アルゴリズム勉強会
セグメントツリー編

突然ですが問題です

問題文

長さ N の数列 $A = (A_1, A_2, \ldots, A_N)$ があり、最初はすべての要素が 0 になっています。以下の 2 種類のクエリを処理してください。

- **クエリ 1**：A_{pos} の値を x に更新する。　（**一点更新**）
- **クエリ 2**：$A_l, A_{l+1}, \ldots, A_{r-1}$ の最大値を答える。（**区間取得**）

ただし、与えられるクエリの数は全部で Q 個であるとします。

制約

- 入力はすべて整数である
- $1 \leq N \leq 100000$
- $1 \leq Q \leq 100000$

A58 - RMQ (Range Maximum Queries)
https://atcoder.jp/contests/tessoku-book/tasks/tessoku_book_bf

入力例 1

Copy

```
8 4
1 3 16
2 4 7
1 5 13
2 4 7
```

要素列長:8、クエリ数:4

クエリ(先頭の要素が1なら更新、2なら取得)

- **1** 3 16 : 3番目の要素を16に変更する
- **2** 4 7 : 4-6番目の要素の最大値を出力する

出力例 1

Copy

```
0
13
```

- はじめ、$A = (0, 0, 0, 0, 0, 0, 0, 0)$ です。
- 1 個目のクエリでは、A_3 の値を 16 に更新します。$A = (0, 0, 16, 0, 0, 0, 0, 0)$ となります。
- 2 個目のクエリでは、$(A_4, A_5, A_6) = (0, 0, 0)$ の最大値 0 を出力します。
- 3 個目のクエリでは、A_5 の値を 13 に更新します。$A = (0, 0, 16, 0, 13, 0, 0, 0)$ となります。
- 4 個目のクエリでは、$(A_4, A_5, A_6) = (0, 13, 0)$ の最大値 13 を出力します。

愚直な実装例

僕「普通に一点更新して、普通に区間取得すればええやん」

↓自分がテキトーに作ったクソコード

```
N, Q = map(int, input().split())
Query = [list(map(int, input().split())) for _ in range(Q)]
lst = [0]*N

for x,y,z in Query:
  if x == 1:
    lst[y-1] = z
  elif x == 2:
    print(max(lst[y-1:z-1]))
```

クエリ（先頭の要素が1なら更新、2なら取得）

- **1** 3 16：3番目の要素を16に変更する
- **2** 4 7：4-6番目の要素の最大値を出力する

愚直な実装例：結果

ジャッジ結果

セット名	Sample	All
得点 / 配点	0 / 0	0 / 1000
結果	AC × 1	AC × 16 TLE × 3

TLE（Time Limit Exceeded）

指定された実行時間以内にプログラムが終了しなかった

愚直な実装例：原因

愚直な実装：1クエリごとに一点更新O(1)、区間取得O(N)

forループでO(Q) →

max関数でO(N) →

```
N, Q = map(int, input().split())
Query = [list(map(int, input().split())) for _ in range(Q)]
lst = [0]*N

for x,y,z in Query:
  if x == 1:
    lst[y-1] = z
  elif x == 2:
    print(max(lst[y-1:z-1]))
```

このプログラムの区間取得にかかる計算量は**O(QN)**となる。
制約として 0 <= N,Q <= 10^5が与えられているため、最大10^10もの計算が必要
1秒間に処理できる計算は10^7 ～ 10^8であるので、計算が間に合わないのも当然

なら、どうするか？

→ セグメントツリーを使う

セグメントツリー

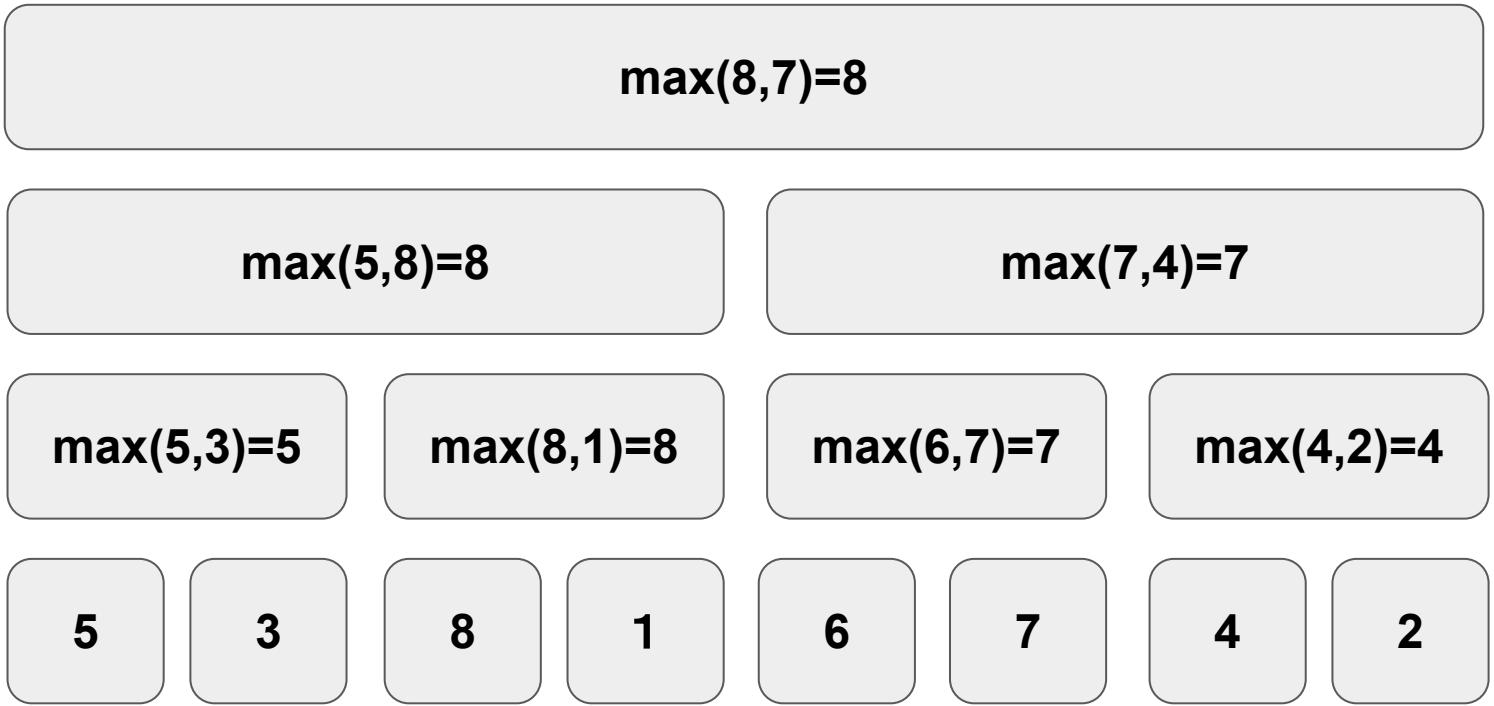
完全二分木で実装された、区間処理に適したデータ構造。具体的には以下の通り

- **配列の一要素を更新　O(logN)**
- **任意区間内の要素に関する特定の演算　(総和、最大値、最小値等) O(logN)**

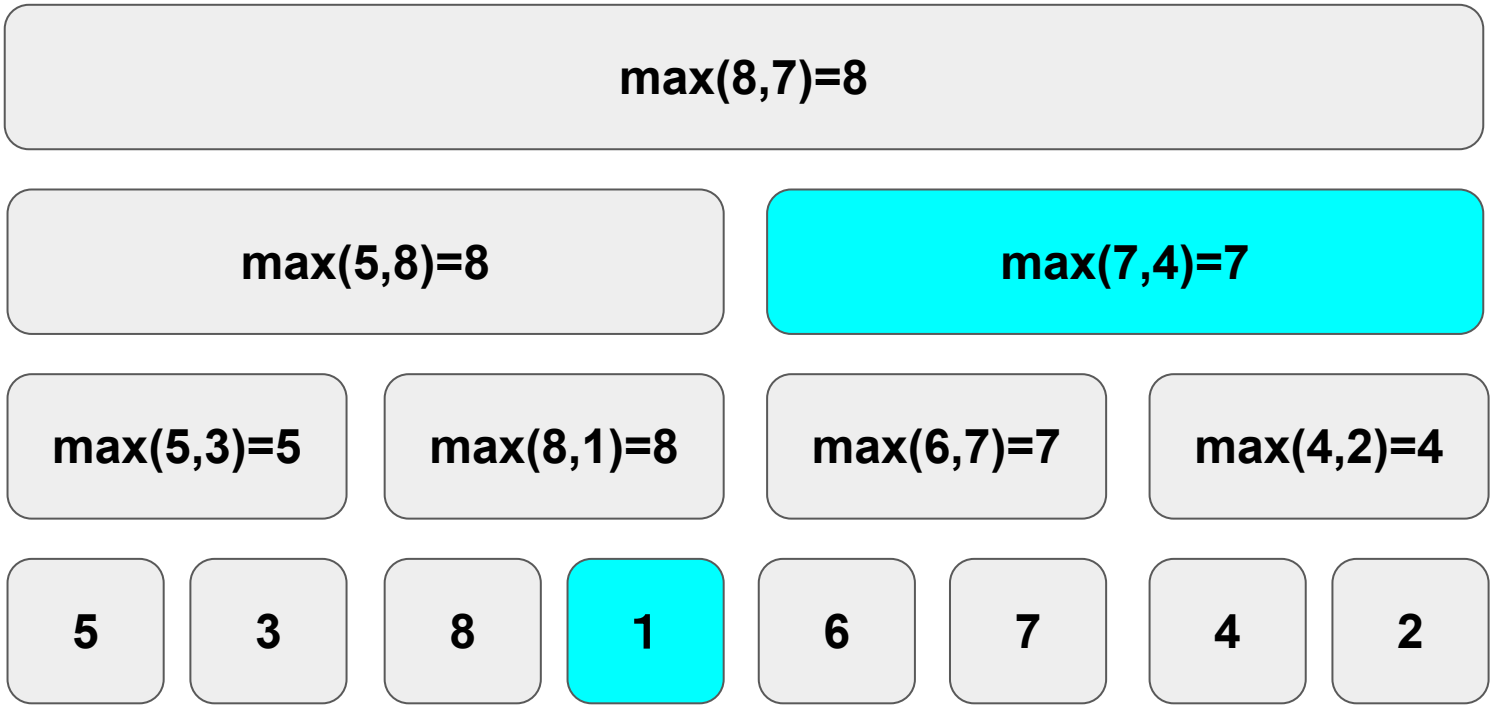
そのため、複数の更新・演算クエリをこなす場合に向いている

セグメントツリー：仕組み

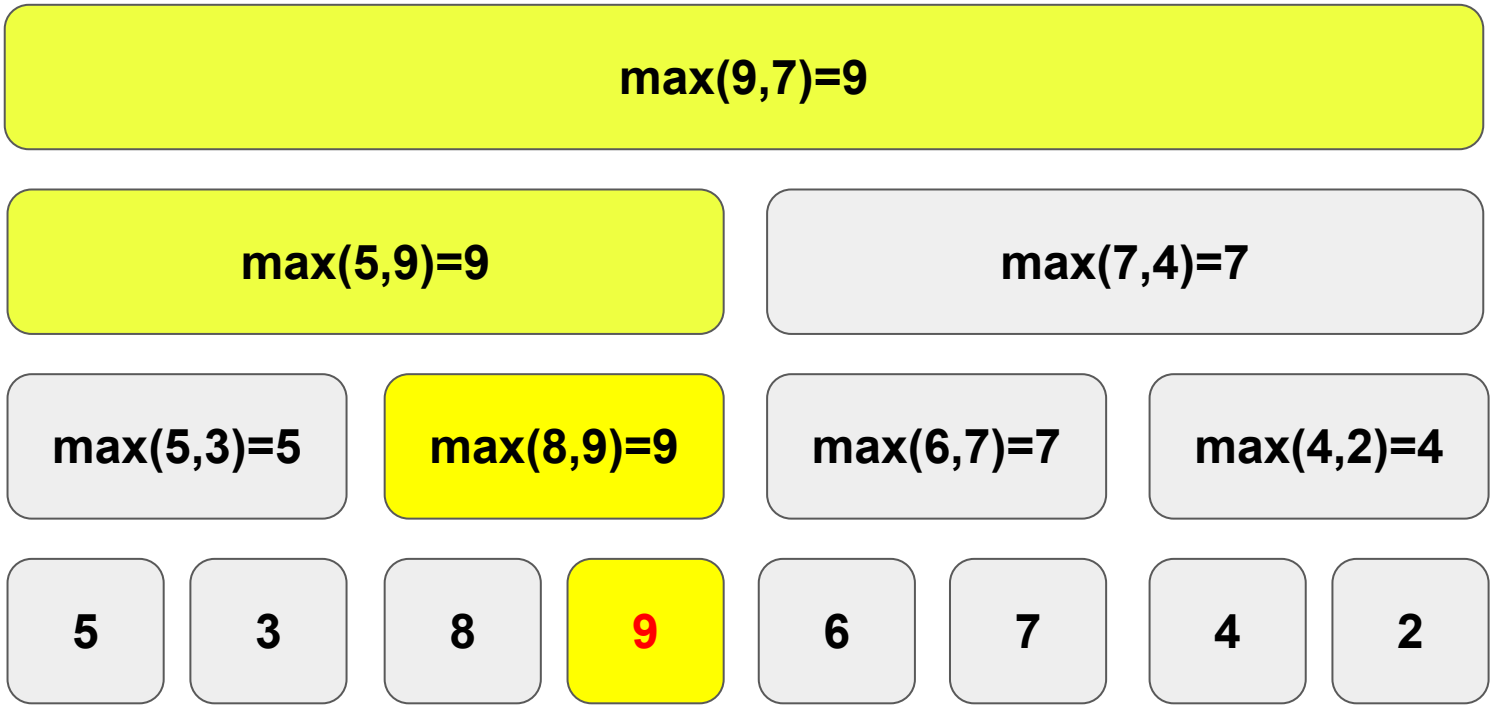
[5, 3, 8, 1, 6, 7, 4, 2]について以下の木構造を考える



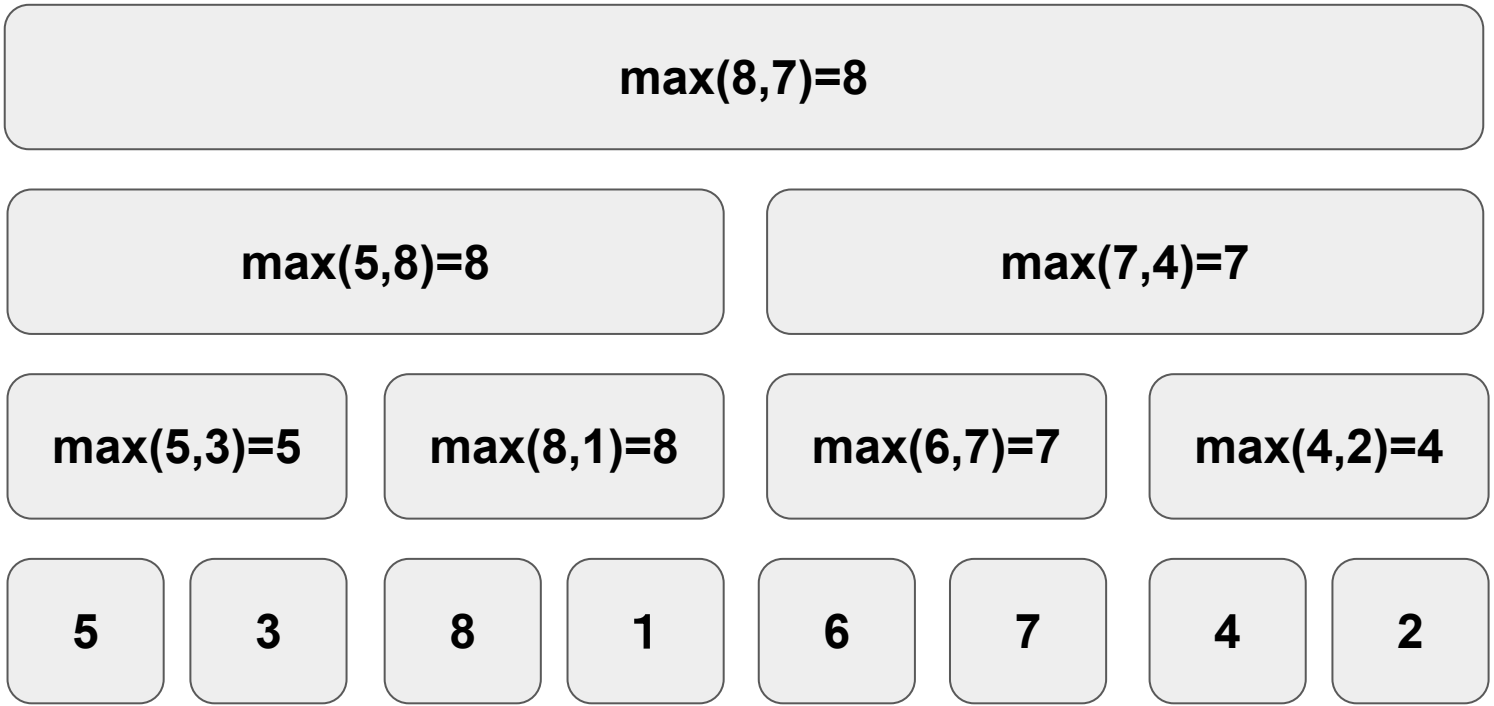
[1, 6, 7, 4, 2]で最大値をとることを考える場合(**区間取得**)
以下の水色部分のmaxを取るだけで良い



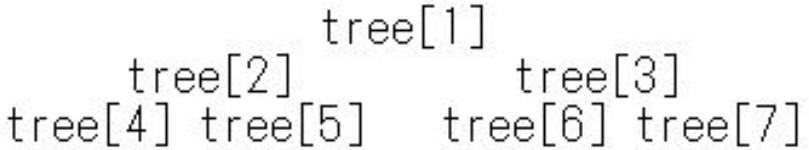
1を9に変更する場合(**一点変更**)
以下の黄色の部分を更新すれば良い



前2つのスライドから
愚直な実装：1クエリごとに一点更新O(1) 区間取得O(N)
セグ木実装：1クエリごとに一点更新 O(logN) 区間取得O(logN)
であることが分かる



セグメントツリー：実装



実装の際は、セグメント木は2*Nのリストによって管理される。
index = 0 は使用しない, 全要素の積はtree[1]に入っている。
子は2*i と 2*i+1、親はi>>1で参照することができる。

葉部分に配列の値をセットし、関数を適用しながら木を構築。更新時は葉部分から上方向に更新する。

タスクに応じて(初期値と)関数と単位元を設定する必要がある
※今回は最大値問題であったが、その他の演算でもセグ木は活用できる
（詳しい話は「モノイド」あたりでググってください）

セグメントツリー：提出・結果

ジャッジ結果

セット名	Sample	All
得点 / 配点	0 / 0	1000 / 1000
結果	AC × 1	AC × 19

愚直な実装：1クエリごとに一点更新O(1) 区間取得O(N)

セグ木実装：1クエリごとに一点更新O(logN) 区間取得O(logN)

全体の計算量はO(QN) → O(QlogN)に抑えられた！

無事、TLEの壁を乗り越えAC(Accept) 👊

制約

- 入力はすべて整数である
- $1 \leq N \leq 100000$
- $1 \leq Q \leq 100000$

（補足）遅延セグメントツリー

セグメントツリーは一点更新と区間取得がO(logN)で可能。

当然の疑問：**区間更新もO(logN)で出来ないのか？**

→ 遅延セグメントツリーという構造を用いる

（遅延配列の導入が必要になる、複雑になるのでここでは割愛）

(参考)

https://smijake3.hatenablog.com/entry/2018/11/03/100133

まとめ

セグメントツリーとは？

完全二分木で実装された、区間処理に適したデータ構造。

- **配列の一要素を更新　O(logN)**
- **任意区間内の要素に関する特定の演算　(総和、最大値、最小値等) O(logN)**

一点（区間）更新と区間取得のクエリが大量に飛んでくるケースに向いている