魚2 (Fish2)

解説　nxteru (戸高空)

問題概要

- N匹の魚が横一列に並んでいる
- 左からi番目の魚の大きさはA_iである
- Q個のクエリ
- $T_j = 1$　　X_j番目の魚の大きさをY_jにする
- $T_j = 2$　　L_jからR_j番目までの魚を取りだして以下の問題を解く

問題概要

- ある隣合う2匹の魚で大きい方が小さい方を食べて、食べた魚の大きさ分だけ大きくなるということが残り1匹になるまで繰り返される
- 最後の1匹として残る可能性がある魚の数を求めよ
- $N, Q \leq 100000$、魚の大きさ$\leq 10^9$

簡単な考察

- ある魚sが残ることができるかを考える
- このときsが隣り合う魚を食べる以外は考えなくてよい
- またsが隣り合う魚を食べられる場合は左右どちらでも食べてよい

- よって「sが隣り合う魚を食べられるなら食べる」を繰り返してすべての魚を食べられるかが分かればよい

小課題1

- $N, Q \leq 500$
- すべての魚についてその魚が残ることができるかを調べる
- 各魚$O(N)$でシミュレーションできるのでクエリ当たり$O(N^2)$で解ける
- $O(QN^2)$

小課題2

- $Q = 1, T_j = 2, L_j = 1, R_j = N$
- クエリが1回
- シミュレーションを高速化する

小課題2

- 右に向かってできるだけ食べることを考えると，今の大きさ以下の魚が連続していたら、その魚たちは大きさの加算分を計算しなくても食べられることが分かる
- 初めて計算が必要になるのは今の大きさより大きい魚が初めて現れる場所
- そのような場所は例えばsegment tree上の二分探索で$O(logN)$で求められる

小課題2

- そのような右に自分より大きな魚が初めてでてくる場所の手前まで食べてしまい，その合計だけ大きさの加算する
- このように右に進めるだけ進む、左に進めるまで進むを繰り返してシミュレーションする
- この左右に伸ばす回数は$\log_2 \max\{A\}$回で抑えられる
- 前の自分より大きな魚を食べると大きさが2倍以上になるためである
- O（$N \log A \log N$ ）

考察

- シミュレーションにおいて状態は$[l, r]$…「l～r番目の魚を食べて今の大きさは$A_l + A_{l+1} + \cdots + A_r$であり左隣が$l-1$、右隣が$r+1$である」というような区間であらわされる

- クエリは$[i, i]$からスタートして$[L, R]$に到達できるようなiはいくつあるかを答える問題である

考察

- ここで左にも右にも進めない状態、つまり

  $A_l + A_{l+1} + \cdots + A_r < A_{r+1}$ かつ

  $A_l + A_{l+1} + \cdots + A_r < A_{l-1}$

  を満たす区間を閉塞区間とする

考察

- 実は「ある魚が区間$[L, R]$に到達できる⇔その魚が$[L, R]$の範囲のどの閉塞区間にも含まれていない」が成り立つ

- 閉塞区間$[l, r]$に含まれている⇒魚$l-1$,魚$r+1$を食べることができない
- 閉塞区間に含まれていない⇒今の状態が閉塞区間でないなら必ず左右のどちらかに区間を伸ばせる．閉塞区間に到達することはないので、これを繰り返すと$[L, R]$に到達できる

イメージ（時間があれば追加する）

- 時間がありませんでした

小課題3

- この閉塞区間を求める
- 閉塞区間$[l, r]$に関して

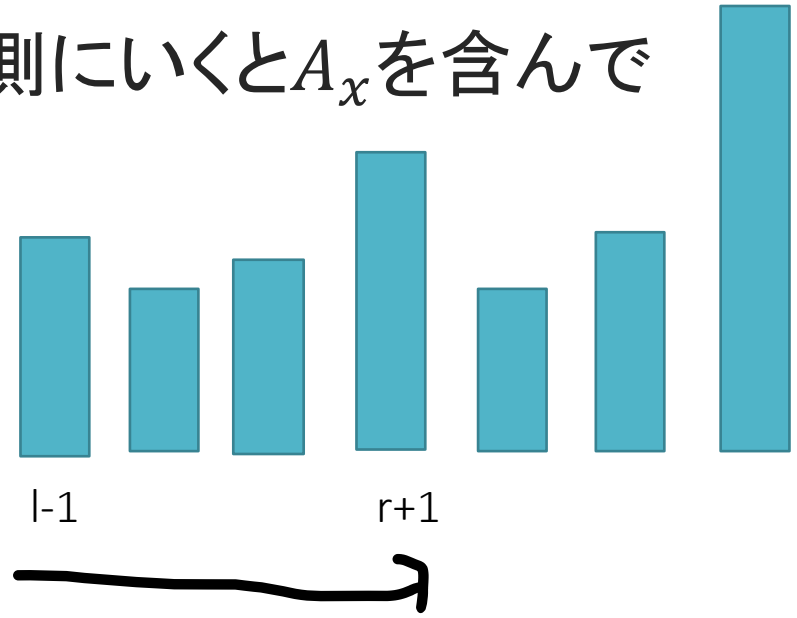
  $A_l, A_{l+1}, \ldots, A_r < A_{r+1}$

  $A_l, A_{l+1}, \ldots, A_r < A_{l-1}$

が成り立つがこのような$l-1, r+1$の組は$O(N)$個しかない

小課題3

- $A_{l-1} < A_{r+1}$ であるような組を考える
- $l-1$を固定したとき$r+1$として考えられるのは$A_{l-1} < A_x$を初めて満たすxのみである
- なぜなら$r+1$がそのようなxより右側にいくとA_xを含んで$A_l, A_{l+1}, \ldots, A_r < A_{l-1}$を満たさなくなる



小課題3

- 同様に$A_{l-1} \geq A_{r+1}$であるような組についてもそれぞれの右端について左端の候補は1つである
- これらの条件を満たす組はstackを用いて左から列を走査することで$O(N)$ですべて求めることができる

小課題3

- A_iを見るとき、stackのtopがA_i以下の間stackをpopし続ける
- そしてA_iをstackに入れる

- このpopのときpopされる側にとってA_iが右側で初めて自分以上の場所
- A_iをstackに入れるときのstackのtopがiにとって左側で初めて自分より大きくなる場所

小課題3

- クエリあたり$O(N)$で閉塞区間を求めて，それらの区間に含まれていない魚の数を求められる
- $O(QN)$

閉塞区間についての考察

- 閉塞区間同士は交差せず内包するか重ならないかのどちらかである

- 右下のように$[a+1, b-1]$と$[c+1, d-1]$が交差したとすると
- cが閉塞区間$[a+1, b-1]$に含まれるので$c < b$
- bが閉塞区間$[c+1, d-1]$に含まれるので$b < c$
- で矛盾

小課題4

- $T_j = 2$（更新クエリがない）

- 最初に閉塞区間を求めておくと更新がないのでそれらを使うことができる

小課題4

- クエリ$[L, R]$にこたえる

- 範囲$[L, R]$のみを考えたとき$L-1$に進めなくなるため，$L-1$を左の壁とした新たな閉塞区間ができることがある
- 同様に$R+1$を右の壁とした閉塞区間ができる

小課題4

- そのような$L-1$を左の壁とする閉塞区間にどこまで含まれるか、つまり右の壁として条件を満たす一番右のものを求める
- その条件は$sum[i]$を$A_1+A_2+\cdots+A_{i-1}$として

$sum[x]-sum[L]<A_x \Leftrightarrow \quad -sum[L]<A_x-sum[x]$

であるxだから$A_x-sum[x]$の配列で$-sum[L]$以上のものでR以下もっとも右を求められればいい

Segment tree上の二分探索で$O(logN)$

小課題4

- こうして新たにできた閉塞区間に含まれる左右の部分を除いた範囲をあらたに$[L, R]$としてクエリにこたえる
- すると$[1, N]$の範囲で求めた最初からある閉塞区間のみを考えて、$[L, R]$の範囲でそれに含まれない場所の数を求める問題になる

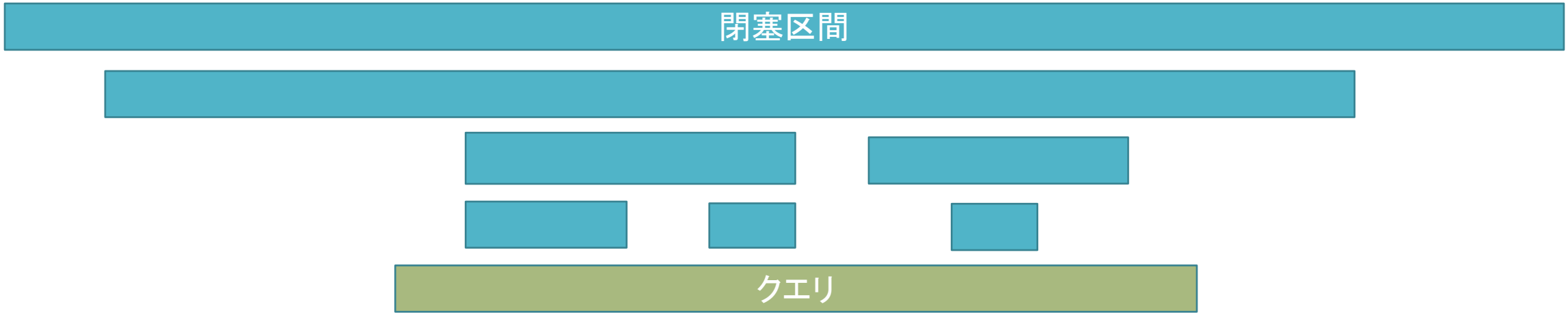
小課題4

- もう一つのうれしい性質
- この新たなクエリの区間と閉塞区間は交差せず、内包するか重ならないかのどちらかである

- 例えば図のように交差しているとすると、$A_{a+1} + \cdots + A_{b-1}$は$A_a$以下だからクエリの$a+1$以降の部分はさきほどの処理で除かれているはずである

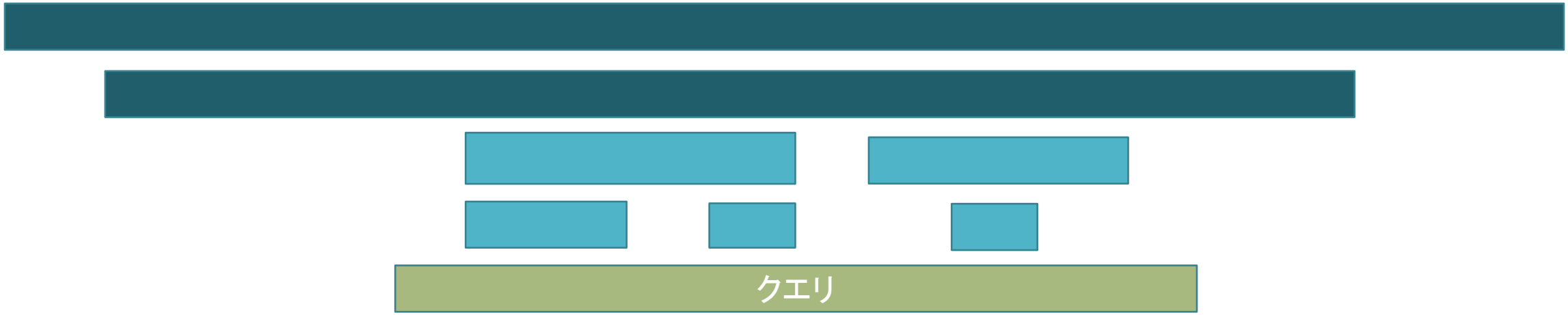
小課題4

- 閉塞区間がクエリを含むか、クエリが閉塞区間を含むという関係



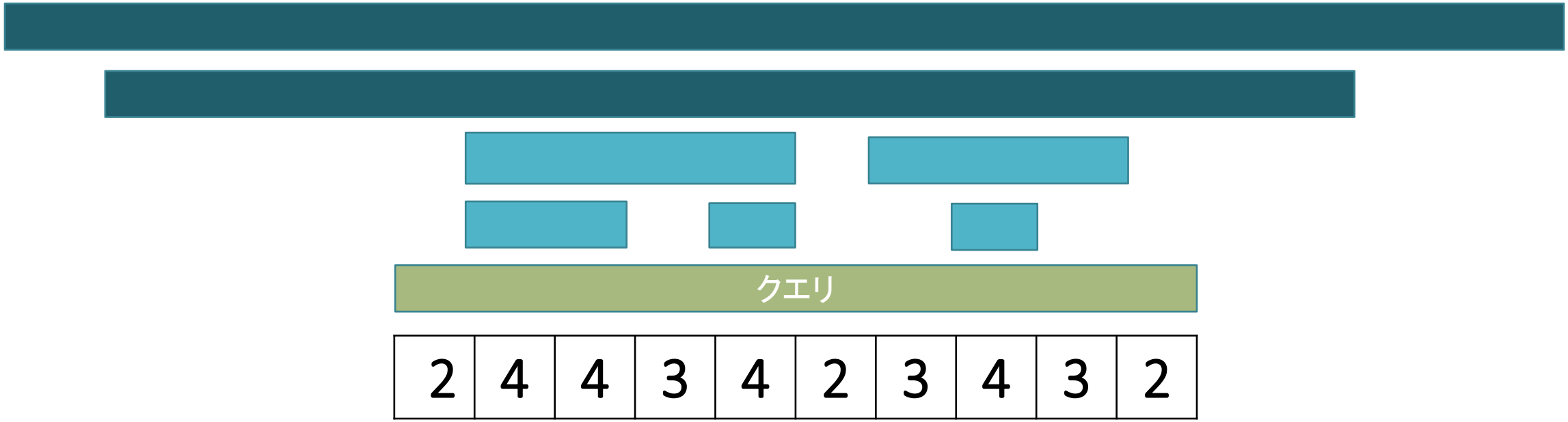
小課題4

- 閉塞区間がクエリを含むか、クエリが閉塞区間を含むという関係
- クエリが含む閉塞区間のみを考えたときクエリの範囲でそれらに含まれない場所の数を求めたい



小課題4

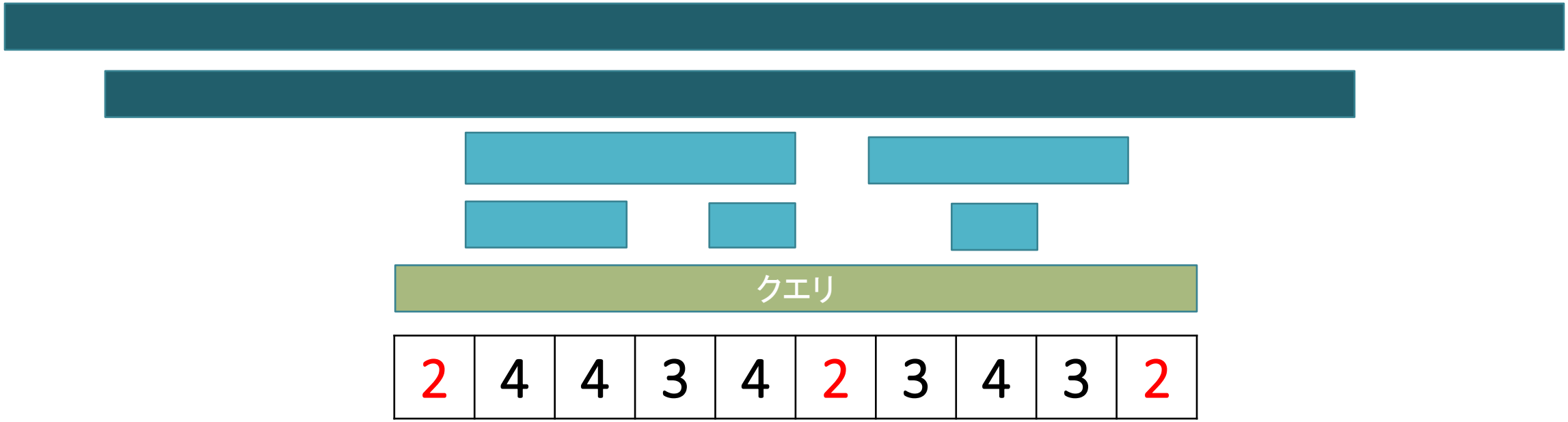
- それぞれの場所がいくつの閉塞区間に含まれているかを考える



小課題4

- クエリが含む閉塞区間のみを考えたときそれらに含まれない場所

⇔その場所が含む閉塞区間の数がクエリの範囲内で最小＝（クエリを丸ごと含む閉塞区間の数）



小課題4

- クエリは以下の問題に帰着
- B_i …場所iが含む閉塞区間の数の配列

- クエリ$[L, R]$
- $\min\{B_L, B_{L+1}, \ldots, B_R\} = B_i$を満たす$i$の個数$(L \leq i \leq R)$を求める

小課題4

- クエリ$[L, R]$
- $\min\{B_L, B_{L+1}, \ldots, B_R\} = B_i$を満たす$i$の個数$(L \leq i \leq R)$を求める

- これはsegment treeでできる

小課題4

- 各ノードで持つ情報は最小値miと最小値の個数cnt
- ノードAとノードBのマージは

$$mi = \min(A.mi, B.mi),\ cnt = 0;$$
$$If(mi == A.mi)cnt \mathrel{+}= A.cnt$$
$$If(mi == B.mi)cnt \mathrel{+}= B.cnt$$

小課題5

- 小課題5は満点とやることはほとんど同じなので省略します
- ただクエリが$[1, N]$のみなので満点と比べて簡単にできるたり省略できたりします

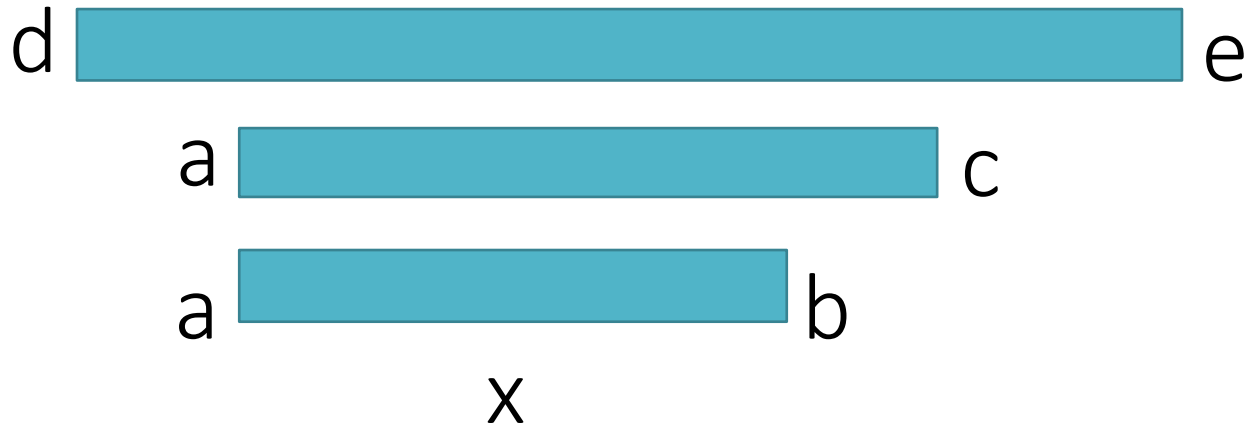
満点

- 更新あり
- 答えを求めるクエリは小課題4と同じようにする
- そのために配列B_i …場所iが含む閉塞区間の数を適切に更新する必要がある

閉塞区間についての考察2

- ある場所xを含む閉塞区間の数は最大で$\log_2 \max\{A\}$個である（同様に区間の重なりの深さも$\log_2 \max\{A\}$個以下である）
- 右下のように重なっているとすると
- $x < \min(a, b)$
- $x + b(> 2x) < \min(a, c)$
- $x + b + c(> 4x) < \min(d, e)$

左右の壁の下限が2倍になっていく



満点

- 場所xを含む閉塞区間の数は$O(logA)$個
- 値が更新されたときに消えたり増えたりする閉塞区間の数も$O(logA)$個である

- 方針としてはその時点での閉塞区間をすべて管理する

満点

- まずは配列B_iの区間最小値個数のsegment treeに閉塞区間追加、削除の機能=区間加算クエリの機能を追加する

- これは遅延segment treeにすればできる

満点

- A_xをyに更新するクエリ
- ①場所xを含む閉塞区間をすべて削除する（$Olog(A)$個）
- ② A_xをyに更新
- ③場所xを含む閉塞区間をすべて見つける（$Olog(A)$個）
- ④見つけた閉塞区間を追加する（$Olog(A)$個）

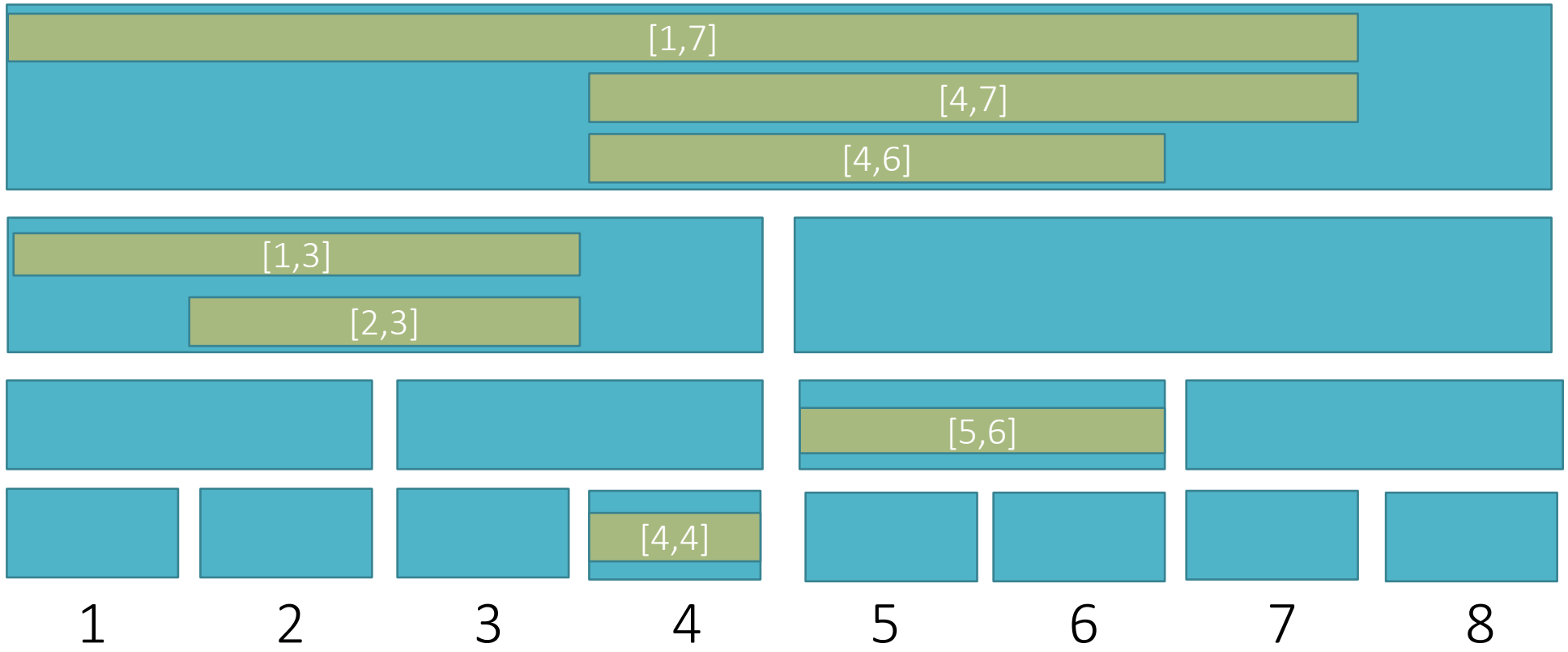
区間を管理するデータ構造

- segment treeに区間を入れていく
- 区間$[l, r]$はその区間を完全に格納できるノードのうち最も小さいものに入れる.

区間を管理するデータ構造

- 管理する区間は交差しないので、各ノードではそれが対応する範囲を$[l,r]$として$(l+r)/2$を含むような区間が入っていく
- それらの格納される区間たちは区間の大きい順に次に大きい区間を内包するというように内包の関係が$(l+r)/2$を中心に一列になっている
- よって各ノードではstackを持ってそこに区間を小さい順に積んでいく
- 区間の追加や削除はstackの一番上のみで行う

区間を管理するデータ構造



区間を管理するデータ構造

- ①場所xを含む閉塞区間をすべて削除する
- Segment treeの葉ノードxから順番に上に上りつつ、各ノードではstackの一番上(一番大きな区間)がxを含む間それらをpopする
- $O(logA + logN)$でできる

区間を管理するデータ構造

- ④見つけた$O(logA)$個の区間を追加する

- 追加する区間を小さい順に持っておく
- Segment treeの葉ノードxから順番に上に上りつつ、各ノードではまだ追加していない区間で最小のものが追加できる間それをpushする

- $O(logA + logN)$

満点

- ③場所xを含む閉塞区間をすべて見つける（$O(log(A))$個）

満点

- 場所xの右の壁となる候補はA_xより大きい必要があるのでまず$r < i$かつ$A_x < A_r$を満たす最小のrが考えられる
- その右で次の候補を考えると，区間はxとrを含むので$r < r2$かつ$A_x + A_r < A_{r2}$を満たす最小の$r2$
- このように右の候補を求めていくとAの値が2倍になっていくので$O(logA)$個しかない
- また〇$< i$かつ△$< A_i$を満たす最小のiはseg木上の二分探索で$O(logN)$で求まる
- $O(logAlogN)$で右の壁の候補をすべて列挙できる

満点

- 同様に左の壁の候補も$O(logA)$個求められる
- この左の壁の候補×右の壁の候補の組み合わせをすべて試す

- $O(logAlogA)$

細かい話

- 最初に与えられる値も$1,1,\ldots,1$（閉塞区間なし）からi番目をA_iに更新するとしてN個の更新クエリに帰着すると楽

- クエリを縮めるパートは壁の候補を列挙するようにすると、計算量は増えるが、区間を見つけるパートと同じ処理でできる

- 更新クエリで削除、追加するのは更新する場所xが壁になる区間、つまり$x-1, x, x+1$を含む区間も処理する必要がある．壁の候補も$x-1, x+1$から始めて見つけるといい

満点（まとめ）

- A_xをyに更新するクエリ
- ①場所xを含む閉塞区間をすべて削除する($logA$個)

区間の管理$O(logA + logN)$、Bの配列の更新$O(logAlogN)$

- ② A_xをyに更新
- ③場所xを含む閉塞区間をすべて見つける（$logA$個）

$O(logAlogN + logAlogA)$

- ④見つけた閉塞区間を追加する（$logA$個）

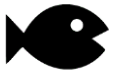
区間の管理$O(logA + logN)$、Bの配列の更新$O(logAlogN)$

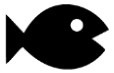
満点（まとめ）

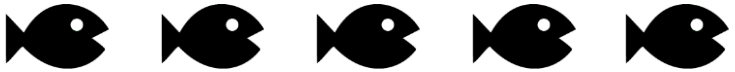
- $[L, R]$にこたえるクエリ
- ⑤左右で新たにできる閉塞区間に含まれる部分を除く$O(logN)$
- ⑥配列B(各場所が含む閉塞区間の数)で区間最小値の個数を求める$O(logN)$

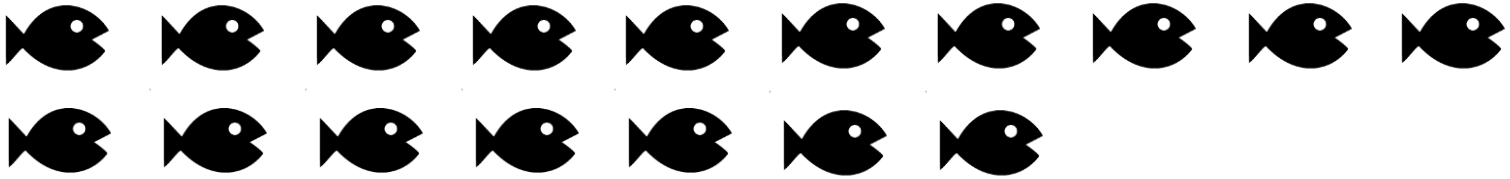
- 全体で$O((N+Q)(logNlogA+logAlogA))$

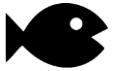
得点分布

48点 

36点 

25点 

13点 

8点 

5点 

0点