蟻本 p.158 column

Sparse Table

by **Suu0313**

目次

やりたいこと

区間に関するクエリに高速に答えたい！

やりたいこと

区間に関するクエリに高速に答えたい！

↑それセグ木でいいじゃん

やりたいこと

区間に関するクエリに高速に答えたい！

↑それセグ木でいいじゃん

だいたいそう

Sparse Table のメリット

- 高速　　区間取得が $\Theta(1)$

Sparse Table のメリット

- 高速　　区間取得が $\Theta(1)$

↑使う理由 is だいたいこれ

$\log$ は定数ではないので落とせると嬉しい

Sparse Table のメリット

- 単位元が必要ない (結合則と冪等性が必要)

Sparse Table のメリット

- 単位元が必要ない (結合則と冪等性が必要)

所謂冪等半群の処理ができる

結合則 : $(a * b) * c = a * (b * c)$ 演算順序を入れ替えられる

冪等性 : $f(x) = f(f(x))$ 同じ操作を繰り返しても結果が同じ

よくあるやつ : 区間の $\min$, $\max$, $\gcd$ など

デメリット

- セグ木のような要素の更新ができない
- 累積和の強化版のようなイメージだが、冪等性が必要なので区間和ができない
- 単位元をつけないので区間長 0 のときの処理に困る

ということで、

Sparse Table とは

結合則と冪等性が成り立つ操作を 前計算 $\Theta(n \log n)$, クエリ $\Theta(1)$

でることができるデータ構造 です

データの持ち方と実装

ここからのインデックスはすべて 0-based-index です

データの持ち方と実装

長さ n の区間に対する演算を考えます

区間内のインデックス i から長さ 2^k ごとの演算結果を持つテーブルを作る ($0 \leq i < n, \quad 0 \leq k, \quad i + 2^k \leq n$)

このテーブルの生成が $\Theta(n \log n)$ (後で説明します)
区間長 2^k のクエリに $\Theta(1)$ で答えられるようになった

クエリの求め方

クエリ l, r : 区間 $[l, r)$ の演算結果を求めなさい

$$a_l * a_{l+1} * \cdots * a_{r-2} * a_{r-1}$$

クエリの求め方

これを、$[l, l+2^k)$ のクエリ と $[r-2^k, r)$ のクエリに分割してみる

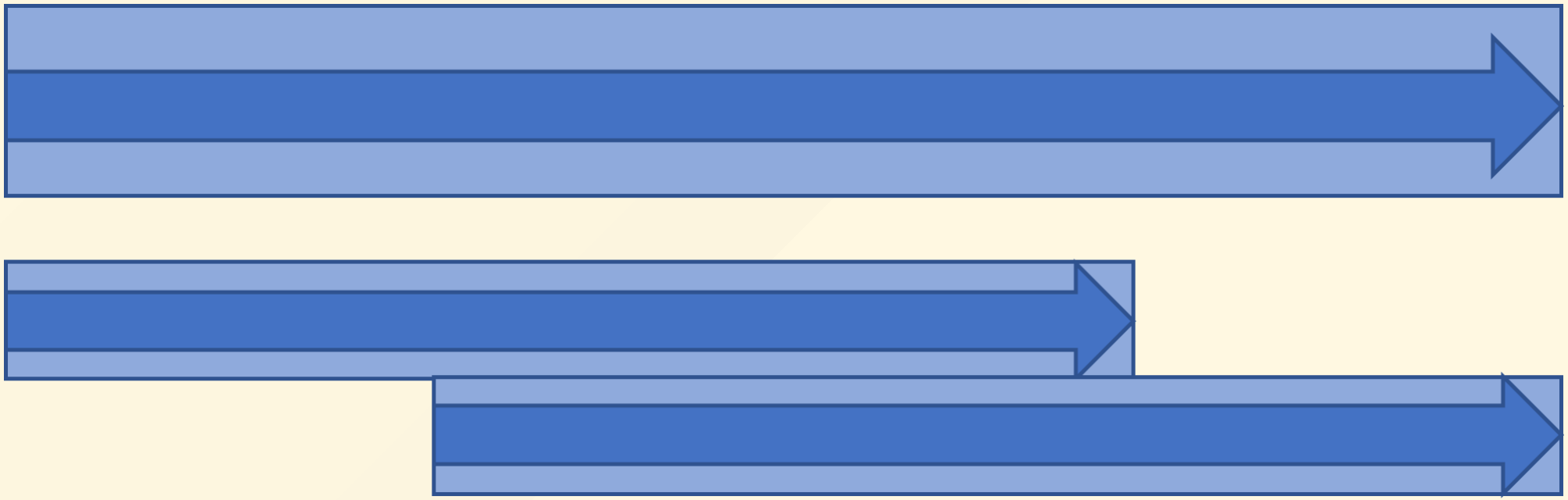
ただし、k は $2^k \leq r-l$ となる最大の k

$$a_l * a_{l+1} * \cdots * a_{l+2^k-2} * a_{l+2-k-1}$$

$$a_{r-2^k} * a_{r-2^k+1} * \cdots * a_{r-2} * a_{r-1}$$

すると、どちらの区間も元の区間の半分以上を覆うことができる

(証明は省きますが、半分に満たなかったらもっと k を大きくできそうですよね？)



$[l, l + 2^k)$ のクエリ と $[r - 2^k, r)$ のクエリ は作っておいたテーブルを見れば書いてあるはずなので、それぞれ $\Theta(1)$ でわかります

それぞれの演算結果をマージすることで、クエリの答えがわかります

このときに、

- 計算順序を入れ替えるために結合則
- 区間が重なっても大丈夫なために冪等性

が必要になってきます

定数個の演算なので計算自体は $\Theta(1)$ で終わりそうに見えますが、

ネックになりそうなのは $2^k \leq r - l$ となる最大の k を求めるパートです

普通に計算しようとすると$\Theta(\log n)$ ほどかかってしまいそうな気持になりますが、これも前計算してしまいましょう

「区間の長さ」は n 通りほどしかないので、$\Theta(n)$ で前計算できそうです

幂等半群の要素列 $\{a_n\}$ の区間に対して演算 f を行った結果が欲しいとします

持たせるデータを

$\mathrm{table_{ij}} =$ 区間 $[j, j+2^i)$ の演算結果
$\mathrm{lookup_i} = 2^\mathrm{k} \leq \mathrm{i}$ となる最大の k とすると

初期化を

$$\mathrm{table}_{0,\mathrm{j}} = a_j$$
$$\mathrm{lookup}_1 = 0$$

というようにして、

$$\mathrm{table}_{\mathrm{i},\mathrm{j}} = f(\mathrm{table}_{\mathrm{i}-1,\mathrm{j}}, \mathrm{table}_{\mathrm{i}-1,\mathrm{j}+2^{\mathrm{i}-1}}) \quad (\mathrm{i} > 0)$$
$$\mathrm{lookup}_{\mathrm{i}} = \mathrm{loolup}_{\lfloor \mathrm{i}/2 \rfloor} + 1 \quad (\mathrm{i} > 1)$$

というように計算をすることができます

疑似コード

```
for j in [0, n) : table[0][j] = a[j]

for i in [1, k) :
  for j in [0, n-(1<<i)] :
    table[i][j] = f( table[i-1][j], table[i-1][j+(1<<(i-1))] )

lookup[1] = 0

for i in [2, n] : lookup[i] = lookup[i>>1] + 1
```

クエリ l, r を求めるときは

$b = \mathrm{lookup}_{r-l}$ として

$$\mathrm{res} = \mathrm{f}(\mathrm{table}_{b,l}, \mathrm{table}_{b,r-2^b})$$

となります

```
b = lookup[r-l]
return f( table[b][l], table[b][r - (1<<b)] )
```

tableの大きさは $(k+1) \times n$ (k は $2^k \leq n$ となる最大の k)

となりますが、この k を求めるには

- 愚直に求める $\Theta(\log n)$
- フッフッフw 伝家の宝刀 †二分探索† $\Theta(\log \log n)$
- 組み込み関数を使う (gccなら __builtin_clz で $32-$ __builtin_clz(n) とするなど)

といった方法があります

例題:

ABC189-C - Mandarin Orange

問題概要

数列 $\{A_N\}$ に対し、適切に $0 \leq l < r < N$ を選んだ時の

$(r-l) \cdot \min_{l \leq i < r} \{A_i\}$ の最大値を求めよ

制約
$1 \leq N \leq 10^4$
$1 \leq A_i \leq 10^5$

思考

$1 \leq N \leq 10^4$ だから **C++なので** $\Theta(N^2)$ が間に合いそう…？

区間minが欲しいな～ うーん、セグ木！

思考

$1 \leq N \leq 10^4$ だから **C++なので** $\Theta(N^2)$ が間に合いそう…？

区間minが欲しいな～ うーん、セグ木！

ん？

思考

$1 \leq N \leq 10^4$ だから **C++なので** $\Theta(N^2)$ が間に合いそう…？

区間minが欲しいな～ うーん、セグ木！

ん？

実行時間制限: 1.5 sec

いくらC++でも $\Theta(N^2 \log N)$ は無理かもしれない…

そんなあなたに Sparse Table！

区間minが $\Theta(1)$ でできる

構築が $\Theta(N \log N)$ で 計算に $\Theta(N^2)$

ギリギリ間に合いそう！！

間に合いました

https://atcoder.jp/contests/abc189/submissions/20469109

実行時間: 182 ms

C++は、速いね！

(こんなことしなくても$\Theta(N^2), \Theta(N \log N), \Theta(N)$でできます)

おさらい

Sparse Table とは

結合則と冪等性が成り立つ操作を 前計算 $\Theta(n \log n)$, クエリ $\Theta(1)$

でしることができるデータ構造

でした

冪等性、いる？

冪等性が必要だったのは、クエリを分割した時に同じ区間を覆う部分が出てくるからでした

がんばれば、冪等性が必要ない同等の性能のものが作れるんじゃないか？

幂等性が必要だったのは、クエリを分割した時に同じ区間を覆う部分が出てくるからでした

がんばれば、幂等性が必要ない同等の性能のものが作れるんじゃないか？

Disjoint Sparse Table < 呼んだ？

Disjoint Sparse Table

やりたいこと

結合則~~と冪等性~~が成り立つ操作を 前計算 $\Theta(n \log n)$, クエリ $\Theta(1)$

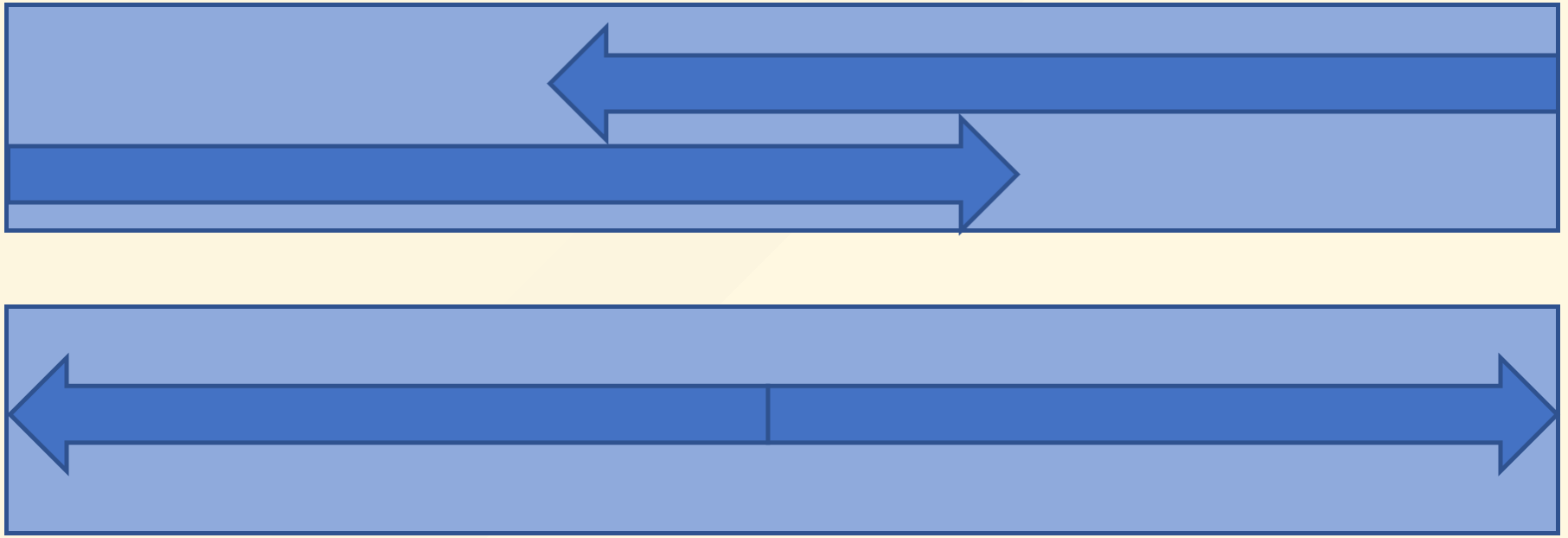
で求めたい

これができれば、min や gcd だけでなく、和や積、xorなども扱える

お気持ち

さっきは 端から真ん中に向けて区間を分けて累積和を持ってきた

じゃあ、**真ん中から端に向けて**区間を分けてみたらどうなるだろう



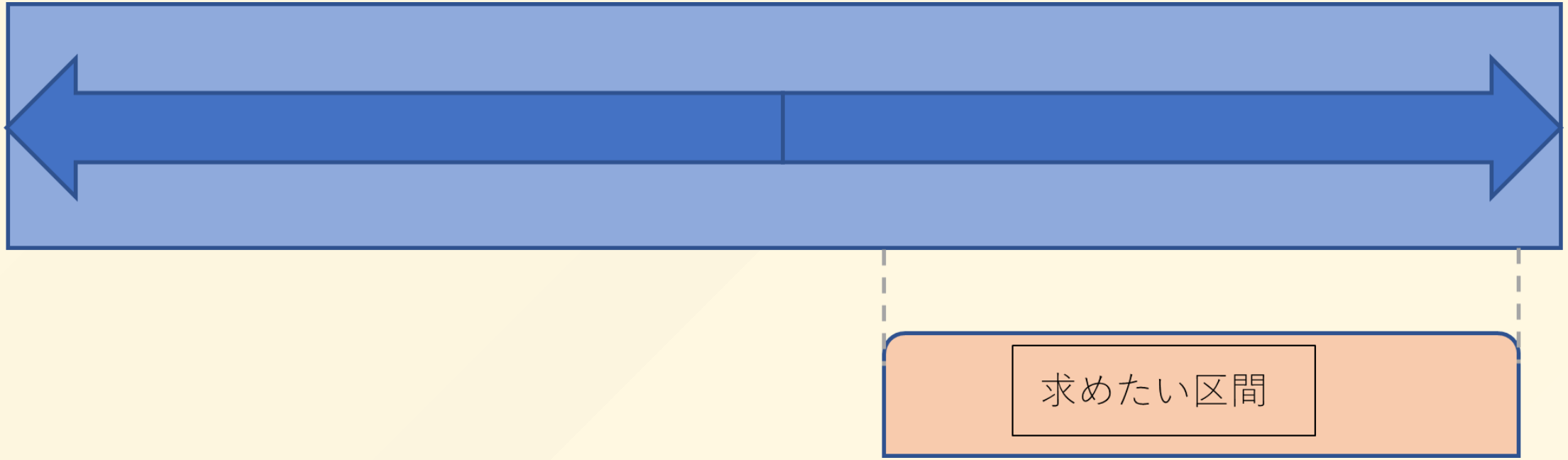
クエリ処理

区間 $[l, r)$ に対して、 真ん中で分けて左右の区間を持ってきて計算する

こうすれば、さっきみたいな感じで $\Theta(1)$ で計算できそう！

ちょっとまて

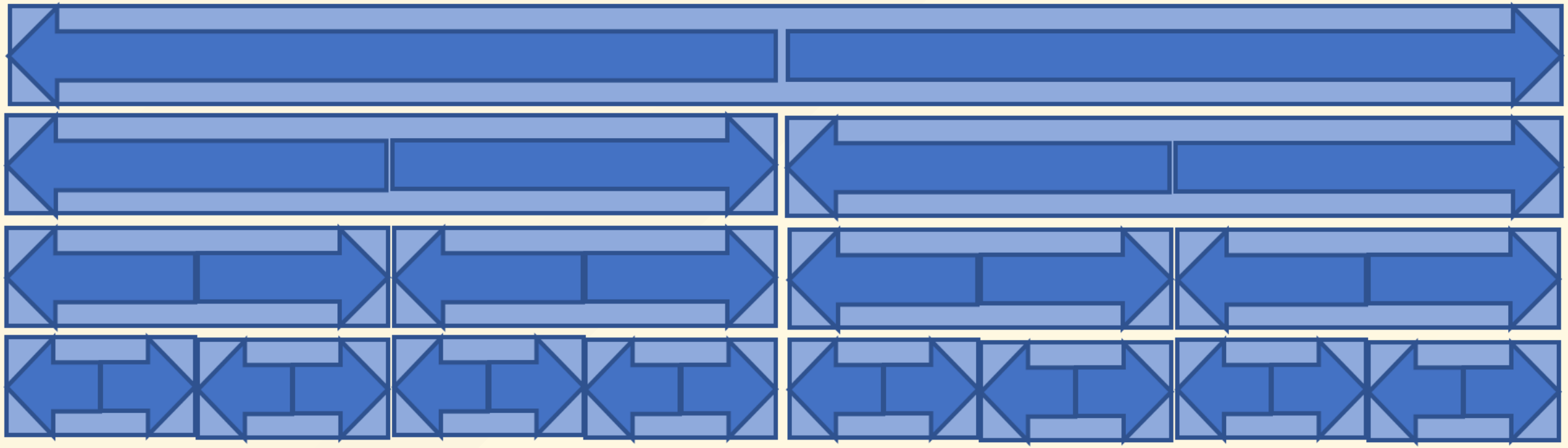
その **真ん中** が区間に入ってなかったらどうするんだ



セグ木を参考にしてみよう

セグ木を参考にしてみよう

セグ木の真似をして、二冪の倍数の位置に真ん中を作ってみよう



よく見ると、どんな区間もどれかでまたがっています

ただし、長さ 1 の区間だけはできないため、別処理をしないと駄目です

この構造は素直に累積和を計算していけば $\Theta(n \log n)$ で作ることができます

クエリ処理

$\Theta(\log n)$ 行のテーブルを作りましたが、結局クエリの処理ではどれを使えばいいのでしょうか

普通に選ぼうとするとまたここで $\Theta(\log n)$ かかってしまいそうです

お気持ち

セグ木の真似をして作ってみたので、bitに注目してみるといい性質があるかもしれない…？

区間がまたがる をbit的に考えるとどうなるか考えてみましょう

二冪の倍数の位置に真ん中を作ったので、それをまたがるということは $t \cdot 2^k - 1$ と $t \cdot 2^k$ をまたがるということです

つまり、そこを通るとその境界に対応するbitがインクリメントされるわけです

ということでbit演算を使います

閉区間 $[l, r]$ に対し、$l \otimes r$ を計算します（ただし $\otimes$ は bitごとの排他的論理和）

その値の、「立っているbitのうち一番左側のものが右から何番目なのか」がどの行を使えばいいのかに対応しています

（$l = r$のとき $l \otimes r = 0$ になってしまい困りますが、長さ 1 なので例外処理をします）

ん？「立っているbitのうち一番左側のものが右から何番目なのか」を見つけるのに結局 $\Theta(\log n)$ かかりませんか？

となりますが、$l \otimes r$ の値も高々 $2n$ 個程度しかありません（繰り上がることがないので）

さっきと同じように、これもうまく前計算してしまいましょう

データの持ち方と実装

半群の要素列 $\{a_n\}$ の区間に対して演算 f を行った結果が欲しいとします

$\mathrm{table_{ij}} = \mathrm{t} \cdot 2^{\mathrm{i}}$ を真ん中にする段において、対応する真ん中から j までの閉区間の累積和

つまり、$[t \cdot 2^i, j]$ または $[j, t \cdot 2^i - 1]$

$\mathrm{lookup_i} = \mathrm{i}$ の最上位bitが右から何番目か とすると、

$$\mathrm{table}_{0,\mathrm{j}} = a_j$$

$$l = t \cdot 2^i, m = j + 2^i, r = m + 2^i \quad (i > 0)$$

$$\mathrm{table}_{\mathrm{i},\mathrm{m}-1} = a_{m-1}$$

$$\mathrm{table}_{\mathrm{i},\mathrm{j}} = f(a[j], \mathrm{table}_{\mathrm{i},\mathrm{j}+1}) \quad (\mathrm{l} \leq \mathrm{j} < \mathrm{m} - 1)$$

$$\mathrm{table}_{\mathrm{i},\mathrm{m}} = a_m$$

$$\mathrm{table}_{\mathrm{i},\mathrm{j}} = f(\mathrm{table}_{\mathrm{i},\mathrm{j}-1}, \mathrm{a}[\mathrm{j}]) \quad (\mathrm{m} + 1 \leq \mathrm{j} < \mathrm{r})$$

$$\mathrm{lookup}_1 = 0$$

$$\mathrm{lookup}_i = \mathrm{loolup}_{\lfloor i/2 \rfloor} + 1 \quad (i > 1)$$

というように計算をすることができます

クエリ $[l, r]$ を求めるときは

もし $l = r$ なら $\mathrm{table}_{0,\mathrm{l}}$ を返すことにして、そうでなければ

$b = \mathrm{lookup}_{\mathrm{l} \otimes \mathrm{r}}$ として

$$\mathrm{res} = \mathrm{f}(\mathrm{table}_{\mathrm{b},\mathrm{l}}, \mathrm{table}_{\mathrm{b},\mathrm{r}})$$

となります

ちなみに、テーブルの長さはセグ木同様二冪にした方が実装は楽ですが、そのままでもそこまで大変にはなりません（端っこまで累積和求めればいい）

ということで

結合則が成り立つ操作を 前計算 $\Theta(n \log N)$, クエリ $\Theta(1)$

で求めることができました！

普通のSparse Tableの完全上位互換ですね

おまけ

その他のSparse Table

その他のSparse Table

- 2D Sparse Table

二次元のSparse Tableもあるらしい

構築 $\Theta(nm \log n \log m)$, クエリ $\Theta(1)$ でできるらしい

お気持ち

table_{ij} の各要素に更にSparse Tableを突っ込むと...？

出典

データの持ち方

$n \times m$ の平面の区域内の長方形領域に対するクエリの処理をしたいとします

まず、$\log n \times n \times \log m \times m$ の四次元配列を用意します

だいたい、各 i, j に対し $\mathrm{table}_{\mathrm{ij}}$ に Sparse Table が入ることになります

正確には、

$\mathrm{table}_{\mathrm{ri},\mathrm{rj},\mathrm{ci},\mathrm{cj}} = [\mathrm{rj}, \mathrm{rj} + 2^{\mathrm{ri}}) \cdot [\mathrm{cj}, \mathrm{cj} + 2^{\mathrm{ci}})$ の演算結果

となります

ここで、

$\mathrm{table}_{0,\mathrm{rj},\mathrm{ci},\mathrm{cj}} = [\mathrm{rj}, \mathrm{rj} + 1) \cdot [\mathrm{cj}, \mathrm{cj} + 2^{\mathrm{ci}})$

となり、これは一次元のSparse Tableと同じになります

実装の流れ

まず、$ri = 0$ として各 rj に対し、（各行の）一次元のSparse Tableを求めます

ここまでの計算は $\Theta(nm \log m)$ となります

そして、ここをベースに全体を計算していきます

一次元のときと同じように

$$\mathrm{table}_{\mathrm{ri},\mathrm{rj},\mathrm{ci},\mathrm{cj}} = \mathrm{f}(\mathrm{table}_{\mathrm{ri}-1,\mathrm{rj},\mathrm{ci},\mathrm{cj}}, \mathrm{table}_{\mathrm{ri}-1,\mathrm{rj}+2^{\mathrm{ri}-1},\mathrm{ci},\mathrm{cj}})$$

というように区間をマージして計算していきます

ここに $\Theta(nm \log n \log m)$ かかります

クエリ処理

クエリ : $[x1, x2)[y1, y2)$ の演算結果

まず、x の方を考えます

一次元のときと同じように二つの二冪に分け、

$[x1, x1 + 2^{xk}), [x2 - 2^{xk}, x2)$ とします

y の方も同じです

一次元のときと同じように二つの二冪に分け、

$[y1, y1 + 2^{yk}), [y2 - 2^{yk}, y2)$ とします

それぞれに対応したtableを選びます

$$R1 = f(\mathrm{table}_{\mathrm{kx,x1,ky,y1}}, \mathrm{table}_{\mathrm{kx,x1,ky,y2}-2^{\mathrm{ky}}})$$
$$R2 = f(\mathrm{table}_{\mathrm{kx,x2}-2^{\mathrm{kx}},\mathrm{ky,y1}}, \mathrm{table}_{\mathrm{kx,x2}-2^{\mathrm{kx}},\mathrm{ky,y2}-2^{\mathrm{ky}}})$$

として

$res = f(R1, R2)$

としてクエリ処理ができました

計算量は変わらず $\Theta(1)$ です

??? 「2D Sparse Tableができるなら 2D Disjoint Sparse Tableもできませんか？」

僕 「うーーーーん...」

できました

考え方は2D Sparse Tableとだいたい同じで、いい感じに持っていい感じにマージしました

これ以上長くなるとあれなので記事にしました

その他のSparse Table

- ±1-RmQ

隣り合う要素の差が必ず ± 1 であるような数列のRmQを、構築 $\Theta(n)$ クエリ $\Theta(1)$ で行うことができるらしい

主にLCAで使うらしい（それ以外を知らない）

Sparse Tableとバケット法の合わせ技

- いい感じに分割してその範囲の最小値を求めておく
- そのバケットを一単位としてSparse Tableを生やす
- 完全に覆われる部分はSparse Tableで、はみ出た部分は別で計算する
- はみ出る部分の変化パターンが少ないので、事前計算して持っておける

変化パターン: 変化ごとに、$+$ か $-$ を選ぶ

$\rightarrow$ バケットの長さを m として 2^{m-1} 通り

先頭の数字を 0 として前計算すると $\Theta\left(m^2 2^m\right)$ かかる

バケットの長さが m なのでバケットの数は $\dfrac{n}{m}$

よってSparse Tableの構築は $\dfrac{n}{m}\log\dfrac{n}{m}$ となる

ここで、$m = \dfrac{\log n}{2}$とする（！？）

$$\begin{aligned}
\frac{n}{m}\log\frac{n}{m} &= \frac{2n}{\log n}\log\frac{2n}{\log n} \\
&= \frac{2n}{\log n}(\log n + \log\frac{2}{\log n}) \\
&= 2n(1 + \frac{\log\dfrac{2}{\log n}}{\log n}) \\
&= \Theta(n)
\end{aligned}$$

はみ出る部分は

$$
\begin{aligned}
& m^2 2^m \\
=& \left(\frac{\log n}{2}\right)^2 \cdot 2^{\log n/2} \\
=& \left(\frac{\log n}{2}\right)^2 \cdot n^{1/2} \\
=& \Theta\left(\log^2 n \cdot \sqrt{n}\right) \\
=& O(n)
\end{aligned}
$$

ということで全体 $\Theta(n)$ で前計算が終わります

クエリは前計算した表を見れば $\Theta(1)$ で求めることができます

やばげな噂

一般の整数列のRmQを、構築 $\Theta(n)$ クエリ $\Theta(1)$ で行うことができるらしい...

出典

ごめんなさいまだ理解してません...

±1-RmQと同じで、$\frac{\log n}{2}$ごとのブロックに分けてSparse Tableで管理するようです

参考文献

https://ei1333.github.io/library/structure/others/sparse-table.cpp | Sparse-Table(スパーステーブル) | Luzhiled's Library
https://ikatakos.com/pot/programming_algorithm/data_structure/sparse_table | Sparse Table [いかたこのたこつぼ]
https://noshi91.hatenablog.com/entry/2018/05/08/183946 | Disjoint Sparse Table と セグ木に関するポエム - noshi91のメモ
https://blog.hamayanhamayan.com/entry/2018/01/03/035508 | 競技プログラミングにおけるSparse Table問題まとめ - はまやんはまやん はまやん

おわり！