Segment Tree 入門

kya
triC

August 02, 2023

目次

Range Minimum Query (RMQ)

数列 $A = \{a_0, a_1, \cdots, a_{n-1}\}$ に対し、次の２つの操作を行うプログラムを作成せよ。

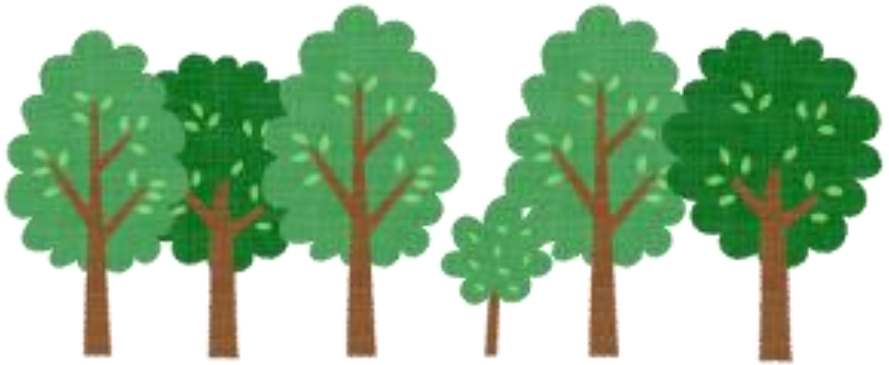
- $update(i, x)$: a_i の値を x に変更する。
- $find(l, r)$: $a_l, a_{l+1}, \cdots, a_{r-1}$ の最小値を出力する。

ただし $a_i (i = 0, 1, \cdots, n-1)$ は $2^{31} - 1$ で初期化されているものとする。

Segment Tree とは

- 配列に対し、以下のような操作を行うことが可能
  - ある一点の値を更新する
  - 任意の区間の総和や最小値, 最大公約数などを取得する

- 競プロ頻出のデータ構造
  - Difficulty 青色以上の問題でよく出る
  - ABC だと F ~ Ex くらい



Segment Tree が扱える演算

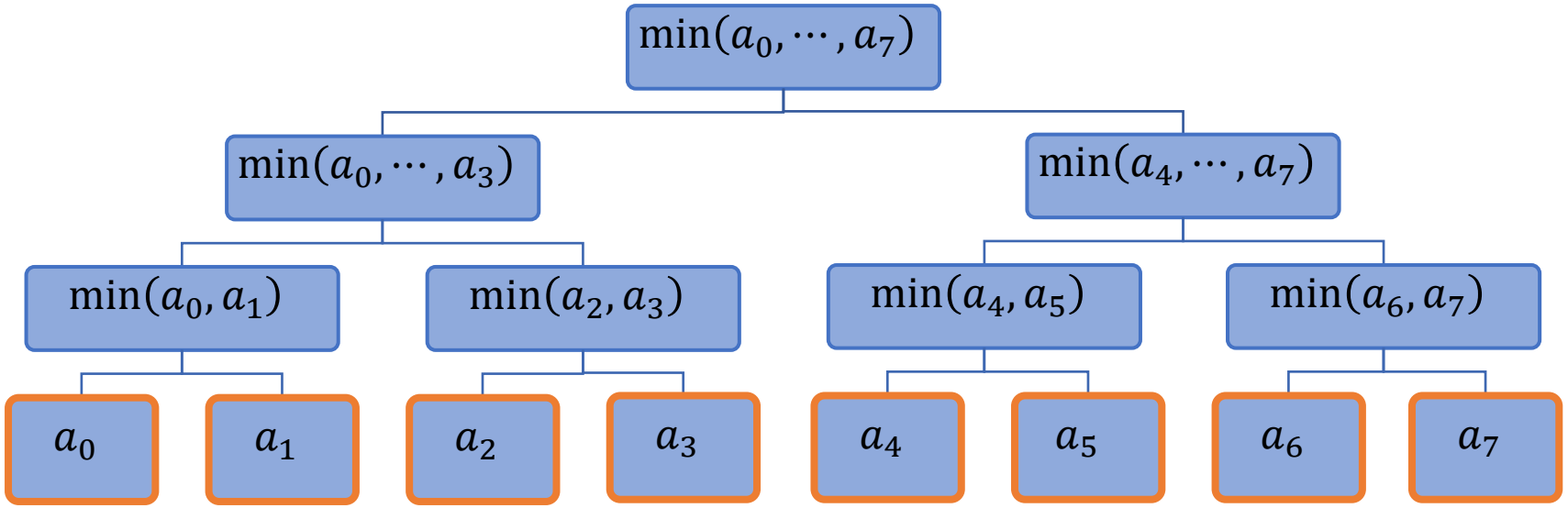
Segment Tree はモノイドの列を扱うことができる

<u>モノイドの例</u>

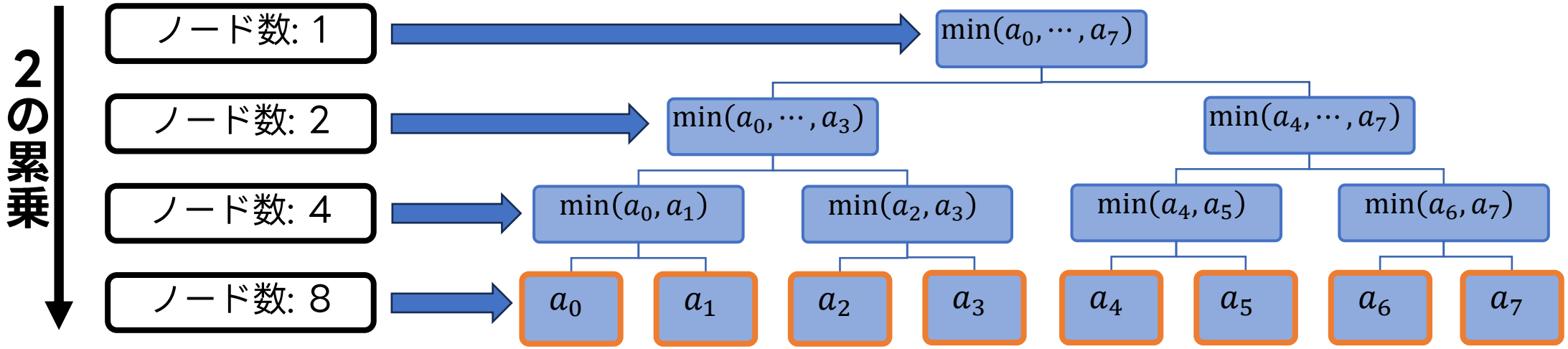
- 四則演算
- min, max
- 最大公約数, 最小公倍数
- 一次関数の合成
  - $f(x) = ax + b, g(h) = cx + d$ に対して$f\big(g(x)\big) = acx + ad + b$

おきもち (1/3)

- 配列を二分木で管理する
  - ノード数は $2N$
  - 木の高さは $O(\log(N))$



ノード数と木の高さ



ノード数

1 段あたりのノード数は倍増

$N = 2^k$ とする

$$1 + 2 + 4 + \cdots + N$$
$$= 2^0 + 2^1 + 2^2 + \cdots + 2^k$$
$$= 2^{k+1}$$
$$= 2N$$

木の高さ

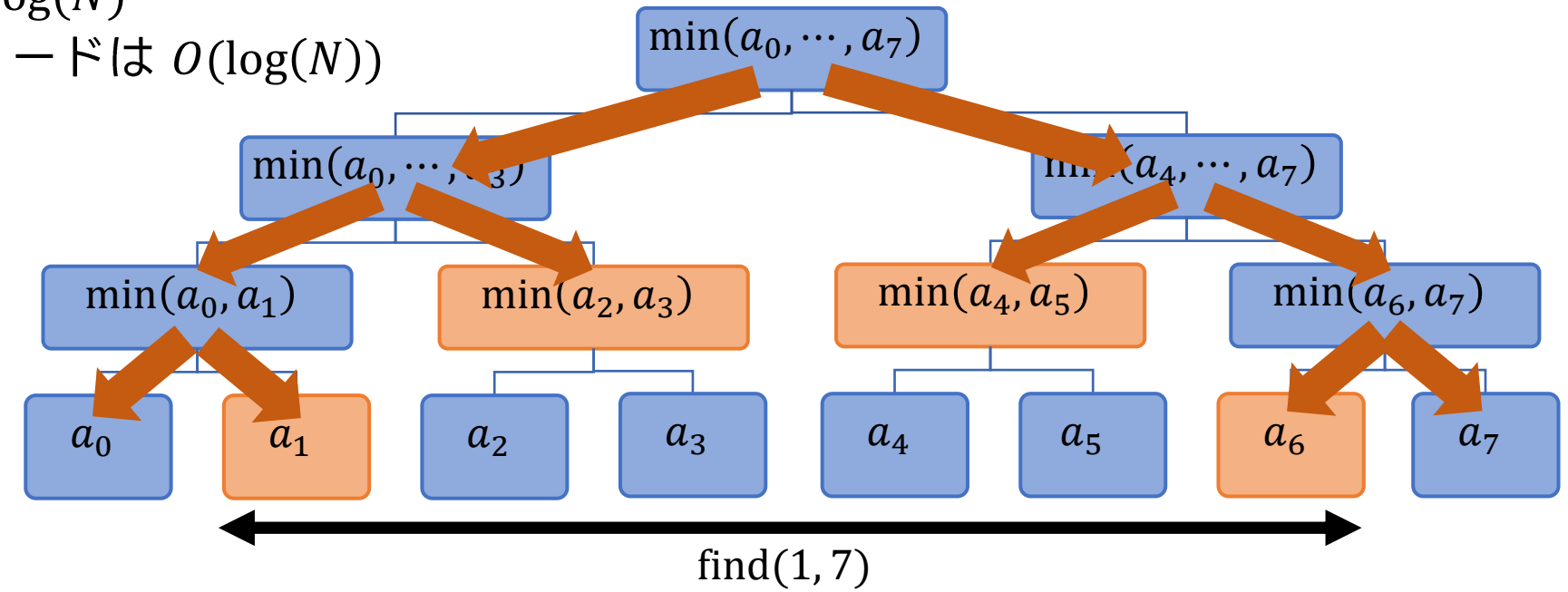
1 段あたりのノード数は親を辿るたびに半分

$$N \times \frac{1}{2^k} = 1$$
$$k = \log(N)$$

おきもち (2/3)

findクエリ

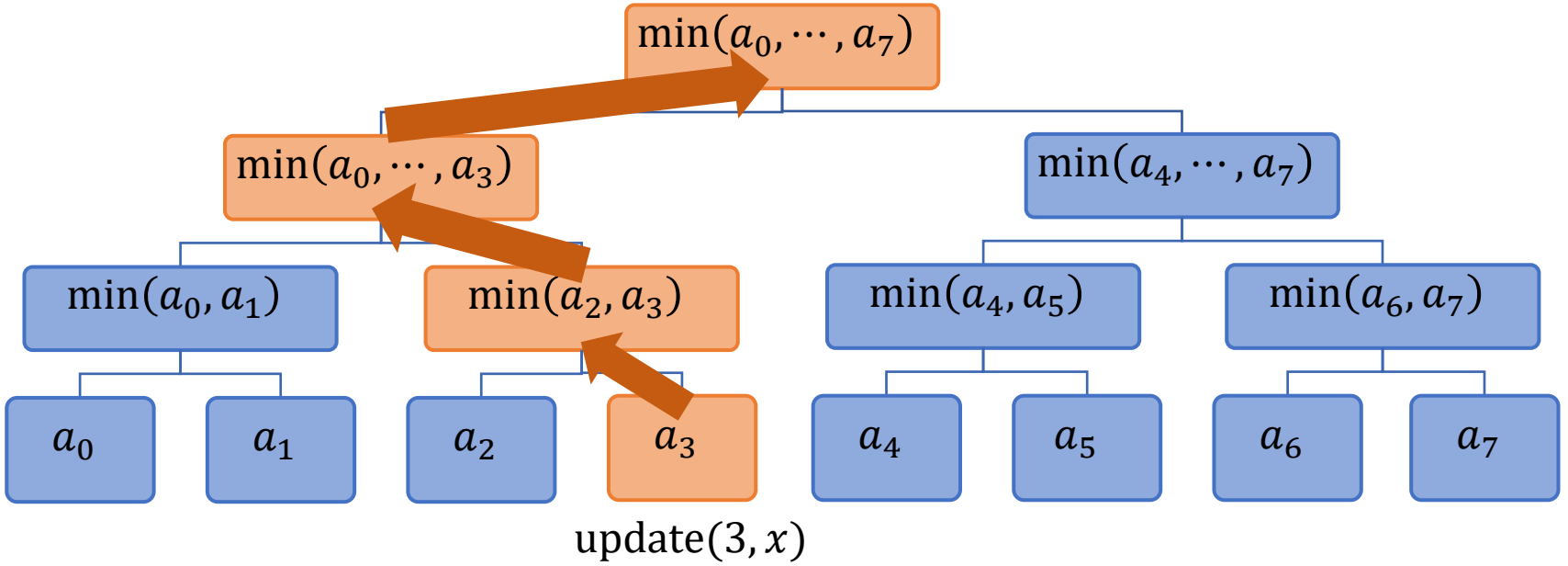
- 根から辿って対象の区間を探していく
  - 各段で対象区間は高々 2 個
  - 木の高さは $\log(N)$
  - 全体で見るノードは $O(\log(N))$



おきもち (3/3)

updateクエリ

- 更新したい場所から親を辿って更新していく
  - 葉から順番に更新

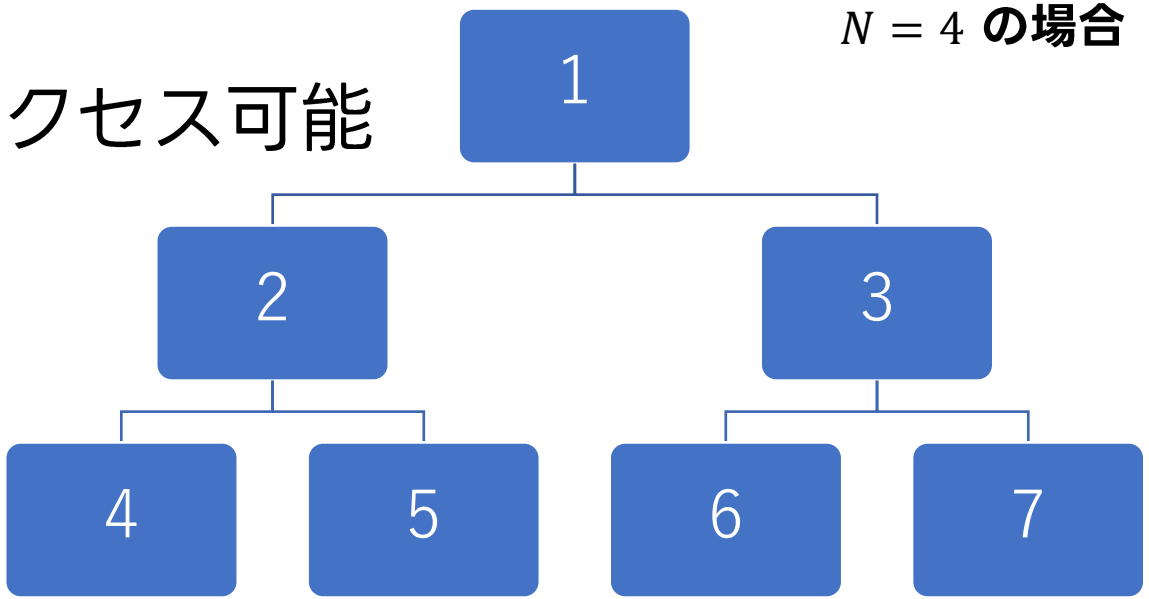


実装方針

どのようにして二分木を表現するか？
→ 1 次元配列で二分木全体を管理

- 要素数は $2N$
- 配列の i 番目は $\mathrm{a}[i+N]$ でアクセス可能
- i 番目のノードに対して
  - 左の子: $2i$
  - 右の子: $2i+1$
  - 親: $i/2$

$N=4$ の場合



ライブコーディングで実装していきます

抽象化と定数倍高速化

抽象化

- 問題によって求められるクエリは様々
  - i.g. max, gcd, lcm, etc…
- 配列の他に関数を渡して初期化できるとうれしい

定数倍高速化

- 再帰は重いので非再帰になるとうれしい
- 要素数を 2 冪にするのも無駄が多い

抽象化

- Segment Tree に限らずデータ構造の抽象化は便利
  - 様々な問題を同じライブラリで解けるようになる

- 初期化時に二項演算, 単位元, 配列を受け取る
- 二項演算の渡し方
  - Python: 関数をそのまま引数に渡せる
  - C++: ラムダ式が便利

- 二項演算や単位元を定義したクラスを利用することも
  - 例) https://noshi91.github.io/Library/data_structure/segment_tree.cpp.html

Segment Tree が扱える演算（再掲）

Segment Tree はモノイドの列を扱うことができる

<u>モノイドの例</u>

- 四則演算
- min, max
- 最大公約数, 最小公倍数
- 一次関数の合成
  - $f(x) = ax + b, g(h) = cx + d$ に対して$f\big(g(x)\big) = acx + ad + b$

非再帰化

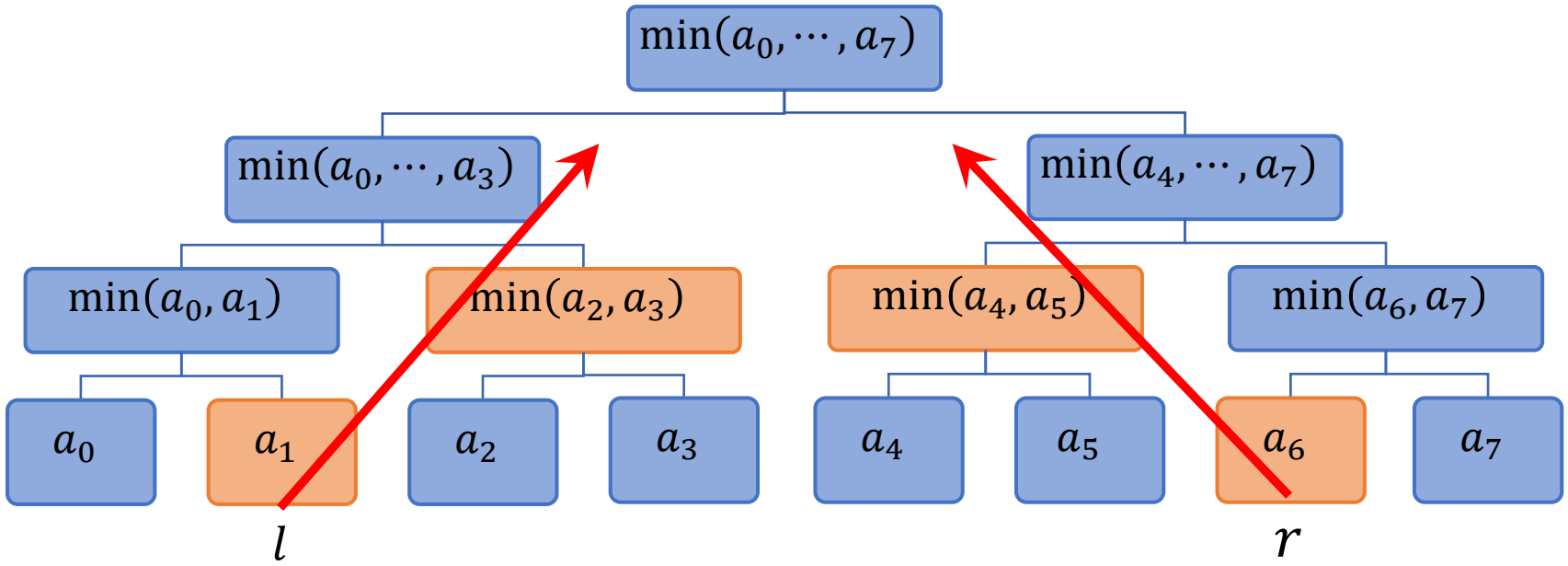
任意の停止する再帰関数は同じ計算量の非再帰で書き換えられる

find クエリの非再帰化

- 再帰関数での実装→トップダウン的に上段のノードから処理
- 非再帰化→下段のノードからボトムアップで処理

非再帰によるfindクエリの実装 (1/2)

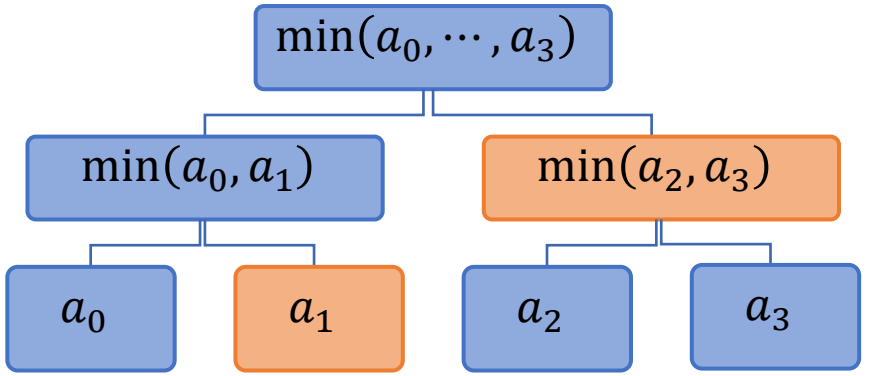
最下段のノードからボトムアップ的に処理



非再帰によるfindクエリの実装 (2/2)

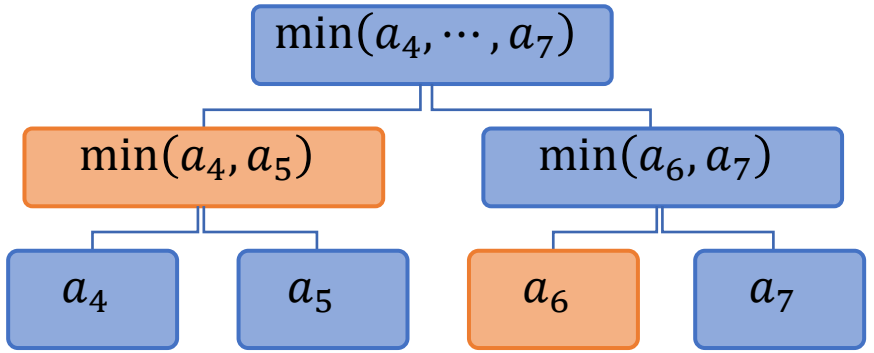
左側（始点側）について

- 右の子: find対象, 次は親の兄弟
- 左の子: 非対象, 次は親



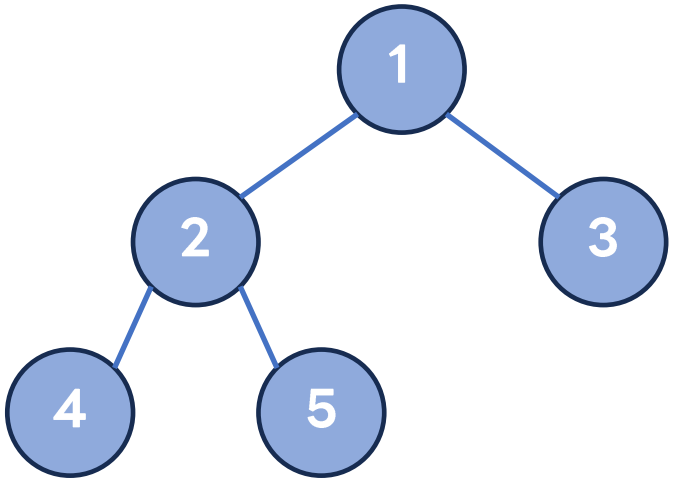
右側（終点側）について

- 右の子: 非対象, 次は親
- 左の子: find対象,次は親の兄弟

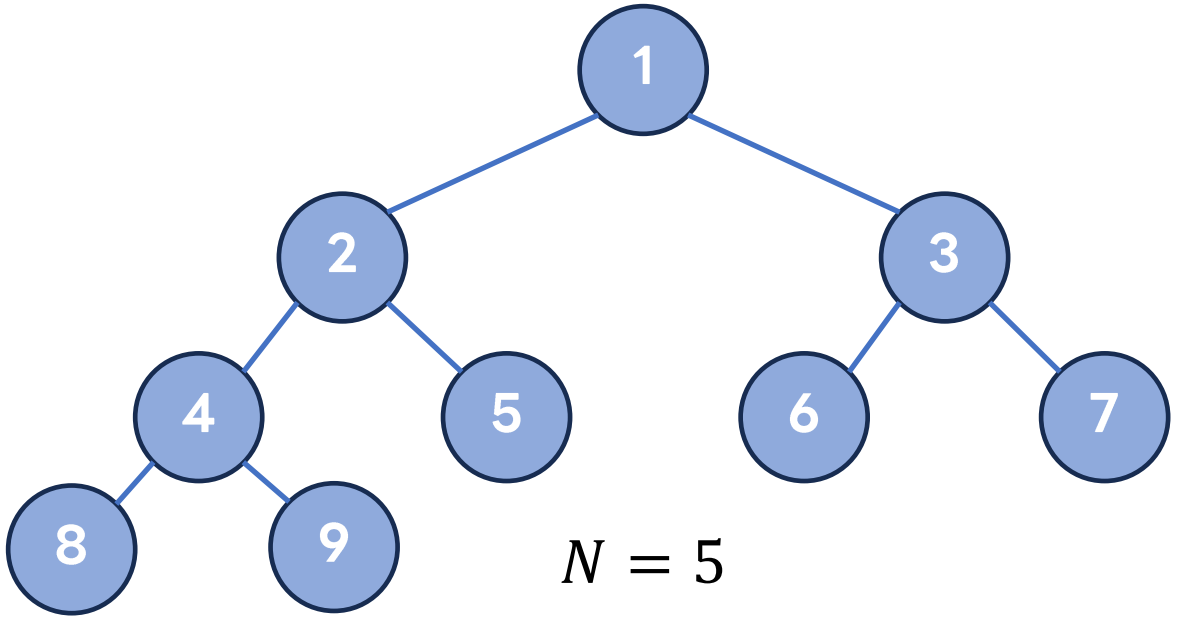


Segment Tree の非2冪実装

結論: 雑に $2N$ の配列に書き変えるだけで動く（非再帰に限る）



$N = 3$



$N = 5$

ぱっと見うまくいきそう（な気がする）

亜種について

Segment Tree の様々な亜種

- Lazy Segment Tree（遅延セグ木）
  - 区間更新・区間取得が可能な Segment Tree
- Dual Segment Tree（双対セグ木）
  - 遅延セグ木の区間更新部分のみを取り出したもの
  - 区間更新・一点取得が可能
- Segment Tree Beats
  - 遅延セグ木を改良したもの
    - 区間chminなどを扱える
  - Angel Beats! が元ネタ

その他

- 動的セグ木
- 永続セグ木
- 2Dセグ木

Segment Tree の例題

スプレッドシートに例題をまとめました

https://docs.google.com/spreadsheets/d/16fTwvnRgsvqLJyQyOk_DZYIfMKwIAJIZ5Pq4bJqXv68/edit?usp=sharing