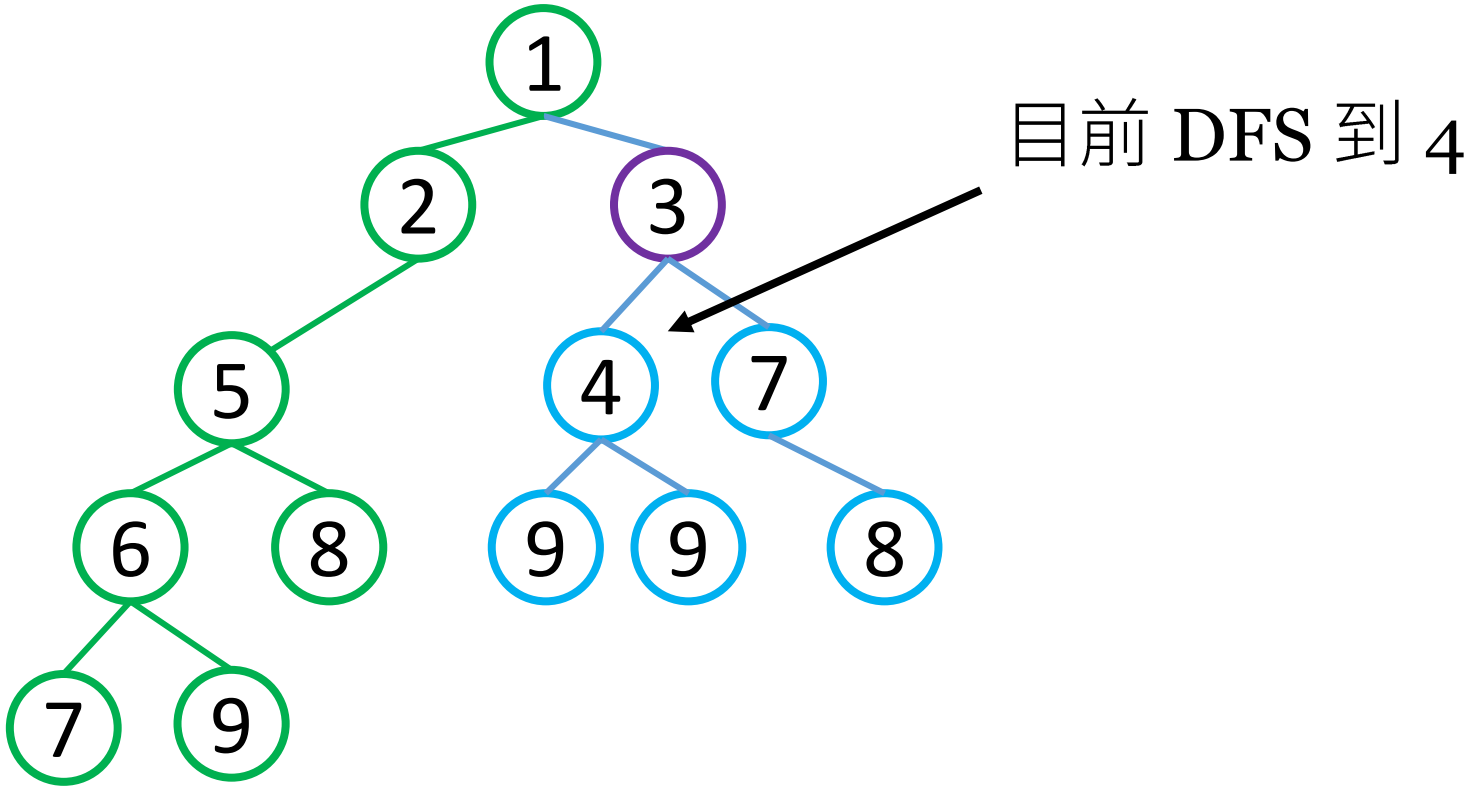


RMQ → LCA

Tarjan's Offline LCA $< O(n), O(1) >$

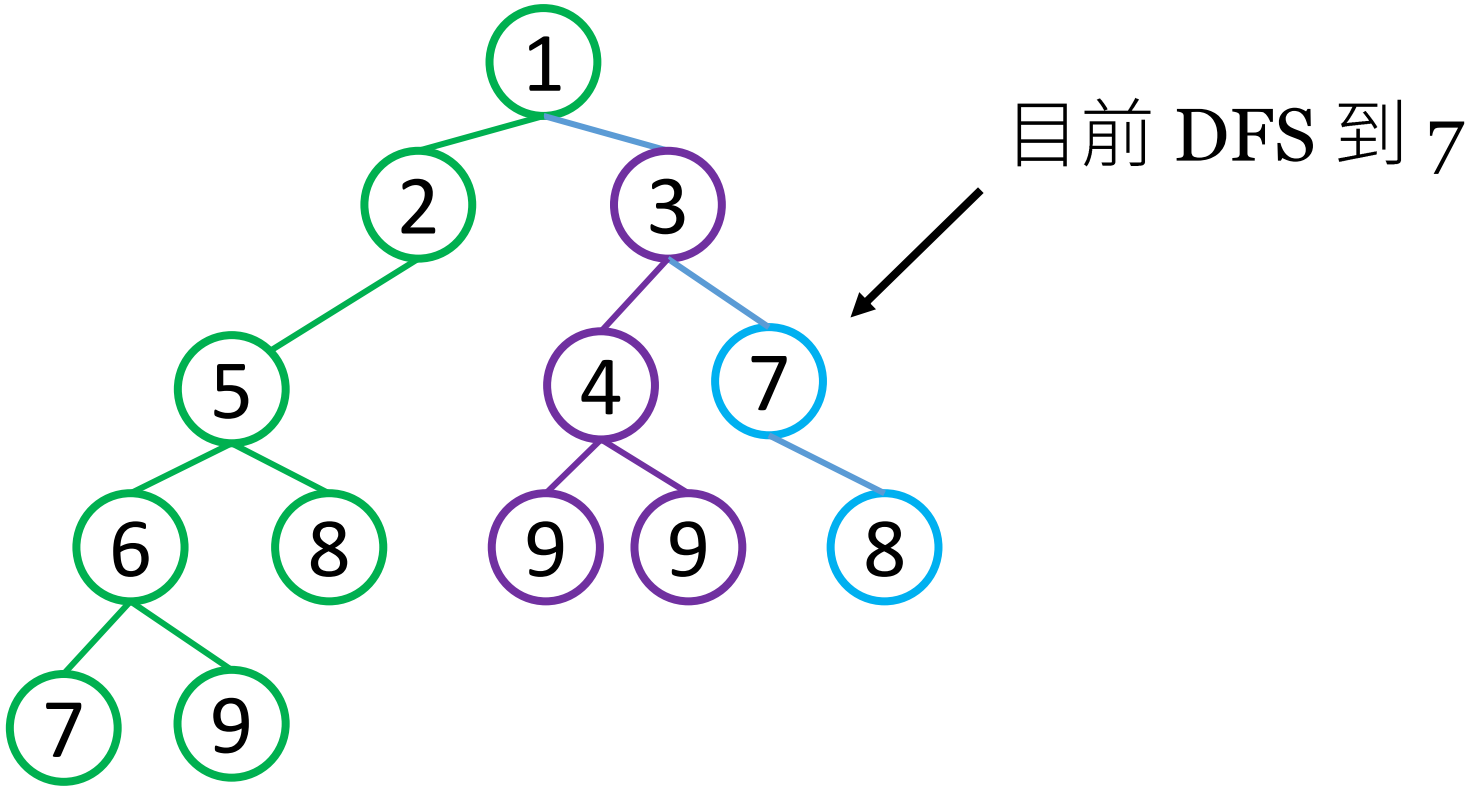


Tarjan's Offline LCA $< O(n), O(1) >$



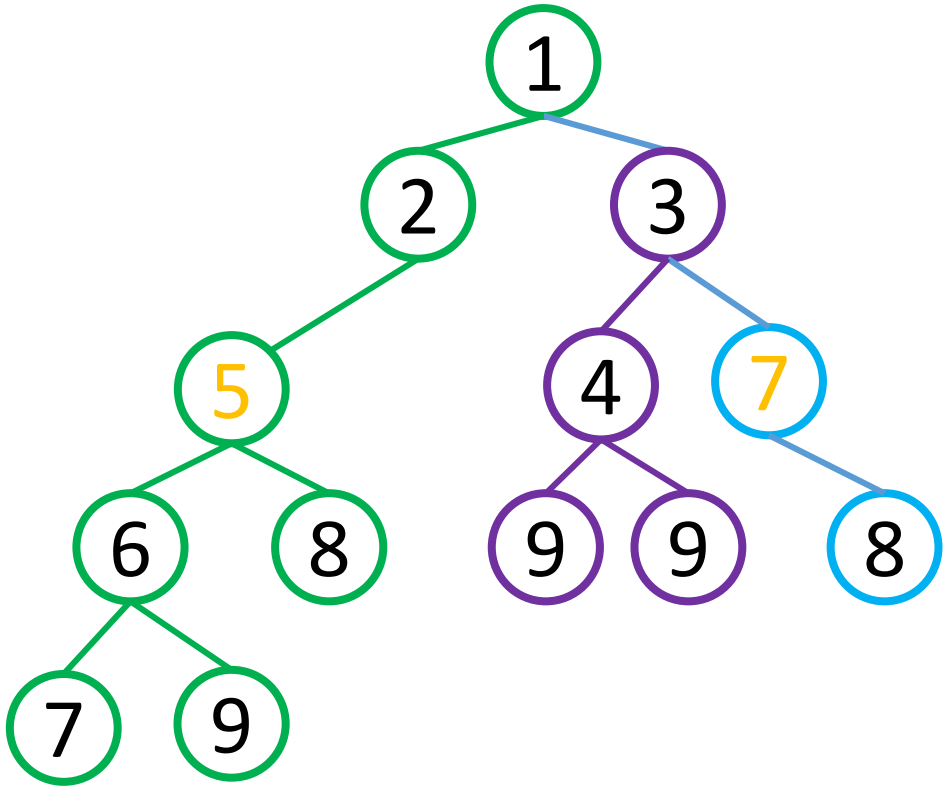
祖先的左子樹，沒有包含自己就全部合併

Tarjan's Offline LCA $< O(n), O(1) >$



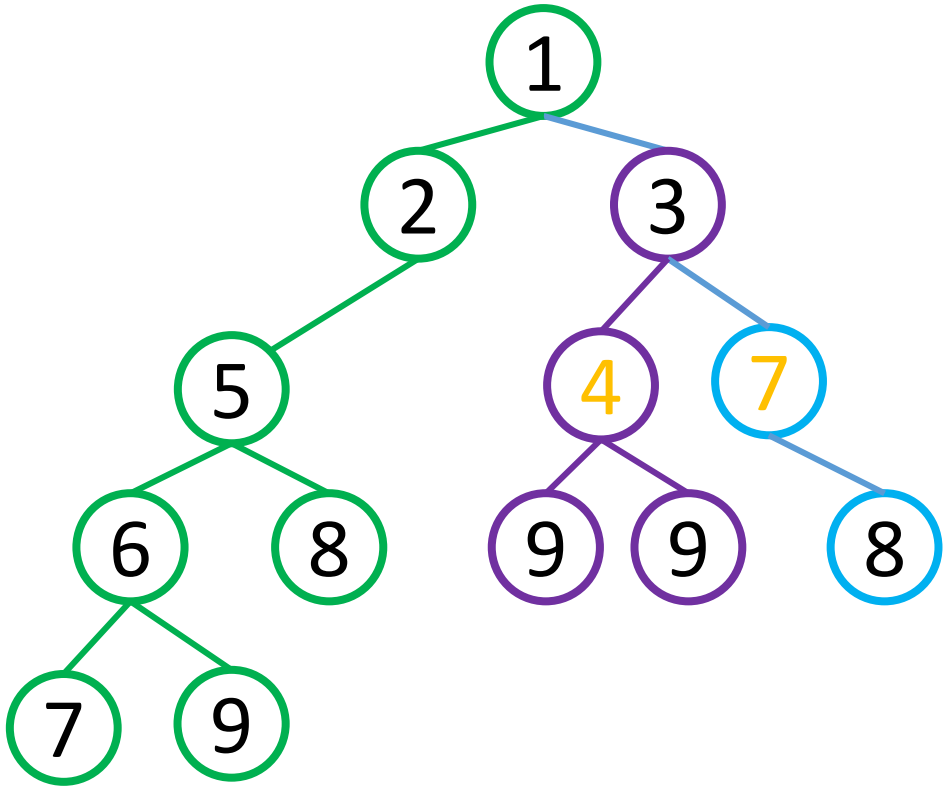
祖先的左子樹，沒有包含自己就全部合併

Tarjan's Offline LCA $< O(n), O(1) >$



LCA(5,7)=1

Tarjan's Offline LCA $< O(n), O(1) >$



LCA(4,7)=3

隨機序列的 Cartesian tree 高度?

uniform 抽一個 1~n 的排列 (note: 1~n 的排列恰好有 n! 種)

最小值落在 $a_{\frac{1}{4}n} \sim a_{\frac{3}{4}n}$ 的機率為 ½

$$h(n) = 1 + \frac{1}{2} \times h\left(\frac{n}{2}\right) + \frac{1}{2} h(n)$$

$$h(n) = 2 + h\left(\frac{n}{2}\right) = O(\log n)$$

線段樹

區間最大連續和問題

Input:

一個長度為 n 的整數序列 $a = [a_1, a_2, a_3, \dots, a_n]$

Query(L, R):

回傳 $\max_{L \le i \le j \le R} \{a_i, a_{i+1}, \dots, a_j\}$

問題

Input:

一個長度為 n 的整數序列 $a = [a_1, a_2, a_3, \ldots, a_n]$

Query(L, R):

回傳 $\max_{L \le i \le j \le k \le R} \{a_i - 2a_j + a_k\}$

Lazy Tag

區間加值，區間求最大值

Input:

一個長度為 n 的整數序列 $a = [a_1, a_2, a_3, \dots, a_n]$

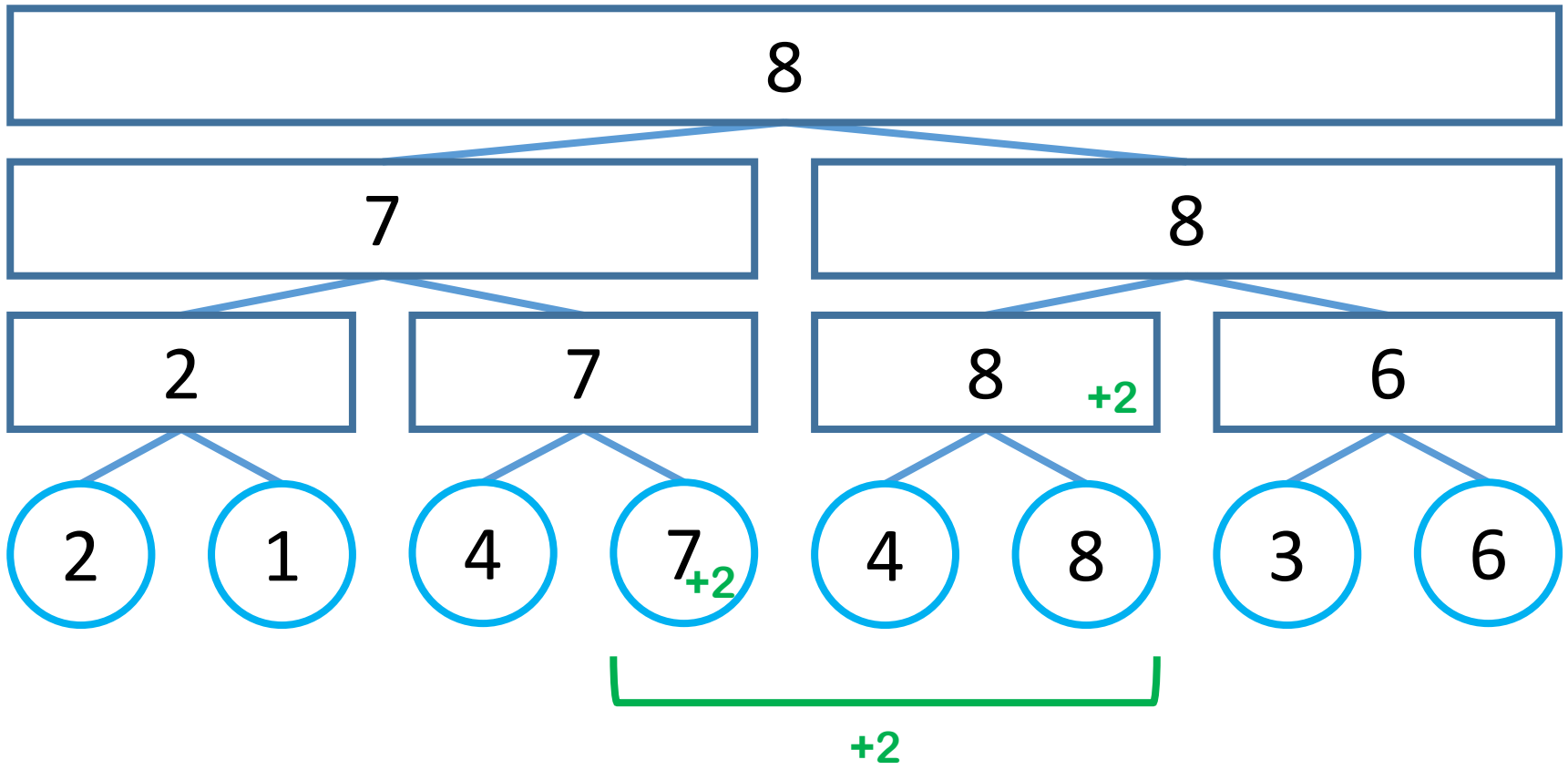
Query(L, R):

回傳 $\max\{a_L, a_{L+1}, \dots, a_R\}$

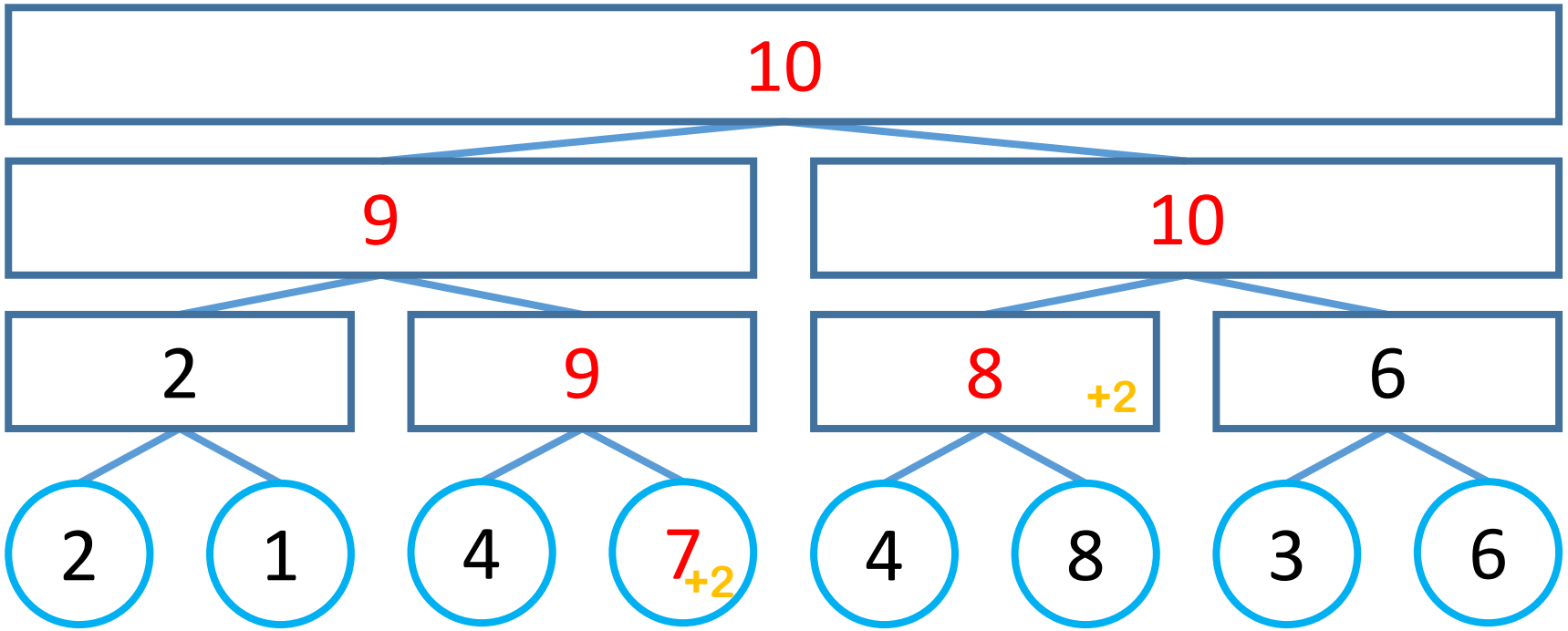
RangeAdd(L, R, x):

將 $a_L, a_{L+1}, \dots, a_R$ 都加上 x

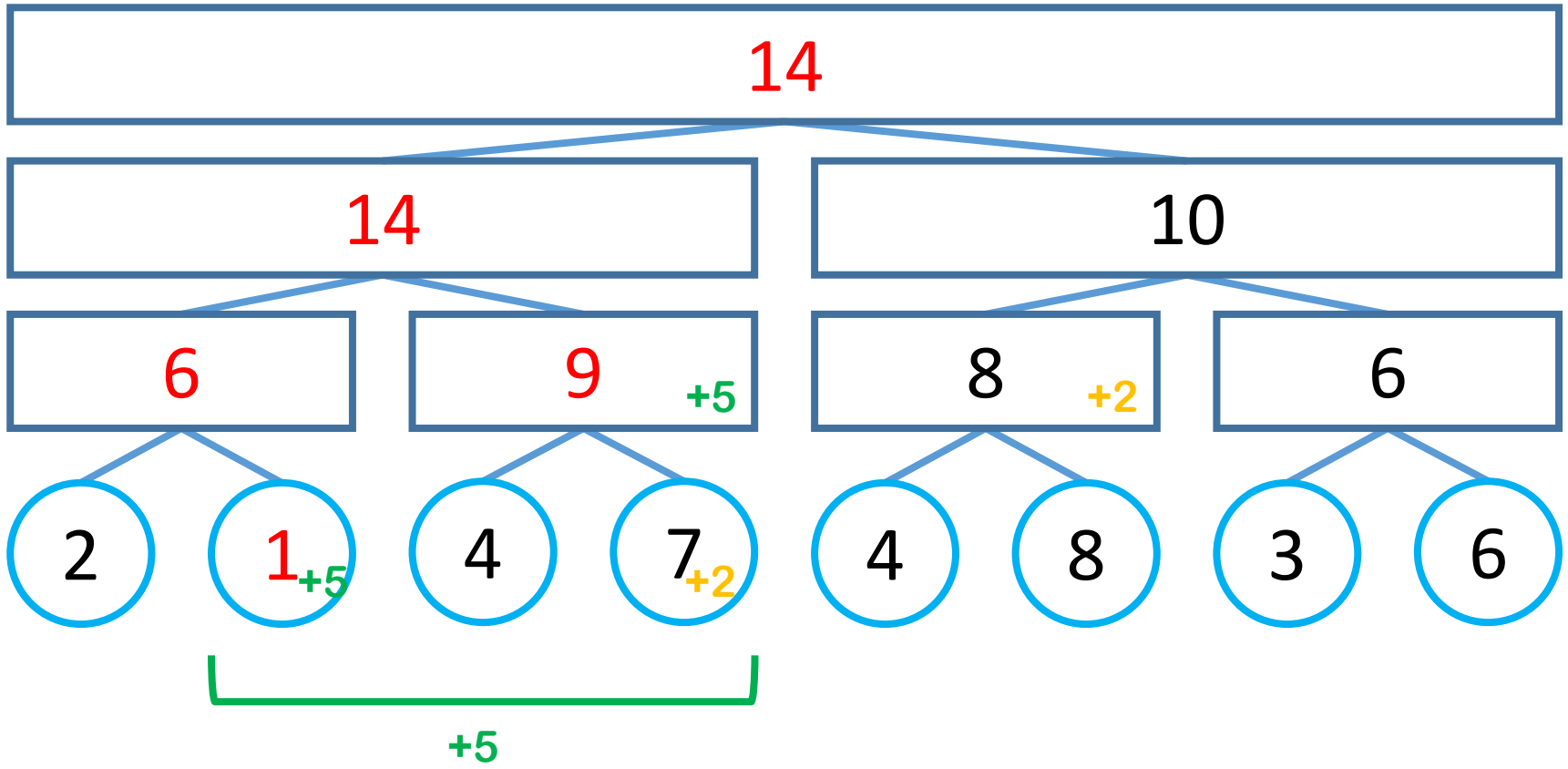
區間加值，區間求最大值



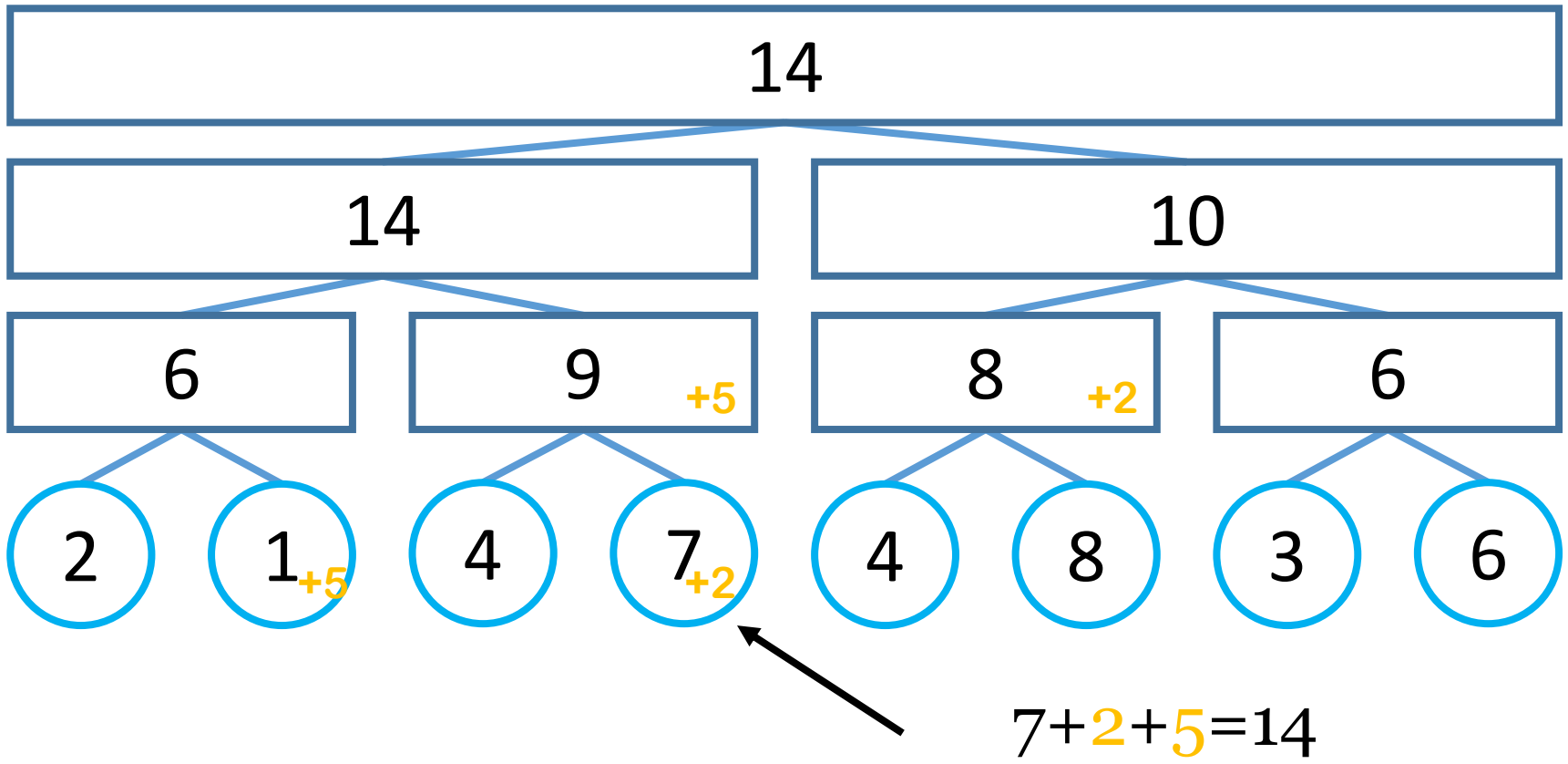
區間加值，區間求最大值



區間加值，區間求最大值

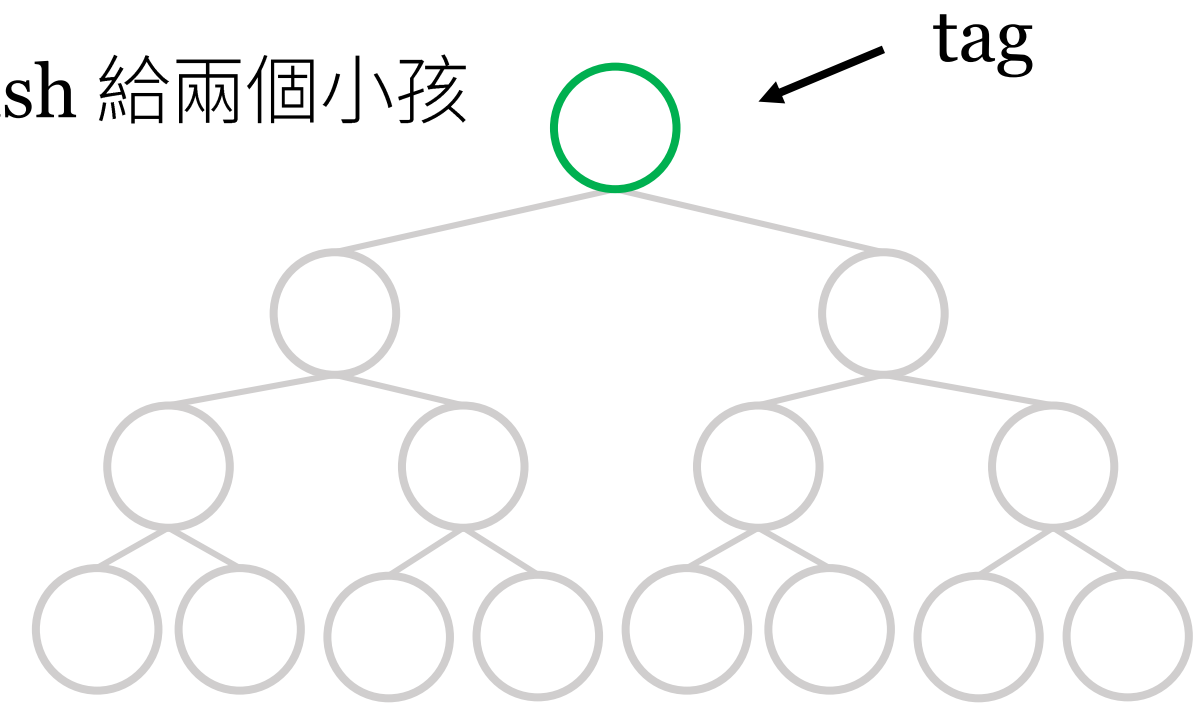


區間加值，區間求最大值



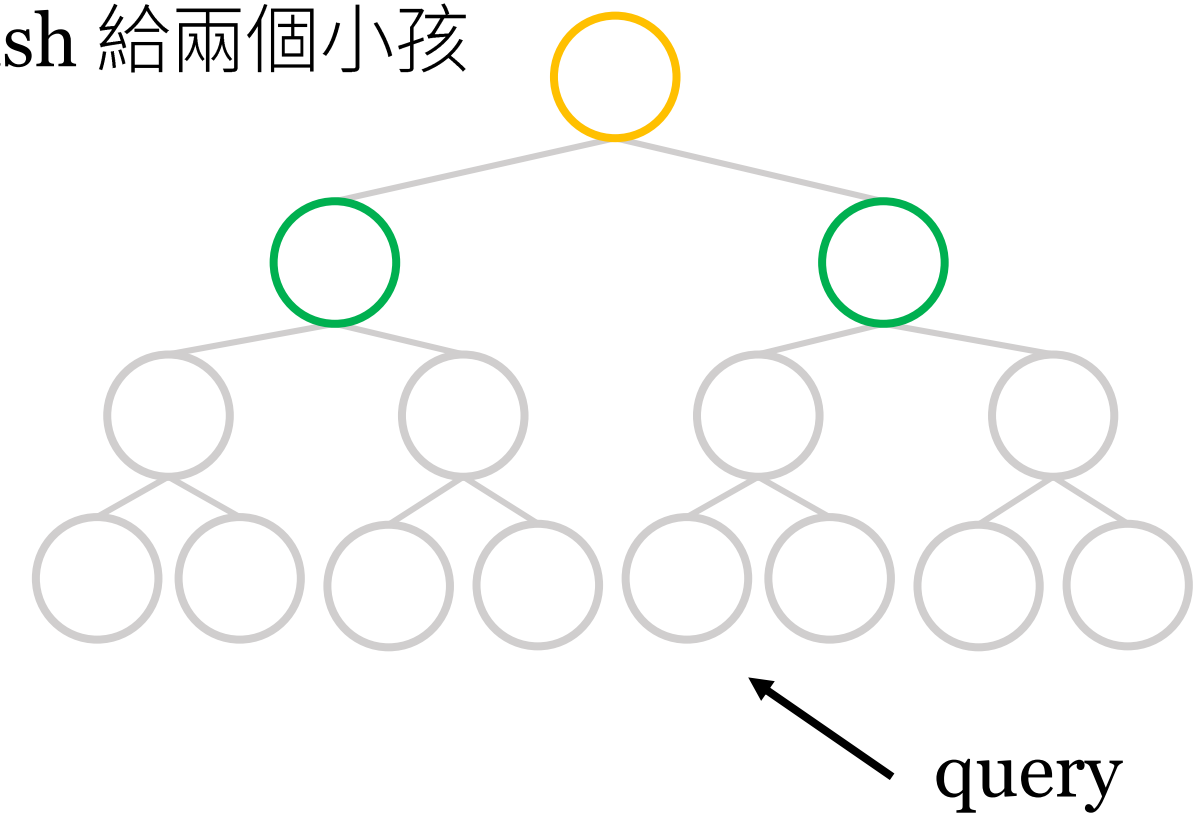
另一種 Lazy Tag 的實作方式

dfs 遇到 tag 就 push 給兩個小孩



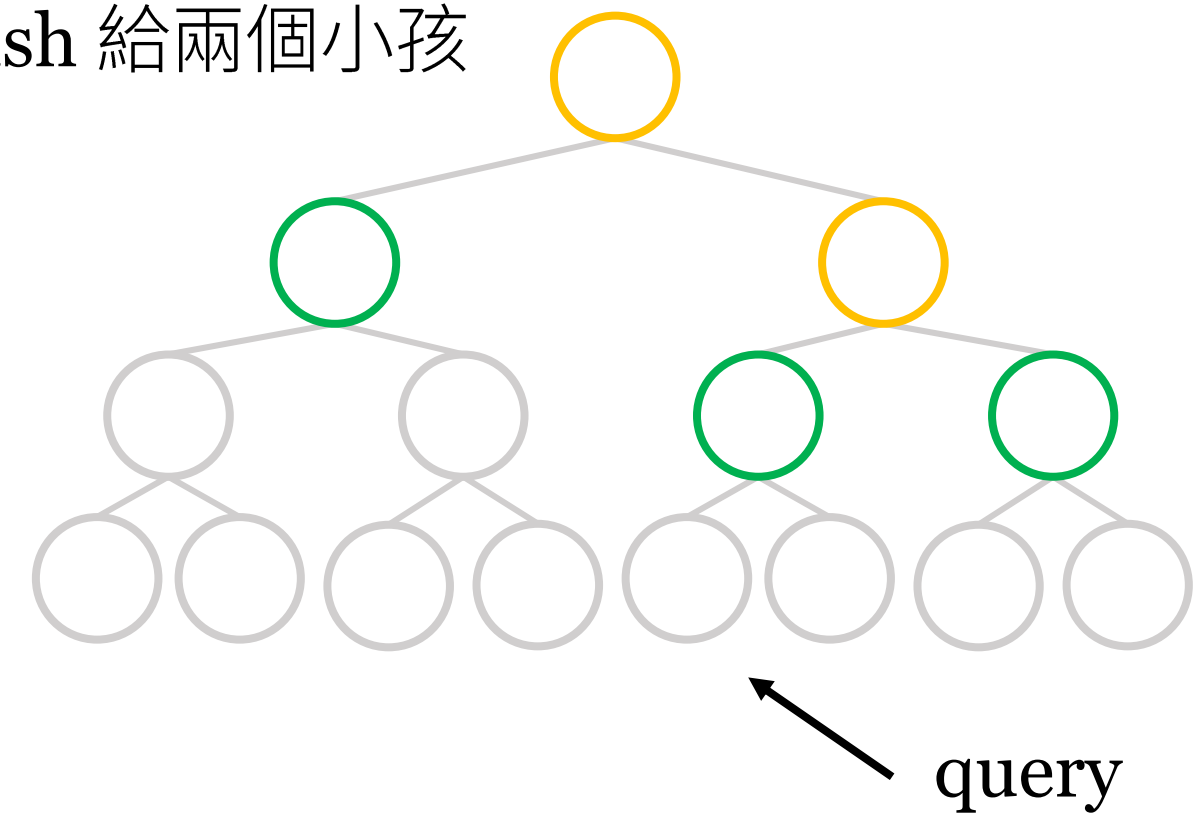
另一種 Lazy Tag 的實作方式

dfs 遇到 tag 就 push 給兩個小孩



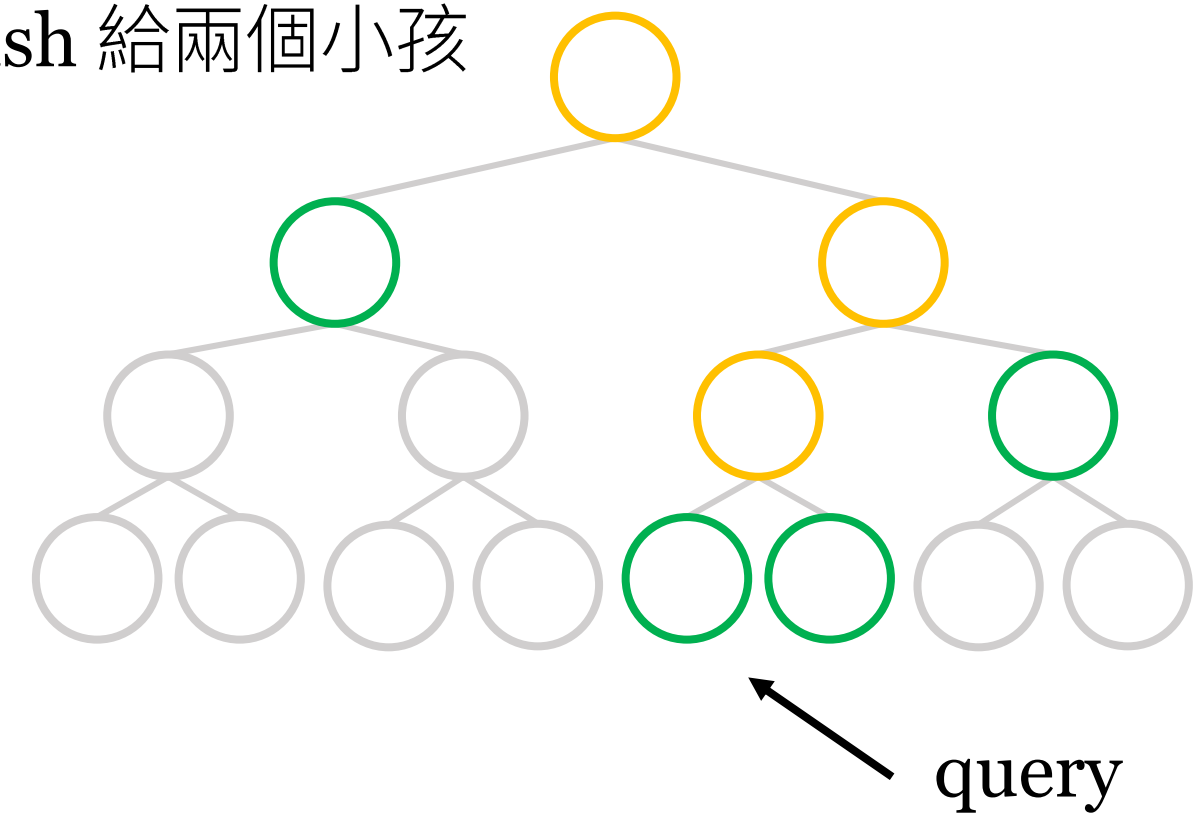
另一種 Lazy Tag 的實作方式

dfs 遇到 tag 就 push 給兩個小孩



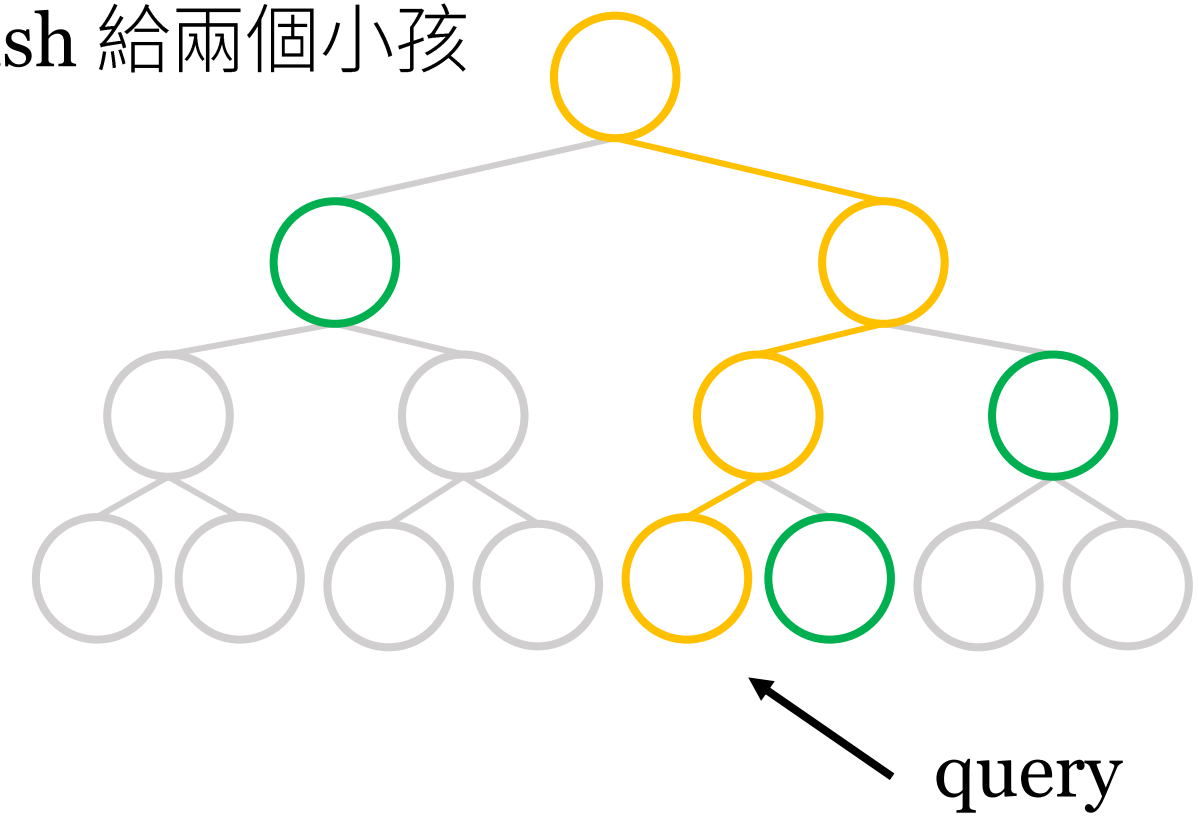
另一種 Lazy Tag 的實作方式

dfs 遇到 tag 就 push 給兩個小孩



另一種 Lazy Tag 的實作方式

dfs 遇到 tag 就 push 給兩個小孩



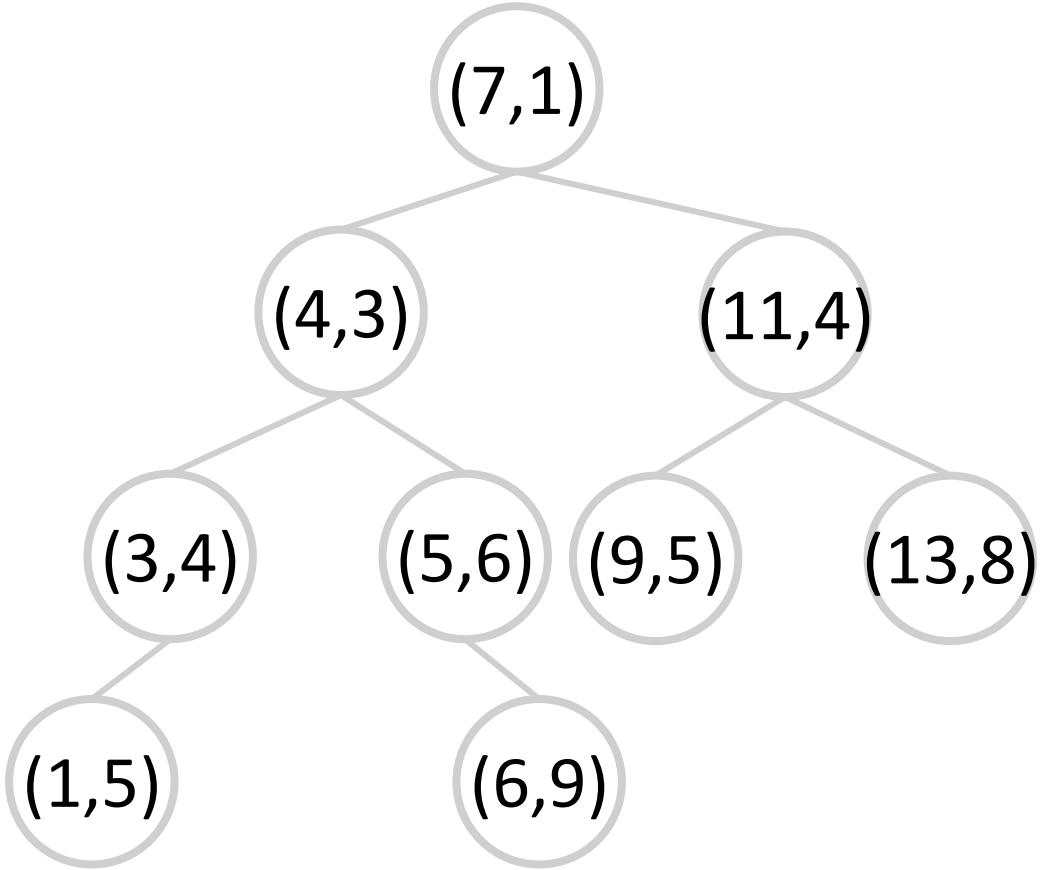
區間問題整理

	單點求值	區間總和	區間極值
無修改	Easy	Prifix Sum	線段樹 Sparse Table
單點加值	Easy	BIT	線段樹
單點改值	Easy	BIT	線段樹
區間加值	線段樹 +tag	線段樹 push BIT+差分	線段樹 +tag
區間改值	線段樹 +tag	線段樹 push	線段樹 +tag

Treap

Treap Node (key, pri)

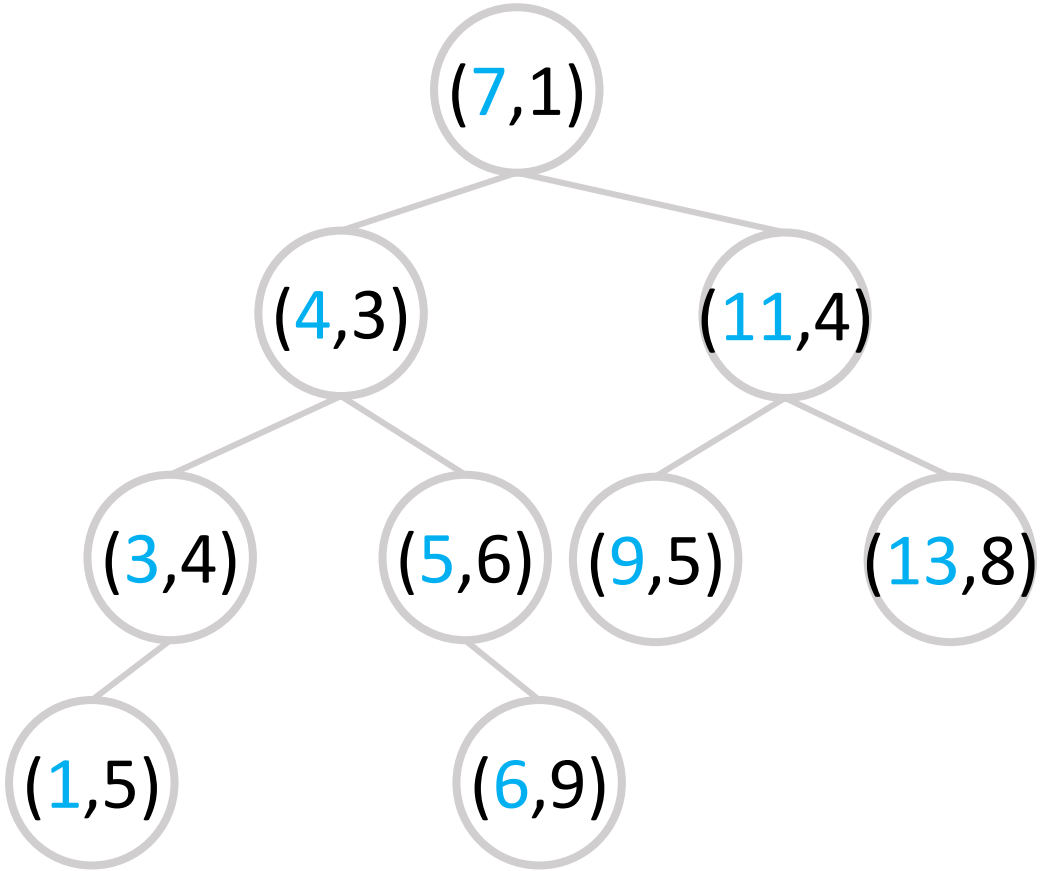
Treap = Tree + Heap



Treap Node (key, pri)

Treap = Tree + Heap

Tree: key 形成 BST

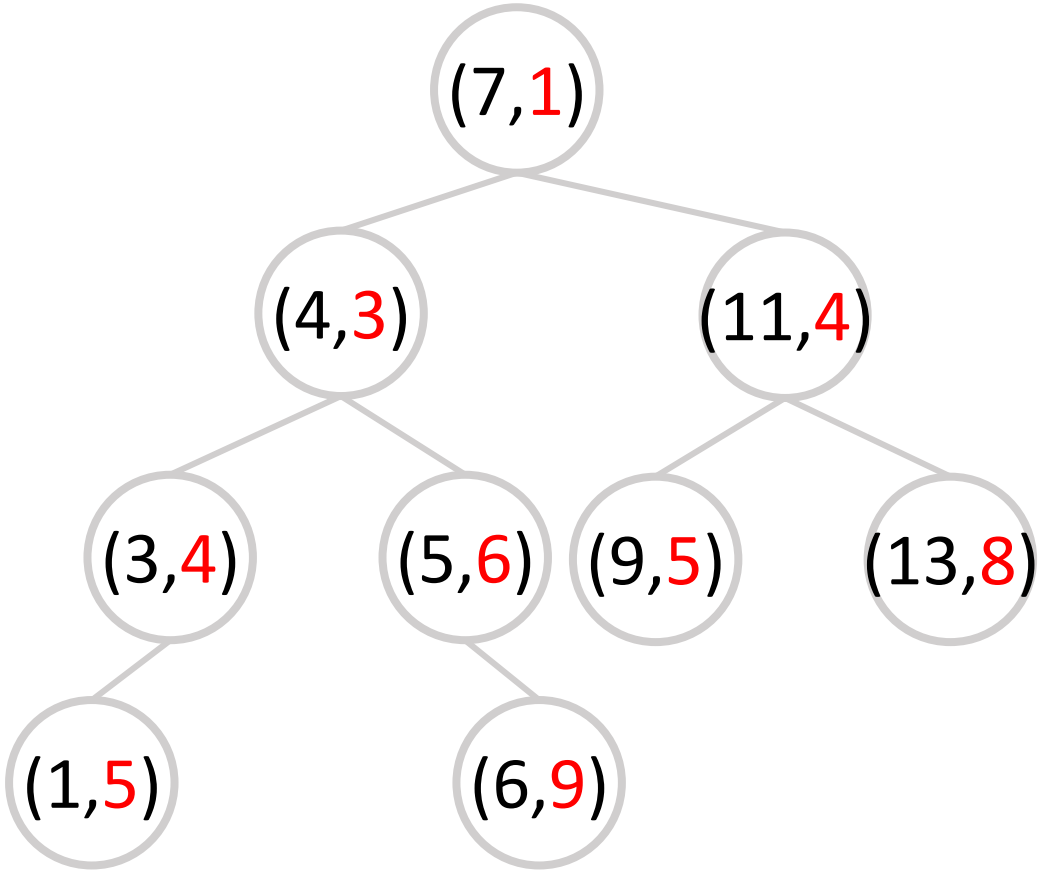


Treap Node (key, pri)

Treap = Tree + Heap

Tree: key 形成 BST

Heap: pri 形成 min heap



Treap Node (key, pri)

key 已經按照順序了，如何在 $O(n)$建立 Treap(?

Treap Node (key, pri)

key 已經按照順序了，如何在 $O(n)$建立 Treap(?

Cartesian tree!

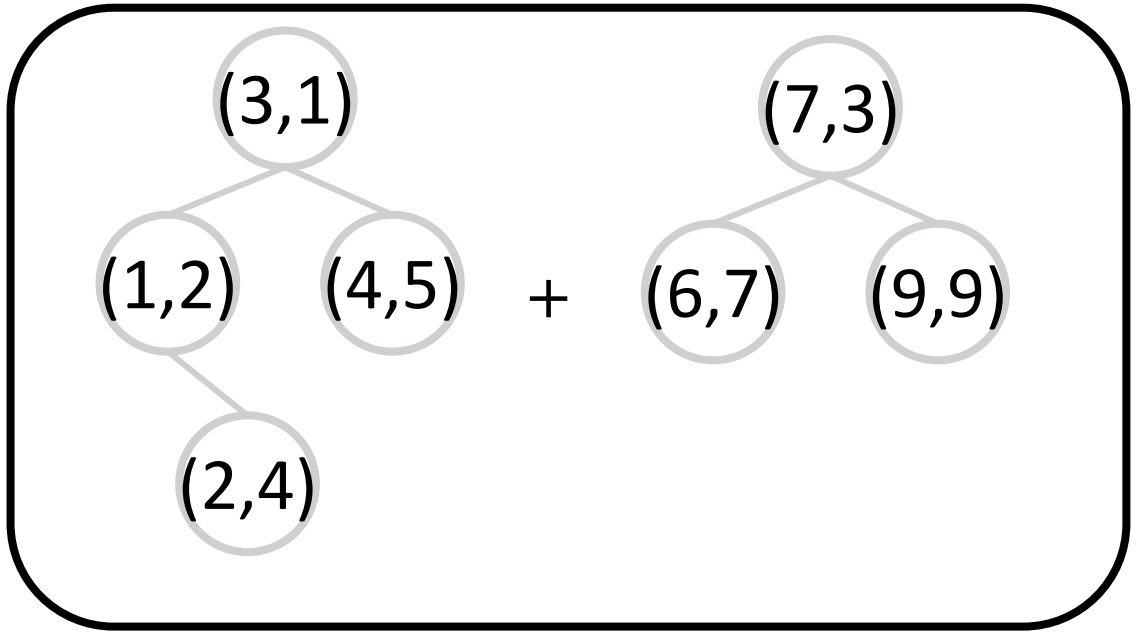
Treap Node (key, pri)

key 已經按照順序了，如何在 $O(n)$建立 Treap(?

pri 是隨機序列 → 樹高為 $O(\log n)$

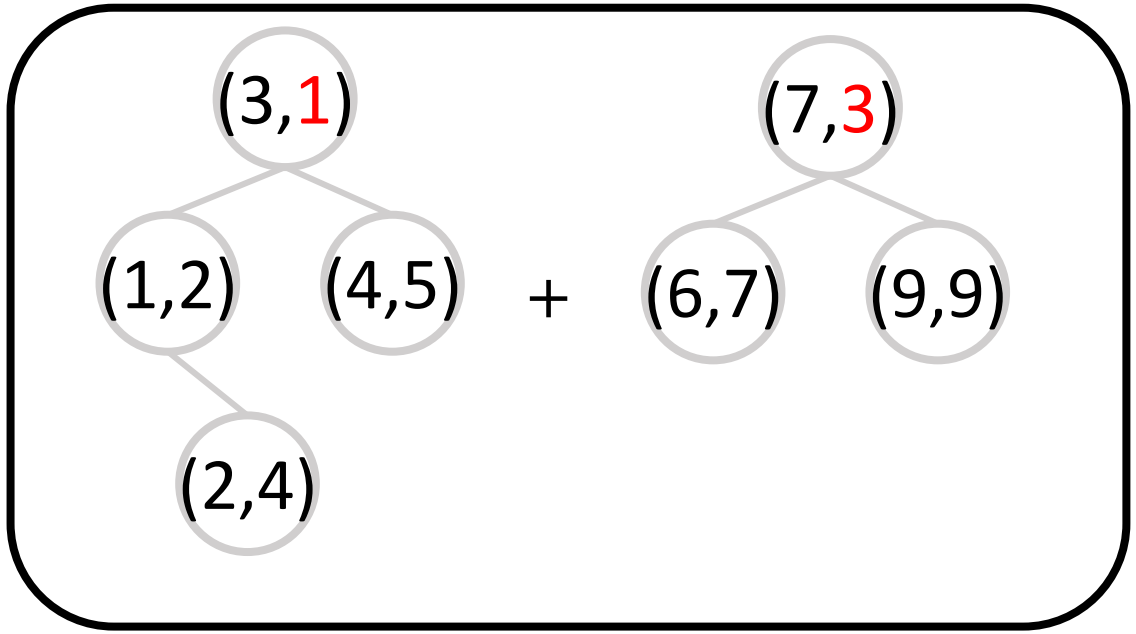
合併 Treap：merge(Treap *a, Treap *b)

假設 a 中最大的 key 小於等於 b 中最小的 key



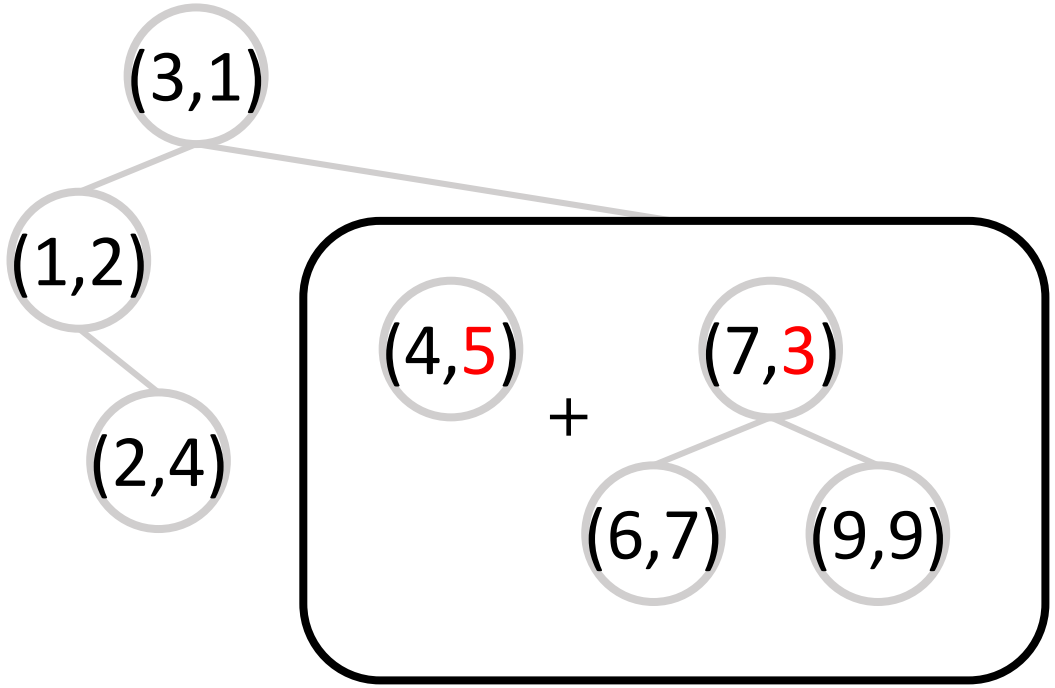
合併 Treap：merge(Treap *a, Treap *b)

比較 pri，小的當 root



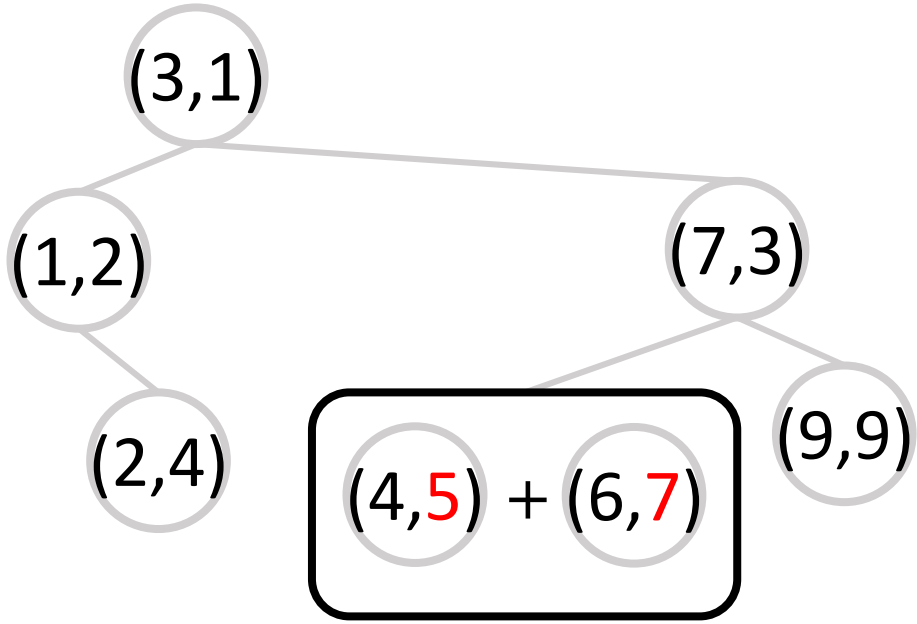
合併 Treap：merge(Treap *a, Treap *b)

比較 pri，小的當 root



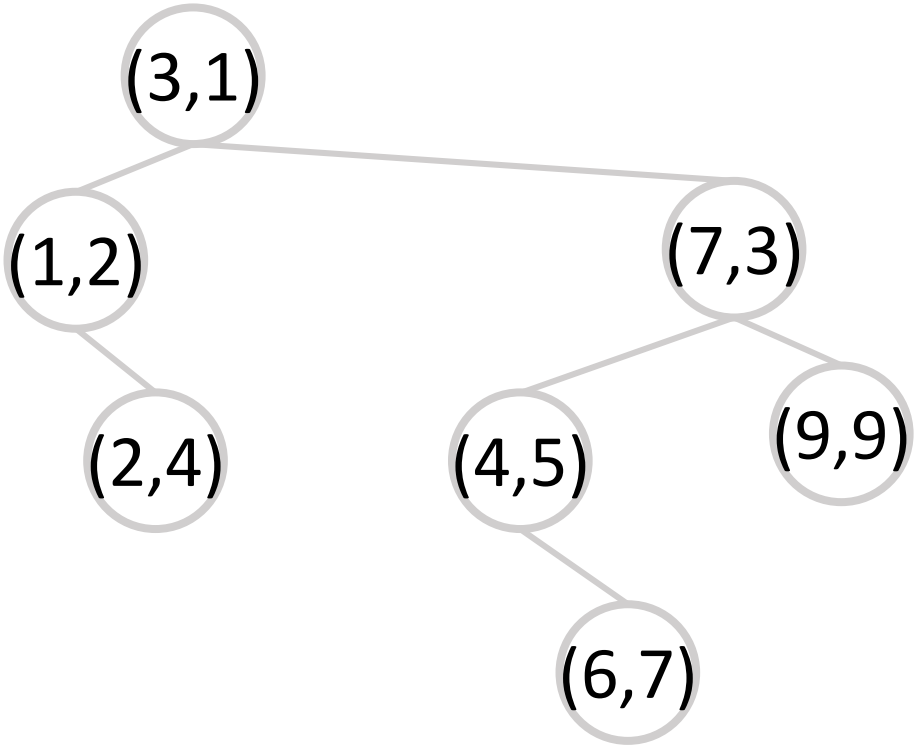
合併 Treap：merge(Treap *a, Treap *b)

比較 pri，小的當 root



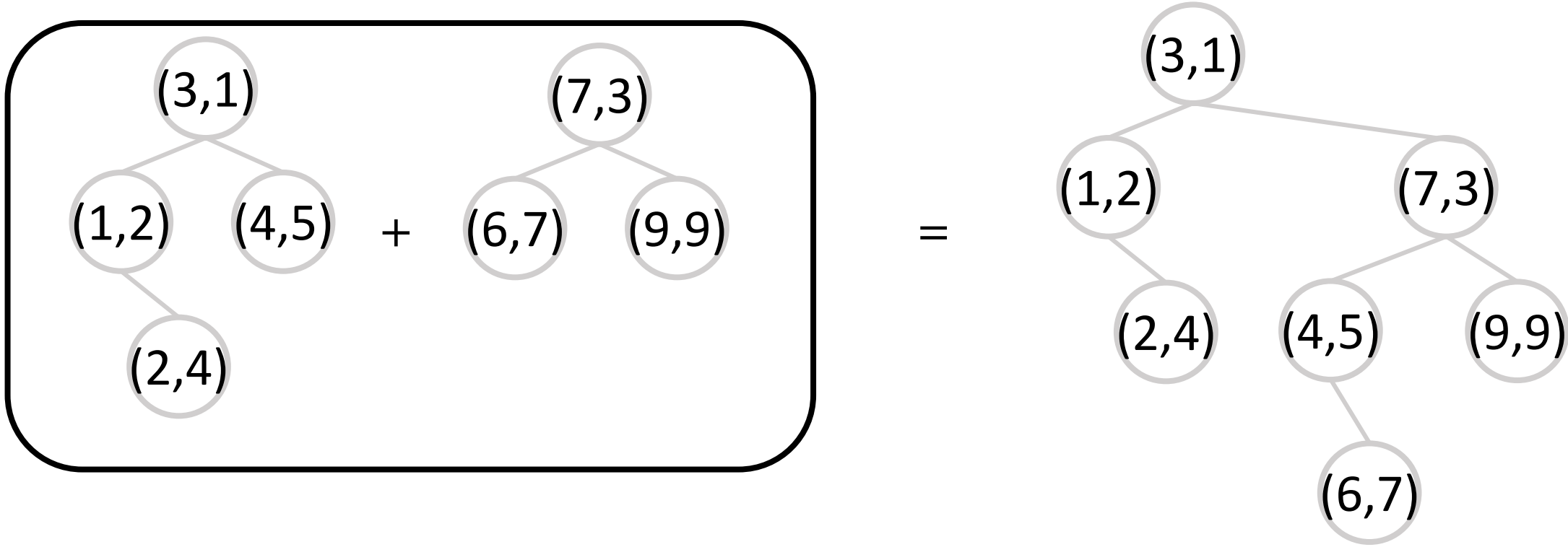
合併 Treap：merge(Treap *a, Treap *b)

比較 pri，小的當 root

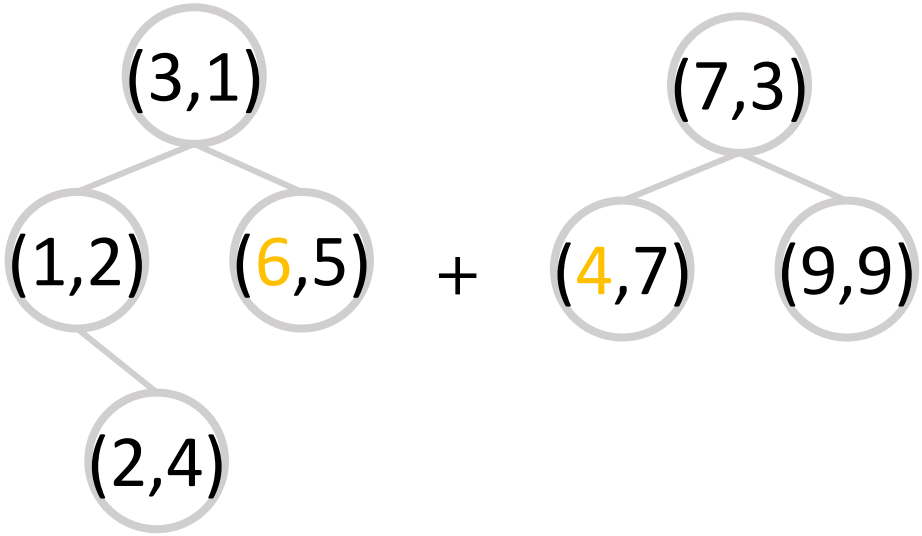


合併 Treap：merge(Treap *a, Treap *b)

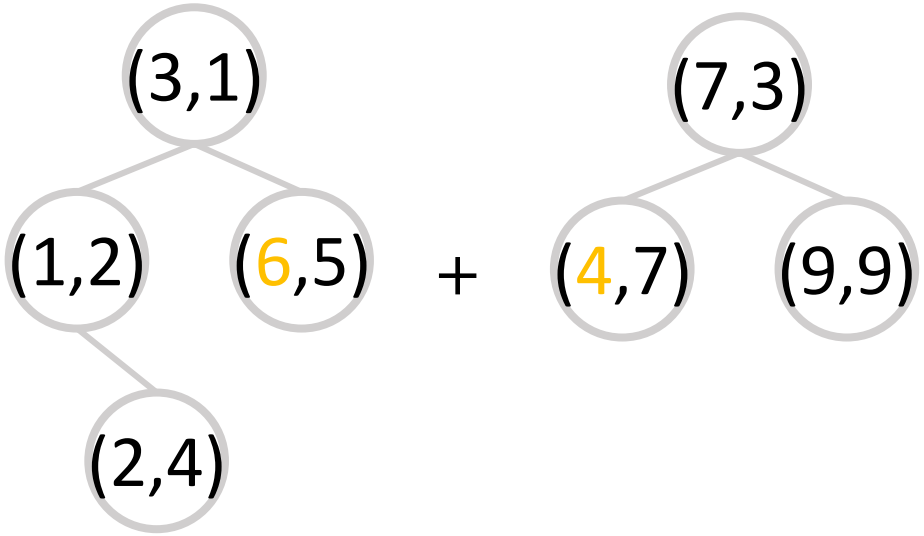
比較 pri，小的當 root



合併 Treap：兩邊的 key 不單調(?



合併 Treap：兩邊的 key 不單調(?



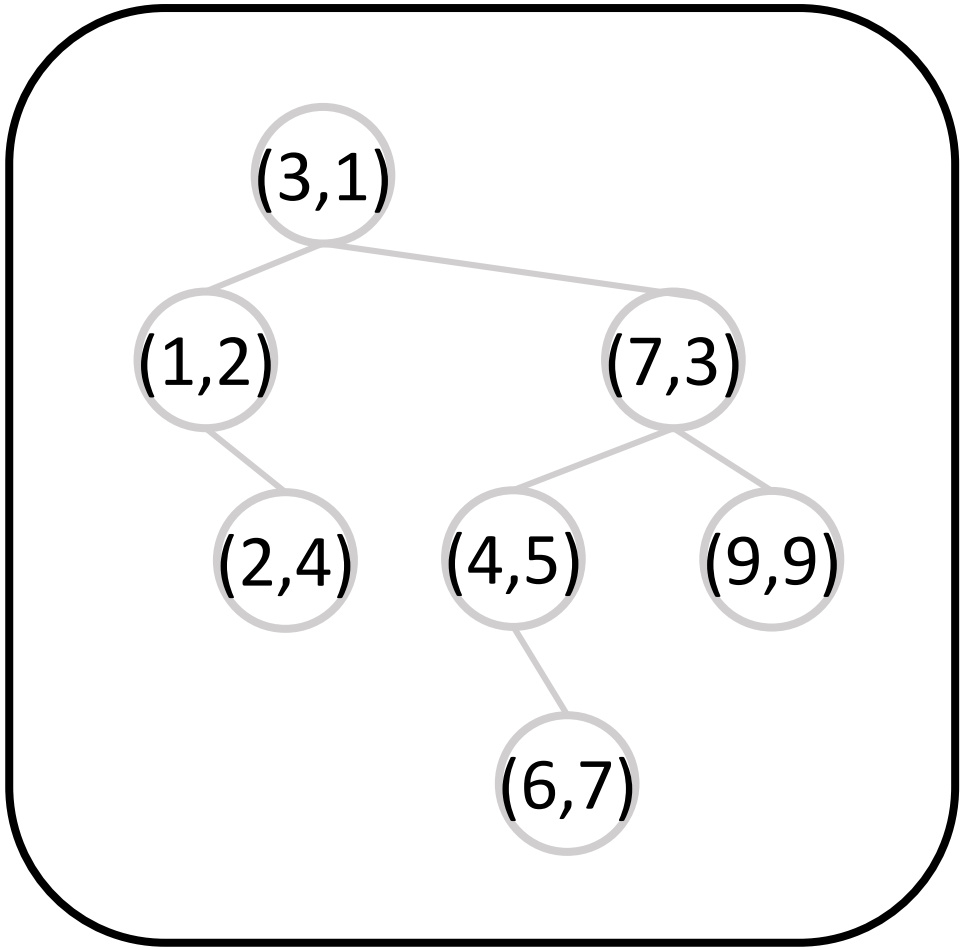
啟發式合併 $O(\text{交錯次數} \times \log(a+b))$

分裂 Treap：split(TP *t, TP*&a, TP*&b, k)

將 Treap t 分按照分成兩半

左半的 key 都小於 k

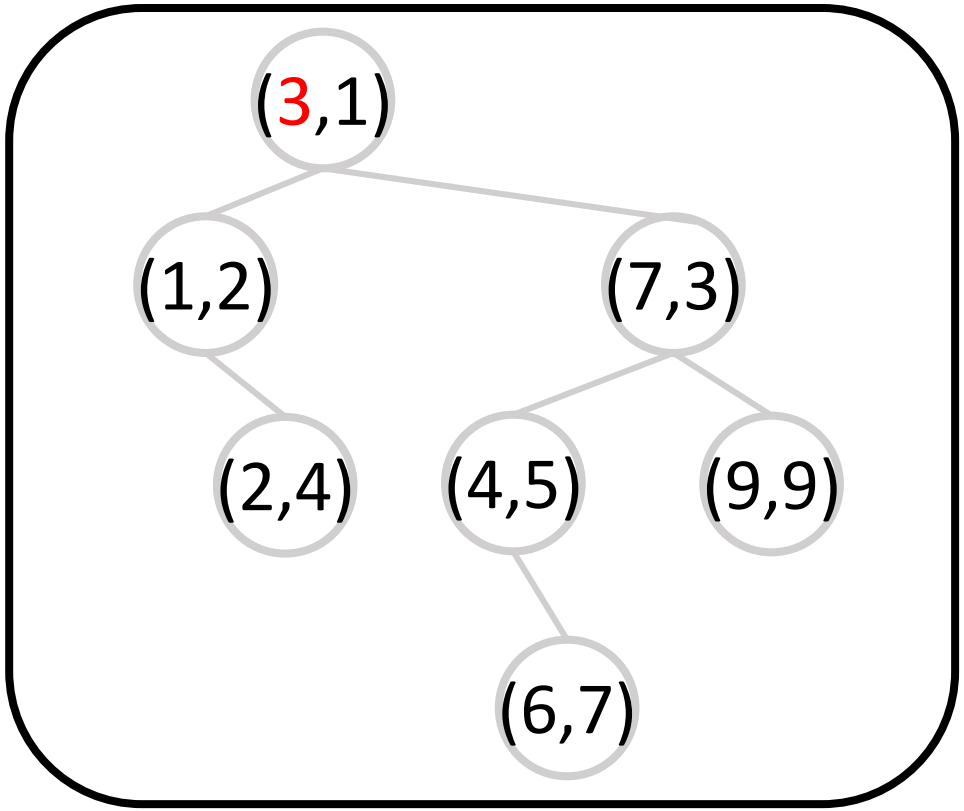
右半的 key 都大於等於 k



k=6

分裂 Treap：split(TP *t, TP*&a, TP*&b, k)

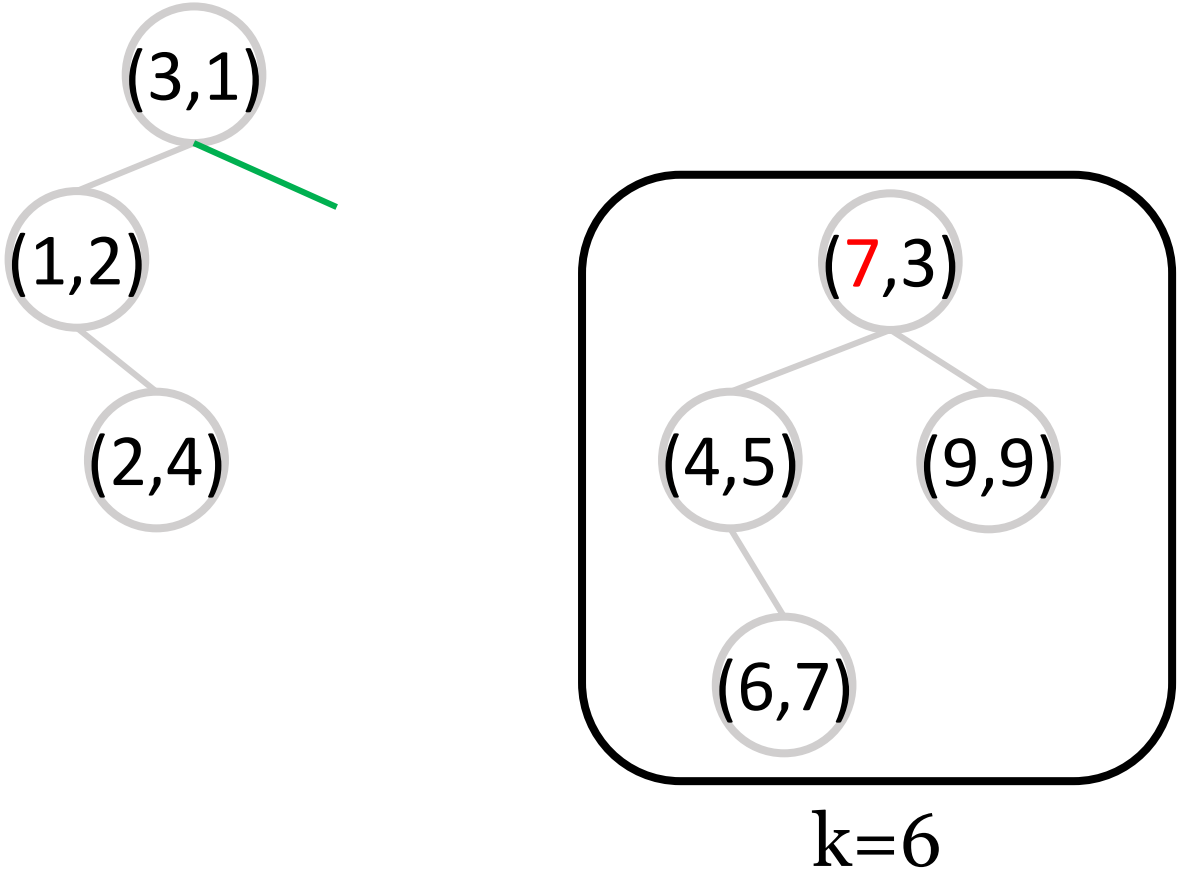
看 root 的 key 要放哪邊



k=6

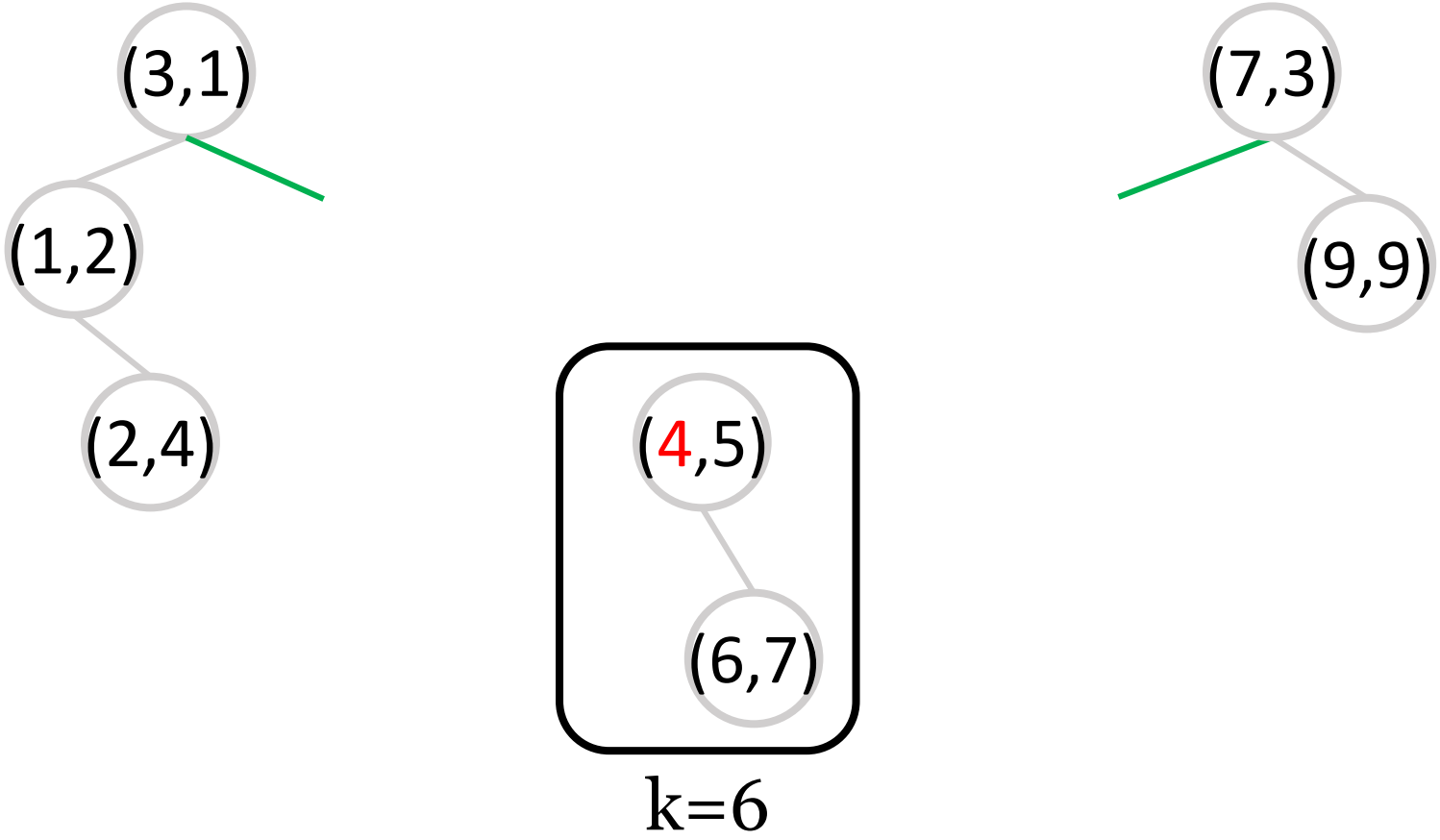
分裂 Treap：split(TP *t, TP*&a, TP*&b, k)

看 root 的 key 要放哪邊



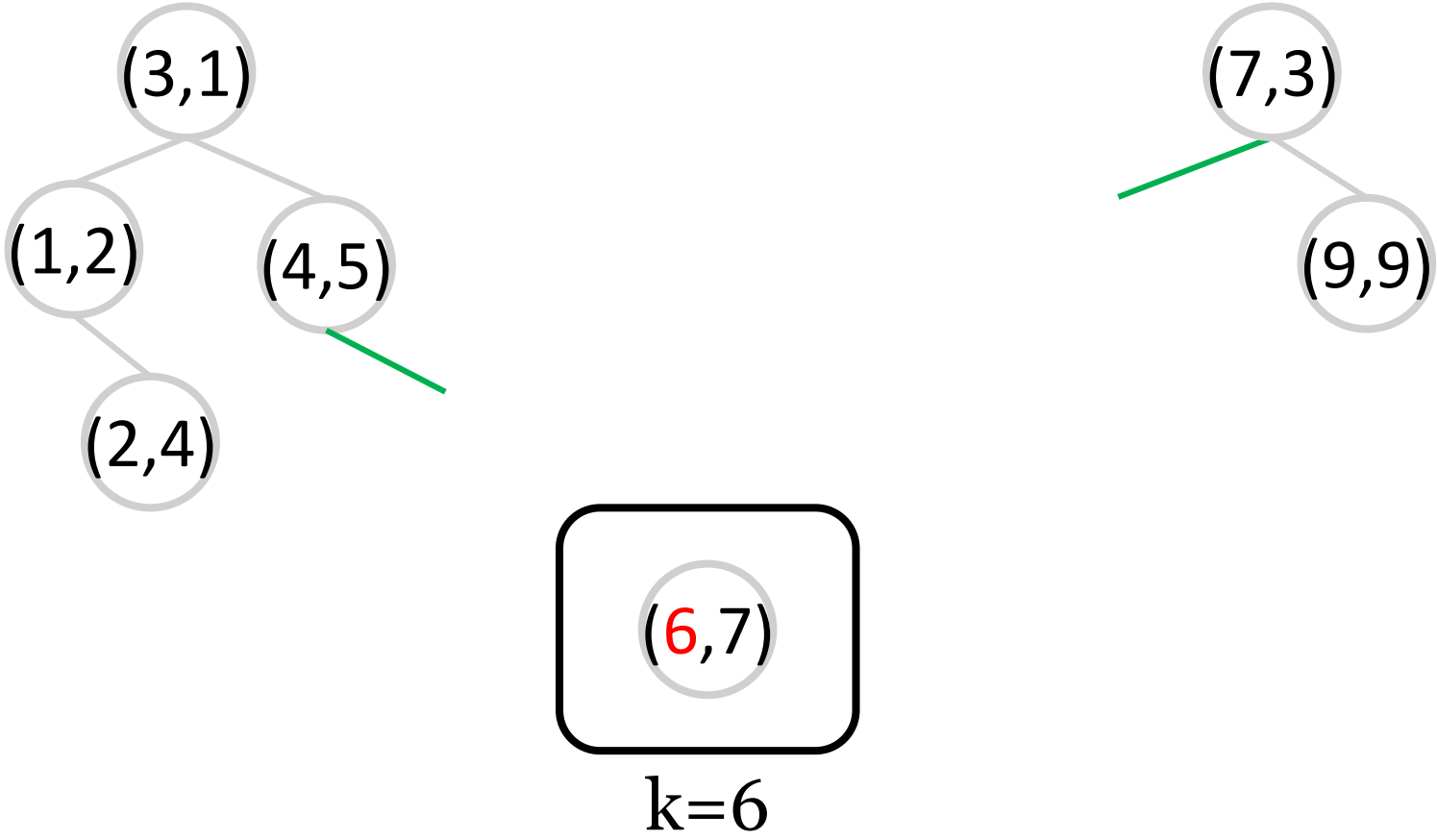
分裂 Treap：split(TP *t, TP*&a, TP*&b, k)

看 root 的 key 要放哪邊



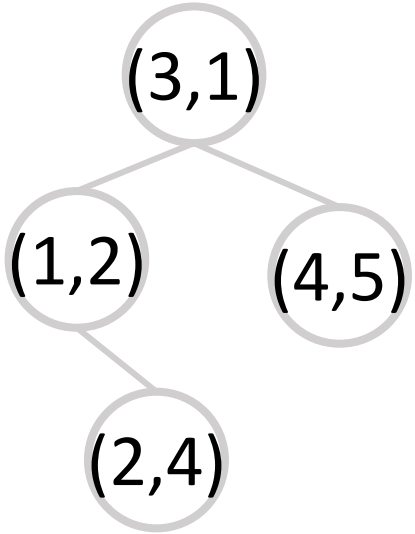
分裂 Treap：split(TP *t, TP*&a, TP*&b, k)

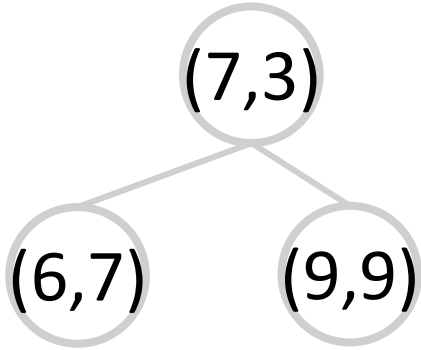
看 root 的 key 要放哪邊



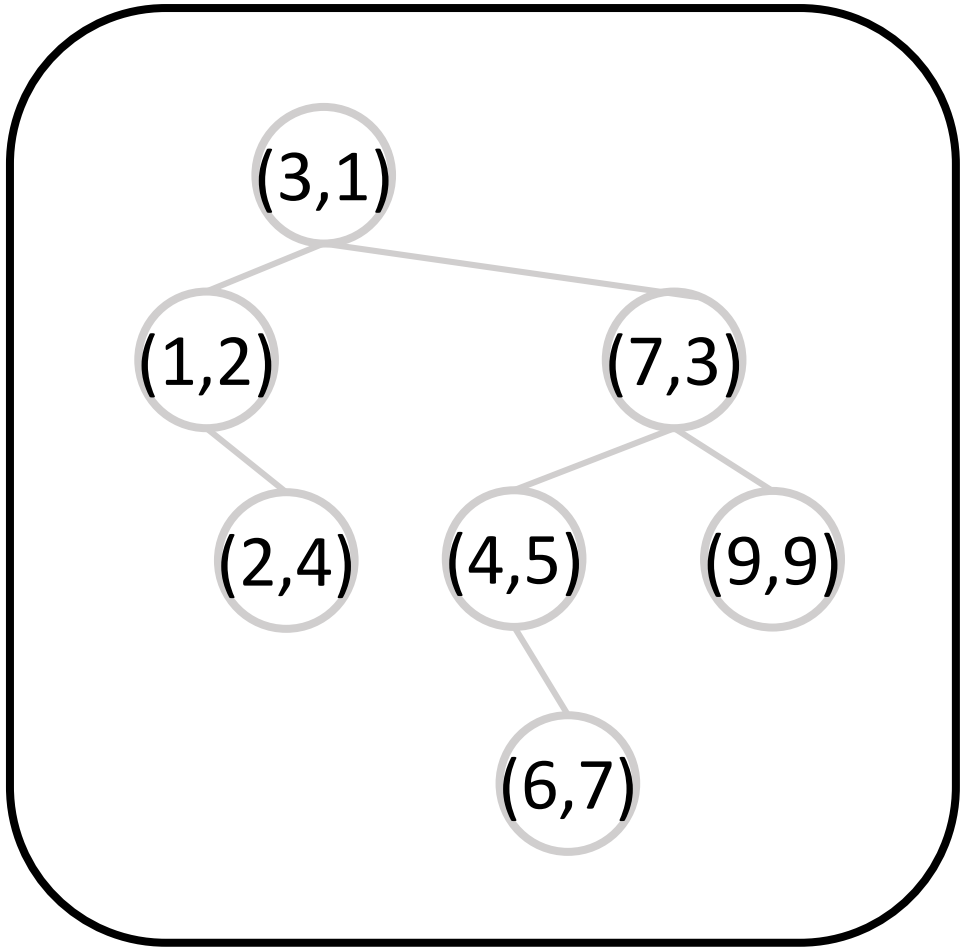
分裂 Treap：split(TP *t, TP*&a, TP*&b, k)

看 root 的 key 要放哪邊



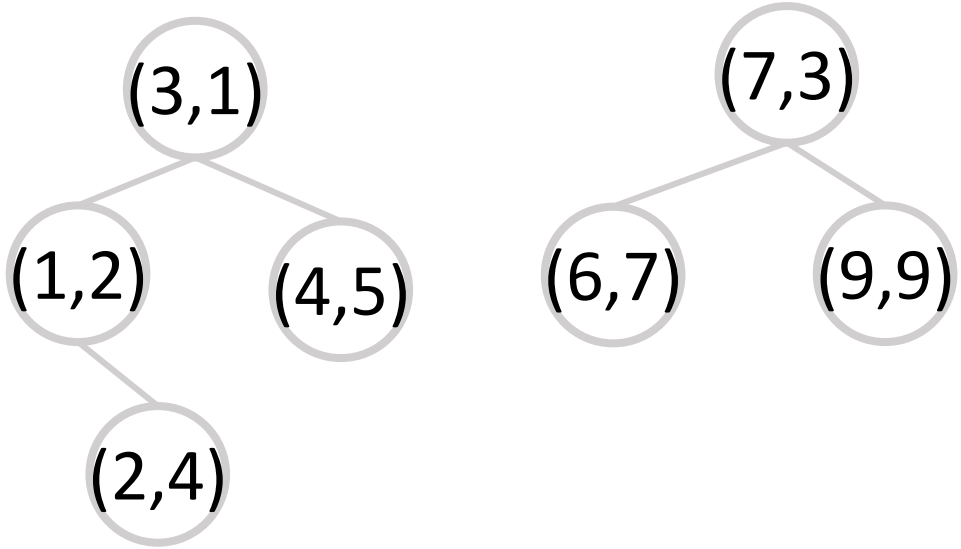


分裂 Treap：split(TP *t, TP*&a, TP*&b, k)



Split, k=6

=



POJ 3580

Input:

一個長度為 n 的整數序列 $a = [a_1, a_2, a_3, \ldots, a_n]$

Operation:

Add(L, R, D): 將 $a_L, \ldots, a_R$ 都加上 D

Reverse(L, R): 將 $a_L, \ldots, a_R$ 反轉為 $a_\mathrm{R}, \ldots, a_\mathrm{L}$

Insert(X, P): 在 a_X後方新增一項，數值為 P

Delete(X): 將 a_X 從序列中移除

Min(L, R): 輸出 $\min\{a_L, \ldots, a_R\}$

2015 TOI 1模 Problem C (TIOJ 1169)

Input:

一個長度為 n 的整數序列 $a = [a_1, a_2, a_3, \ldots, a_n]$

Update(x, v):

將 a_x 改為 v

Query(L, R, v)

回傳 $a_L, \ldots, a_R$ 內沒有出現 v 的最長子區間長度

範圍限制

$n \leq 2 \times 10^5, \mathrm{Q} \leq 2 \times 10^5$

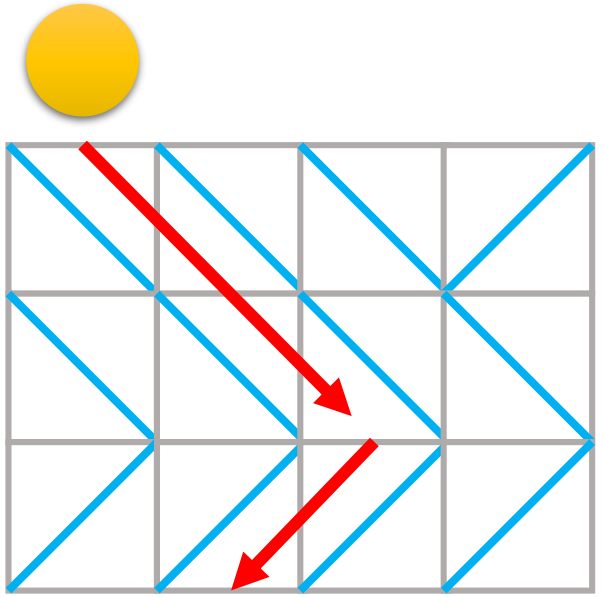
2017 ICPC 越南站 problem C (LA8436)

Input:

一個 $n \times m$ 大小的棋盤格 ，每個有一個隔板

Query:

輸出從左上落下的求會掉到底下的哪一格



Flip(r, c)

把座標為 (r, c) 的隔板換方向

範圍限制

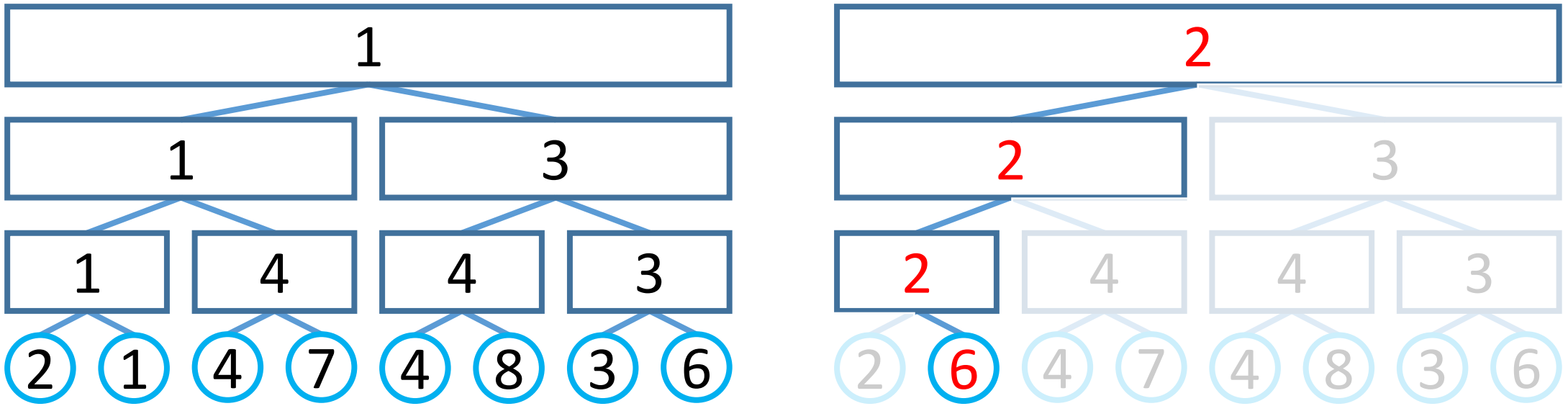
$n \times m \leq 10^5, Q \leq 10^5$

持久化資料結構

持久化線段樹

一個新的版本只要 $O(\log n)$個 node

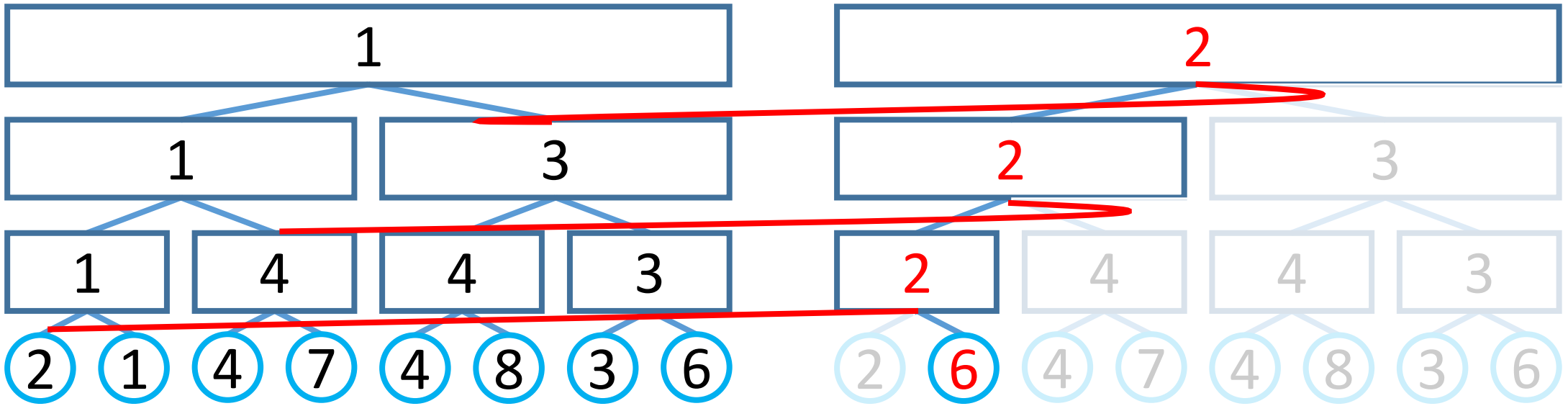
n 個版本只需要 $O(n \log n)$ 空間



持久化線段樹

一個新的版本只要 $O(\log n)$個 node

n 個版本只需要 $O(n \log n)$ 空間



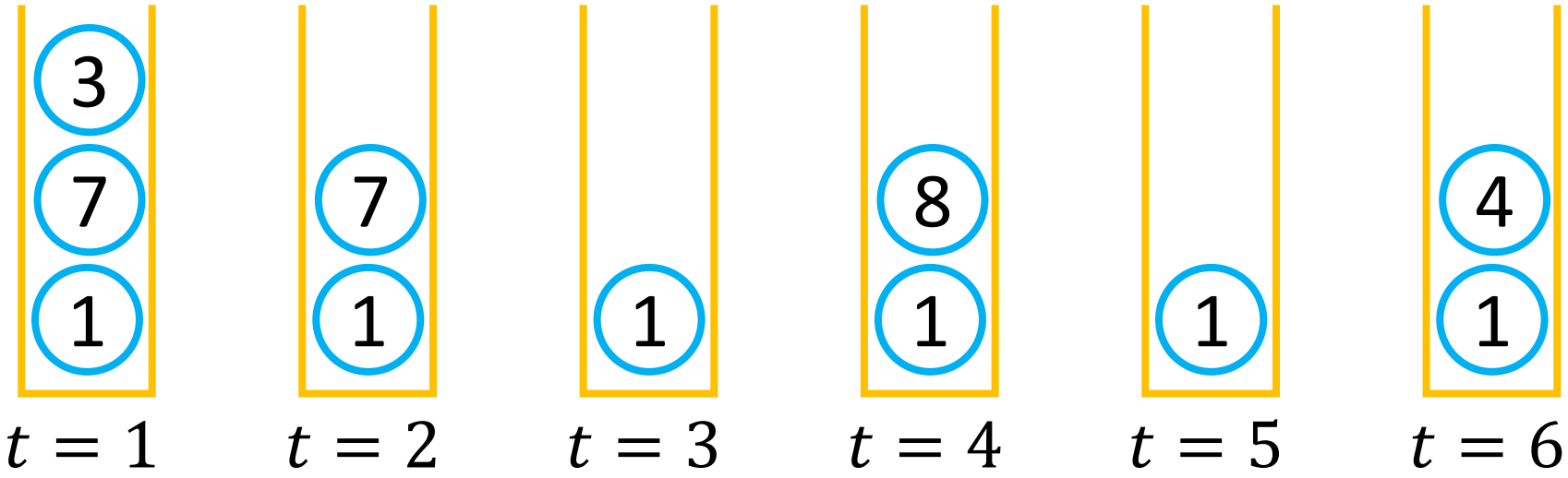
持久化 Stack

Input:

stack 的初始狀態，以及每一個時間進行的操作 (push, pop)

Query(t, k):

輸出時間點 t 時 stack 由底下向上數的第 k 個數字



持久化 Stack

Input:

stack 的初始狀態，以及每一個時間進行的操作 (push, pop)

Query(t, k):

輸出時間點 t 時 stack 由底下向上數的第 k 個數字

用持久化線段樹實作持久化 stack
空間 $O(n \log n)$，查詢時間$O(\log n)$

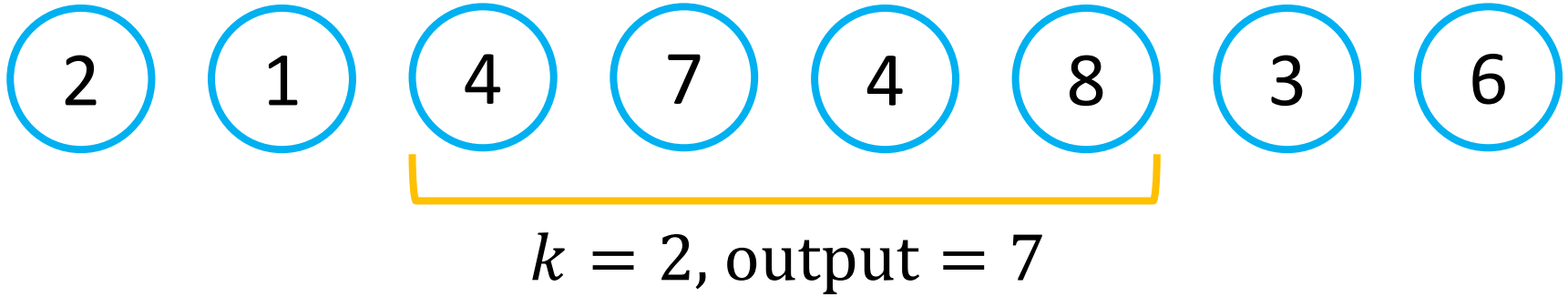
區間第 k 大查詢

Input:

一個長度為 n 的整數序列 $a = [a_1, a_2, a_3, \ldots, a_n]$

Query(L, R, k):

回傳 $a_L, a_{L+1}, \ldots, a_R$ 中第 k 大的數字



區間第 k 大查詢

Input:

一個長度為 n 的整數序列 $a = [a_1, a_2, a_3, \dots, a_n]$

Query(L, R, k):

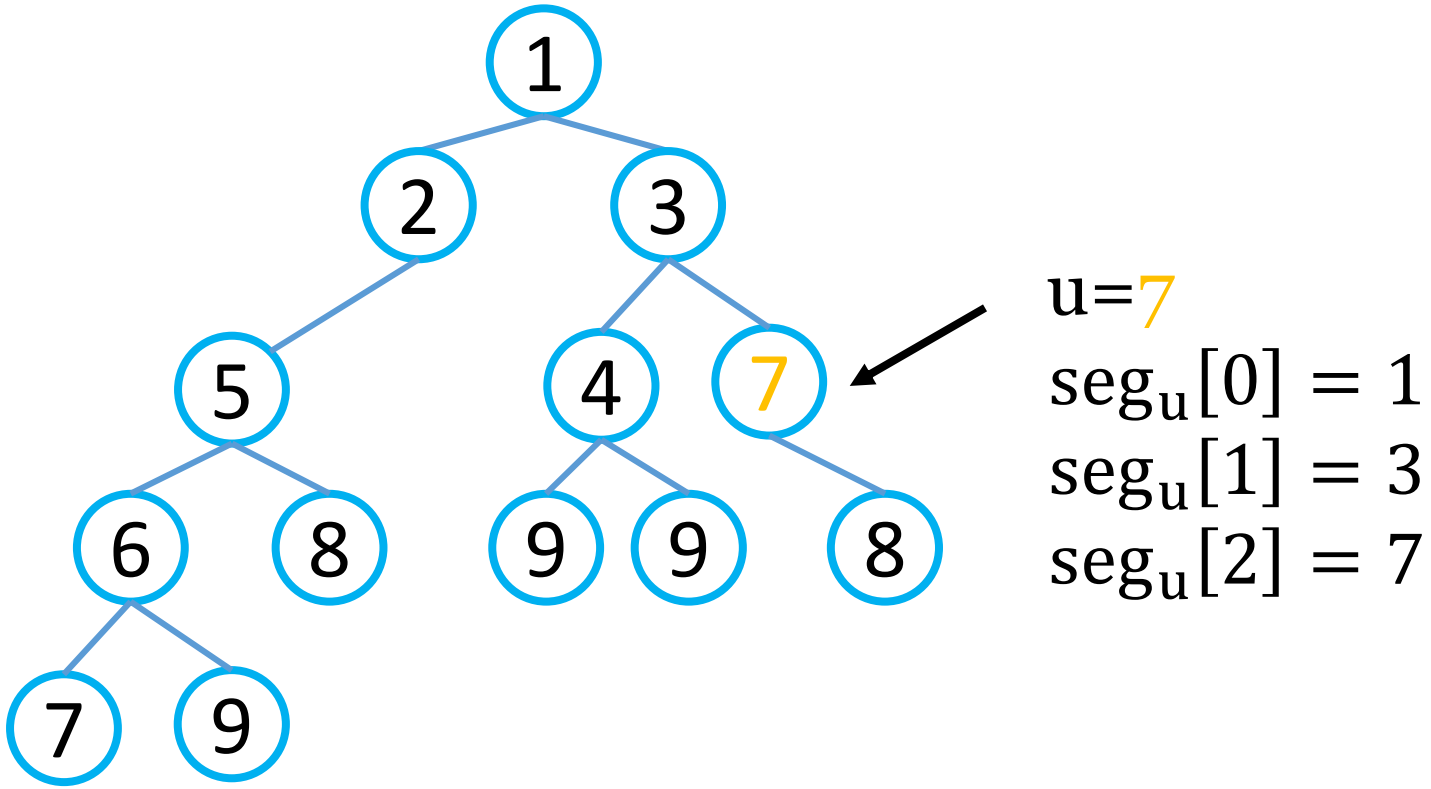
回傳 $a_L, a_{L+1}, \dots, a_R$ 中第 k 大的數字

$$\text{seg}_t[i,i] = \begin{cases} 1, \text{if } a_i \le t \\ 0, \text{otherwise} \end{cases}$$

二分搜尋最小的 t ，滿足 $\text{seg}_t[L, R] \ge k$

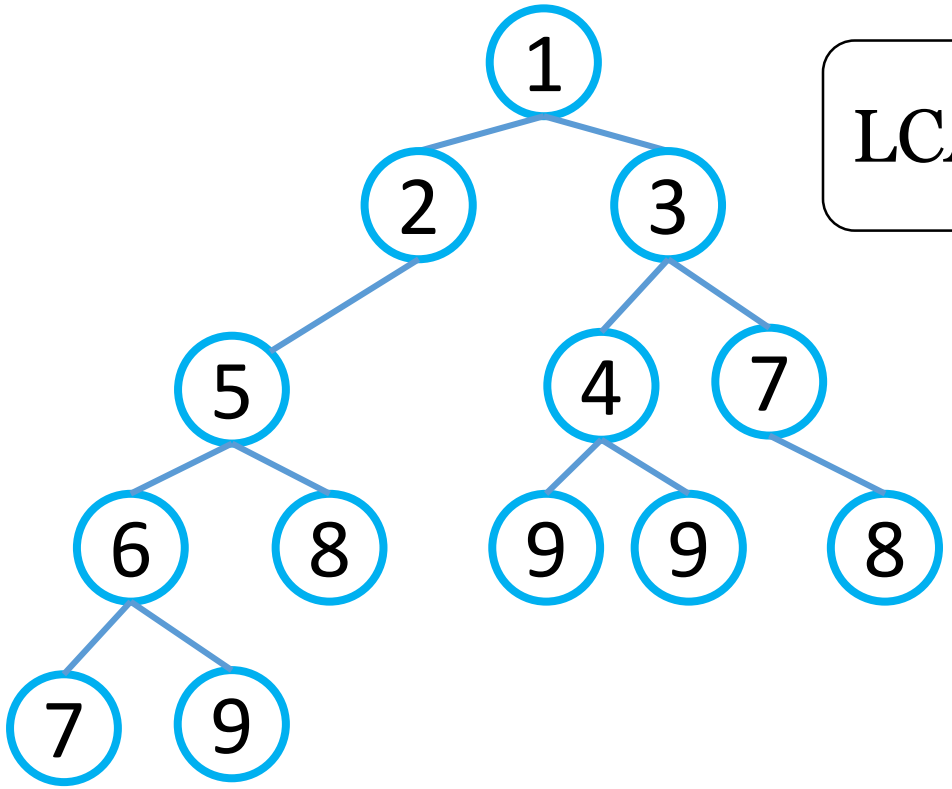
持久化祖先陣列

$seg_u[i]$ = 節點 u 從 root 往下數的第 i 個節點上的數字



持久化祖先陣列

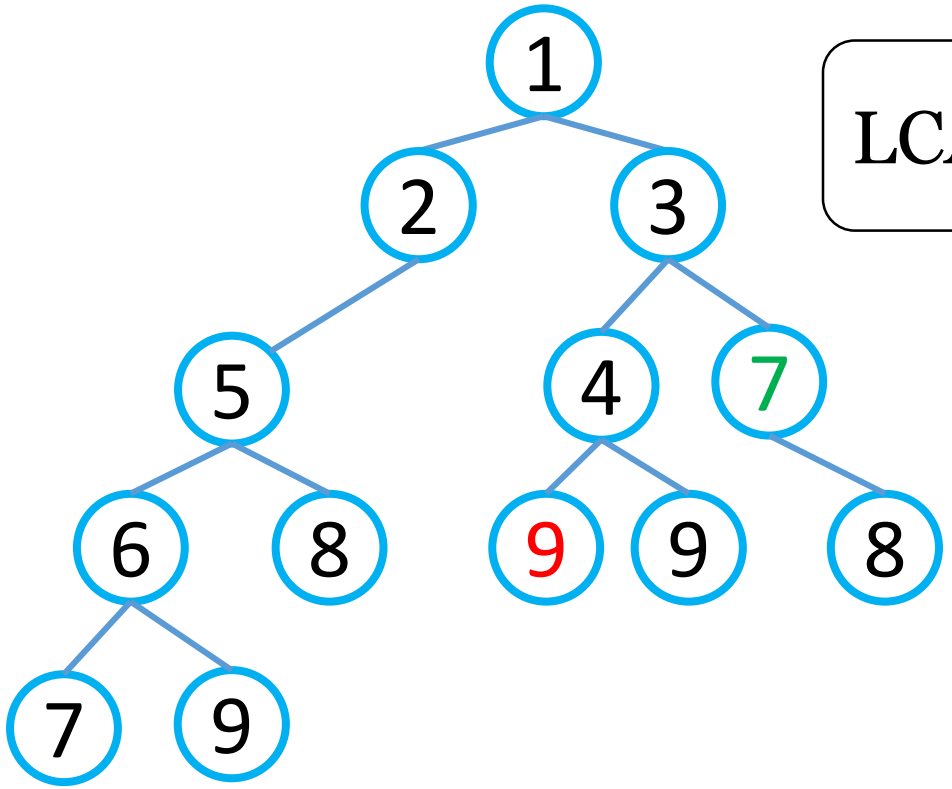
$seg_u[i]$ = 節點 u 從 root 往下數的第 i 個節點上的數字



LCA = 兩個版本的祖先陣列 LCP

持久化祖先陣列

$seg_u[i]$ = 節點 u 從 root 往下數的第 i 個節點上的數字



LCA(9, 7)= LCP([1,3,4,9], [1,3,7])

資料結構嵌套

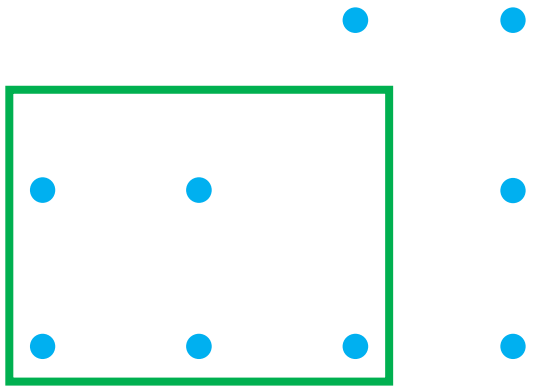
問題

add (X, Y):

在座標 (X, Y) 的位置新增一個點

Query(X, Y):

回傳平面上有幾個點 (x, y) 滿足 $x \leq X$ 且 $y \leq Y$



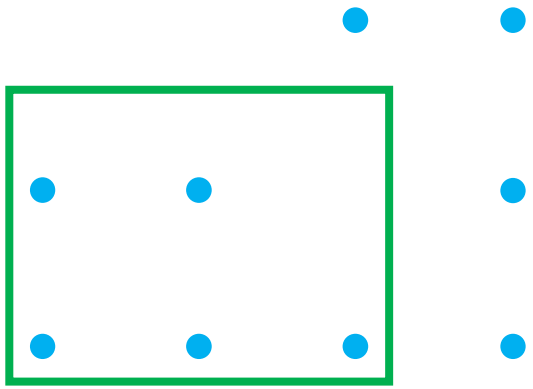
問題

add (X, Y):

在座標 (X, Y) 的位置新增一個點

Query(X, Y):

回傳平面上有幾個點 (x, y) 滿足 $x \leq X$ 且 $y \leq Y$



2D BIT

2D BIT