Habatu 解説

原案 / Writer / 解説 : Cyanmond
Tester : shiomusubi496

注意？

前半は現地で説明することを想定されているのに対して、後半は解説イベント後に付け足した部分が多いです。

前半は再掲などが割と多いですが、後半は読み返しながら進めることを前提としています。

問題概要

- パ研部員が一列に並んでいて、それぞれにパソコン力 A_i が定められている。
- 隣り合う部員には仲良し度 B_i が定められている。
- 派閥争いが始まると、最も仲良し度の高い隣り合う 2 派閥において「小さいほうが大きい方に吸収される」イベントが派閥が 1 つになるまで続く。
- パソコン力 / 仲良し度の変更クエリ、派閥争いで最後に残る派閥を求める区間クエリを捌け。

問題概要

- $N, Q \leq 100000$ (!?!?!?)
- 小課題 1 : $Q = 1$
- 小課題 2 : 変更クエリは A (パソコン力) だけ、区間クエリは全体のみ
- 小課題 3 : 変更クエリは B (仲良し度) だけ、区間クエリは全体のみ
- 小課題 4 : 変更クエリは事実上飛んでこない、区間クエリを捌け
- 小課題 5 : 全部載せ

以下、 0-indexed で解説を進めます。ただ、図には 1-indexed のものがあるので適宜読み替えてください。

小課題 1

- $Q = 1$

1 クエリだけなので、例えば $O(N \log N)$ とかで解けそうという想像ができる。

小課題 1

派閥争いの性質を考えてみる。

非常に簡単な性質として、各派閥はどの時点においても人の並びのうちのある区間をなすことがわかる。

区間を `std::set<std::pair<int, int>>` などで管理して、実際にシミュレーションを行うことで解ける。計算量は $O(N \log N)$ となる。

$< O(N), O(1) >$ rmq を用いて操作を逆から見ると $O(N)$ で解ける。

小課題 2

- 変更クエリは A だけ。

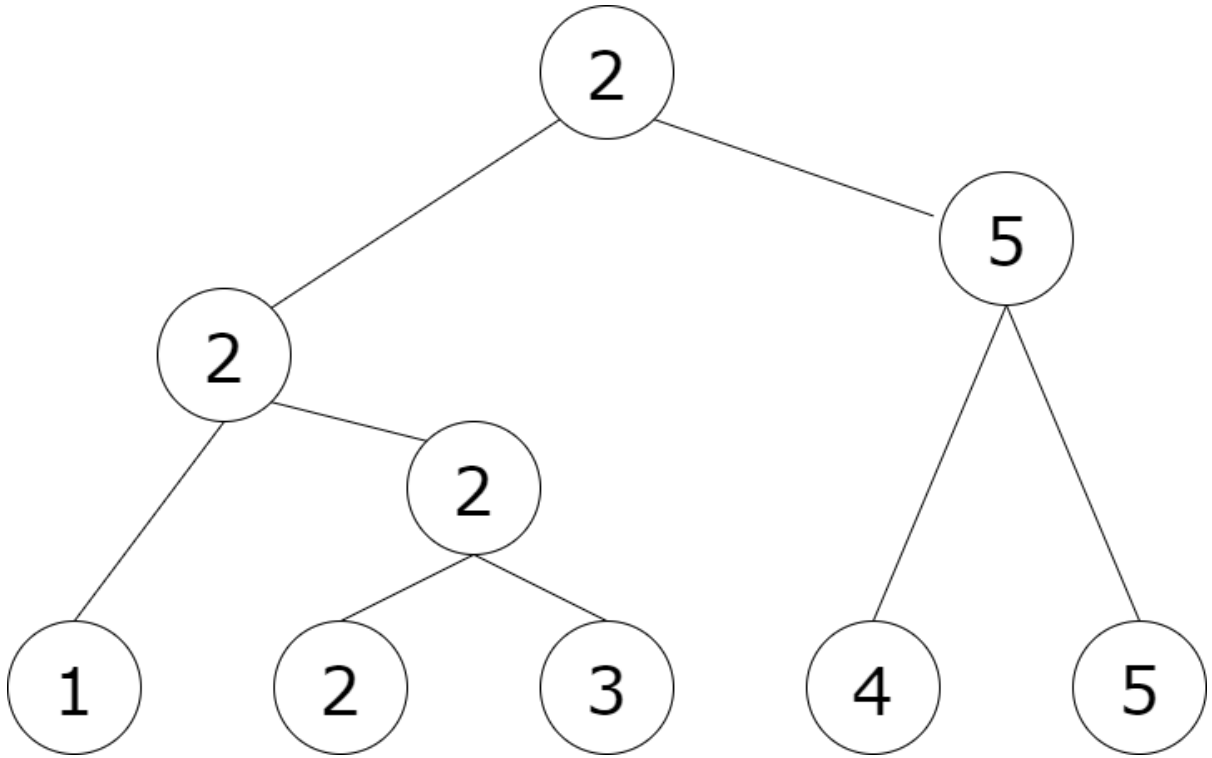
つまり、派閥争いの過程の木は変わらない (後述)

小課題 2

派閥争いの過程は、木構造と捉えることができる。A の要素を $\inf$ として A, B をマージしたときの Cartesian Tree と言うのが適切か。

右図は Sample 1

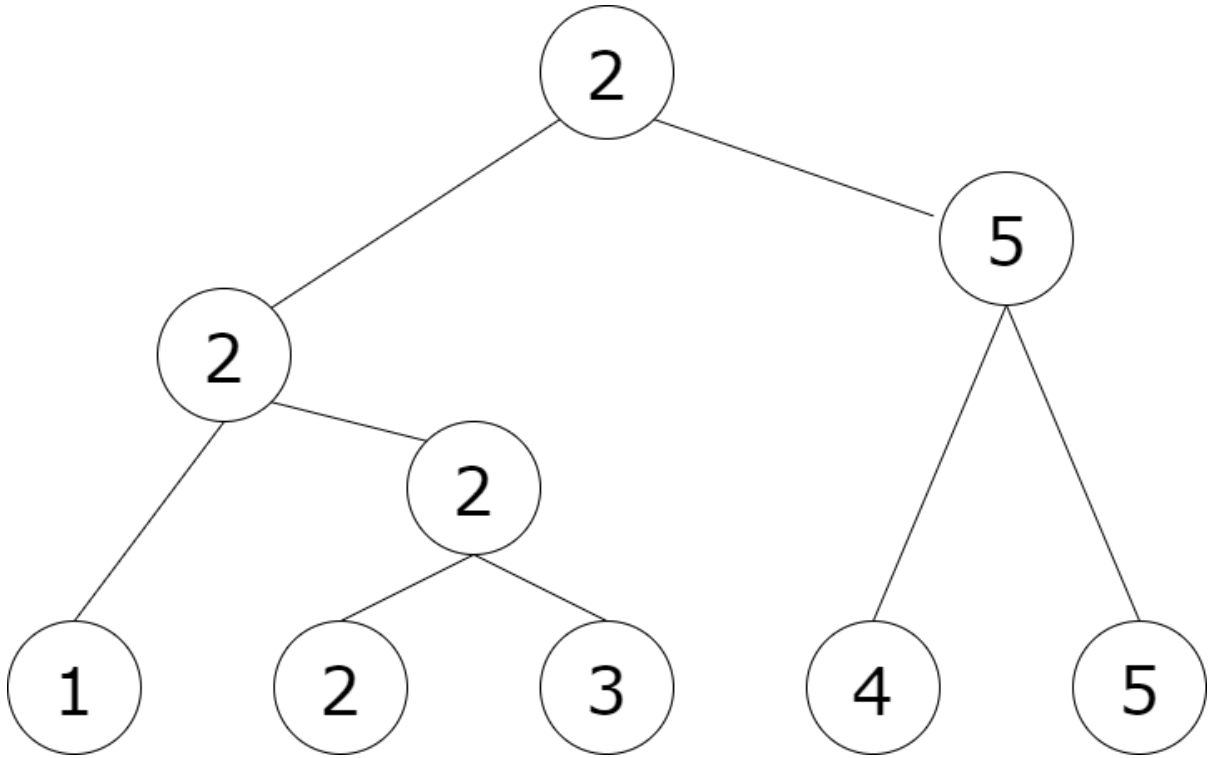
$A = \{5, 9, 9, 7, 9\}$
$B = \{6, 7, 3, 4\}$



小課題 2

$A = \{5, 9, 9, 7, 9\}$
$B = \{6, 7, 3, 4\}$

ここで頂点に書かれている数字は
対応する派閥を表す。

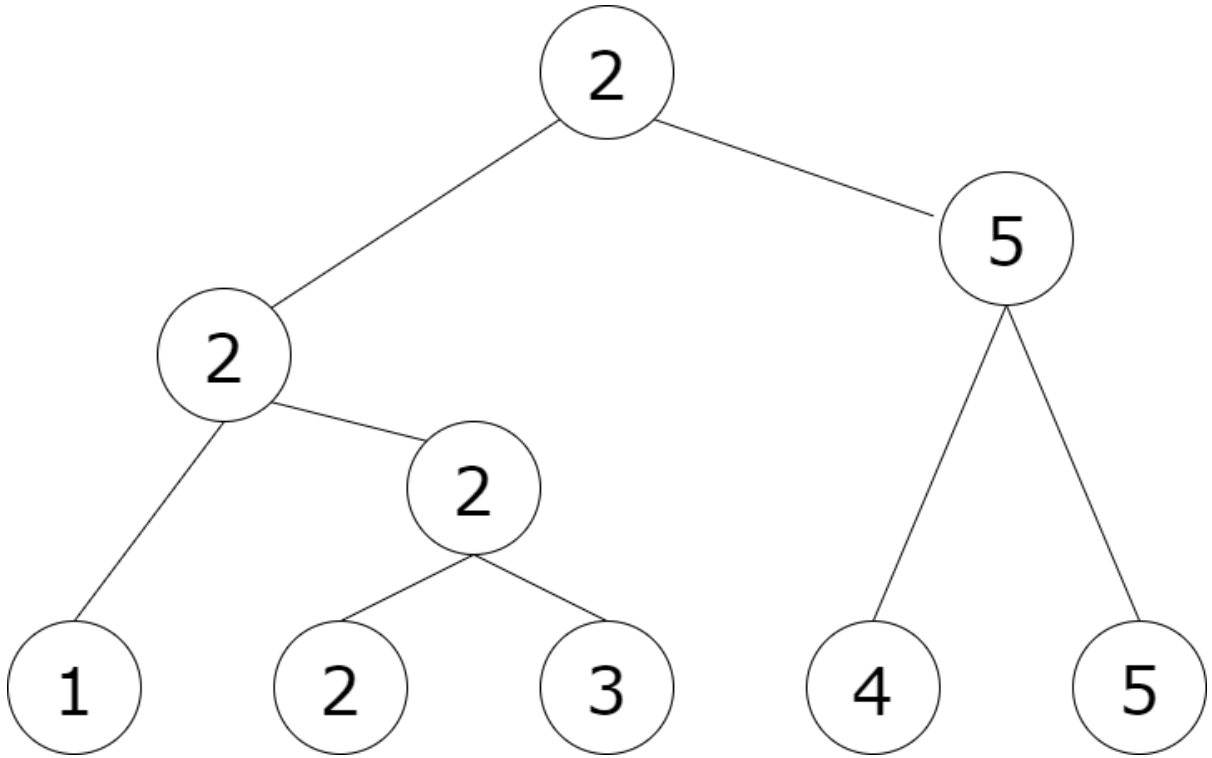


小課題 2

$A = \{5, 9, 9, 7, 9\}$
$B = \{6, 7, 3, 4\}$

ここで頂点に書かれている数字は対応する派閥を表す。

B が変わらないということは、この木の形は変わらない。



小課題 2

計算量を改善するために、もう 1 つ重要な考察を用いる。

- **ある人が所属する派閥は高々 $O(\log N + \log \max A)$ 回しか変わらない。**

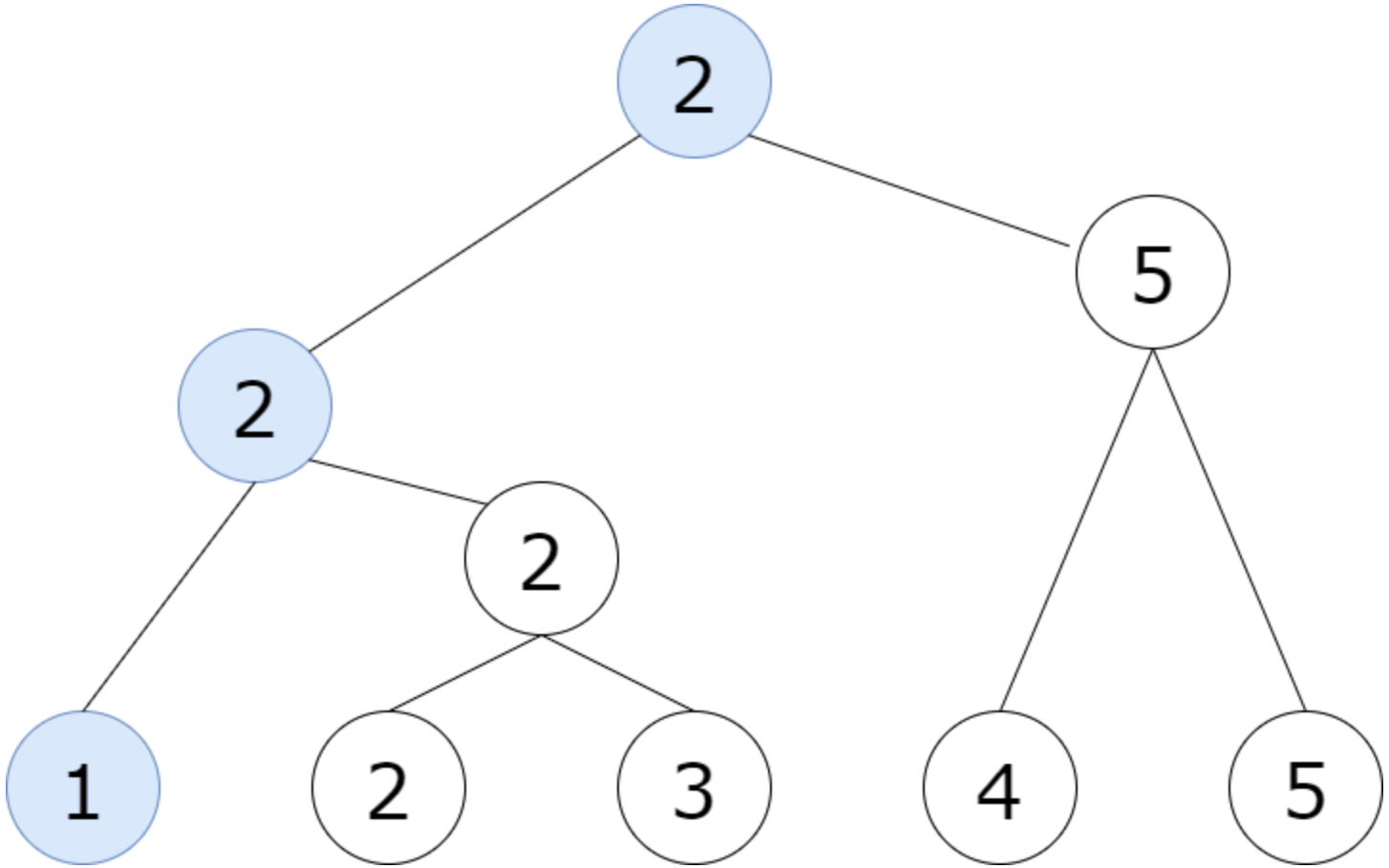
証明の気持ち

- ある人の所属する派閥が変わるとき、新しい派閥のパソコン力の合計は古い派閥のそれの 2 倍以上である。ところで全体のパソコン力の合計は $O(N \max A)$ なので、派閥が $\log_2(N \max A)$ 回以上変わることはありえない。

この考察は典型

小課題 2

ある人が所属する派閥というのは、先の木上で葉から根のパスの形で表される。



小課題 2

- ある人が所属する派閥は高々 $O(\log N \max A)$ 回しか変わらない。ある時点で所属する派閥のパソコン力の合計が x である人の所属する派閥が変わるためにはパソコン力の合計が x 以上の派閥と合体するときのみ。

この考察と合わせると、 HLD と遅延セグメント木等を用いて各クエリ $O(\log N(\log N \max A))$ で解くことができる。

ここまで 250 点。少し渋め。

小課題 3

- 小課題 3 : 変更クエリは B (仲良し度) のみ、クエリは区間全体

先の木を Link-Cut Tree で頑張って管理しようとする方針がある。

- ちょっと TL も厳しそう。区間クエリが辛そう。

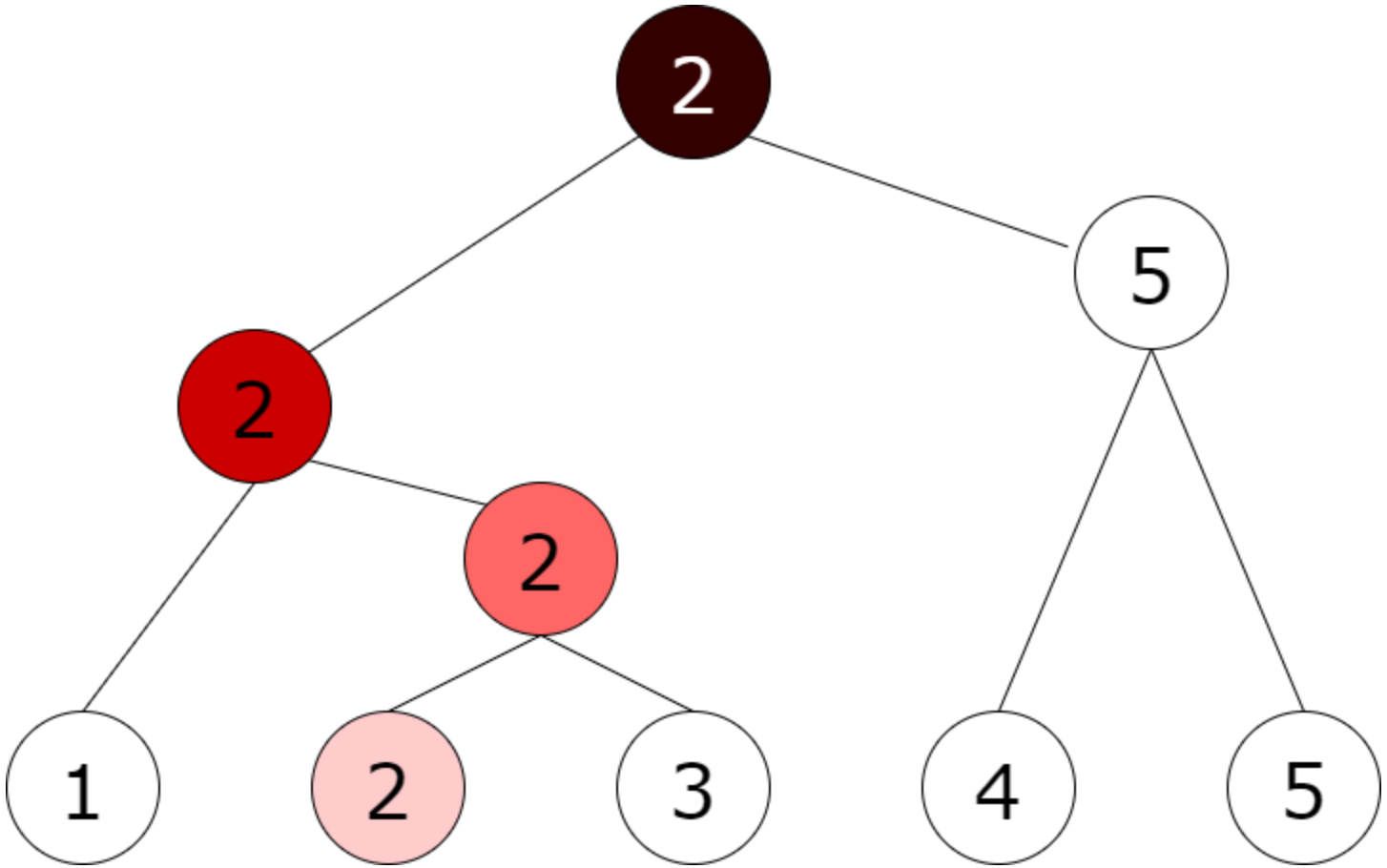
小課題 3 以降では操作を逆から見ることで解く。(小課題 2 の配点が渋いのはそういうことです)

小課題 3

操作を逆から見るとは？

小課題 3

操作を逆から見るとは？



小課題 3

ただ、木を陽に持つべきではない。(木はコロコロ変わるので)

Link Cut 木... とかも、いい感じに計算量が落ちませんでした。log 3 つならできるかも。

木を陽に持つのではなく、区間を縮めていくイメージでやる。

小課題 3

例えば、以下のような解法が考えられる。

- 初め区間 $[l, r) = [0, N)$ とする。以下を $r - l = 1$ になるまで繰り返す。
  - $m = \underset{k=l}{\overset{r-1}{\arg\min}}\, B_k$ とする。$\sum_{k=l}^{m-1} A_k \geq \sum_{k=m}^{r-1} A_k$ ならば r を m に置き換え、そうでない場合 l を m に置き換える。
- $r - l = 1$ になったとき、このクエリの答えは派閥 l である。

小課題 3

例えば、B が Random であれば期待計算量が $O(N \log^2 N)$ となる。

- もちろんそんなことはない。ちゃんと sorted だったり almost sorted だったりするテストケースが入っている。この場合最悪 $O(N^2 \log N)$ となる。

そこで、先ほどの考察を再度用いる。

- ある人が所属する派閥は高々 $O(\log N + \log \max A)$ 回しか変わらない。

小課題 3

- ある人が所属する派閥は高々 $O(\log N + \log \max A)$ 回しか変わらない。

これは、操作を逆から見るとどのようなうれしい性質があるかというと...

- ある区間 $[l, r)$ を今見ているとする。$x = \sum_{k=l}^{r-1} A_k$ として、パソコン力が $\frac{x}{2}$ になるまでは適当に進んでよい。

小課題 3

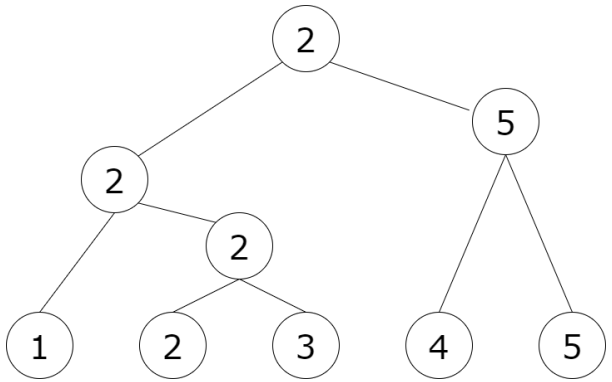
- ある区間 $[l, r)$ を今見ているとする。$x = \sum_{k=l}^{r-1} A_k$ として、パソコン力が $\frac{x}{2}$ になるまでは適当に進んでよい。

もう少し詳しく説明する。

小課題 3

- $\min_{k=l}^{r-2} B_k \leq \max(B_{l-1}, B_{r-1})$ を満たすような (つまり、派閥争いのいずれかの段階でその区間からなる派閥ができるような) 区間のことを、派閥区間と呼ぶことにする。

例えば下の図の場合、派閥区間は
$[0,1), [1,2), [2,3), [3,4), [4,5), [1,3), [3,5), [0,3), [0,5)$ の 9 個。



小課題 3

$[l, r)$ が派閥区間であるとする。$\sum_{k=l}^{r-1} A_k = x$ とする。このとき、

派閥区間 $[a, b)$ $(l \leq a \leq b \leq r)$ であって $\sum_{k=a}^{b-1} > \frac{x}{2}$ である極小なものはただ 1 つ存在する。そして、先の解法において $[l, r)$ から $[a, b)$ まで飛ばしてよい。

小課題 3

つまり、

- 初め区間 $[l, r) = [0, N)$ とする。以下を $r - l = 1$ になるまで繰り返す。
  - $m = \underset{k=l}{\overset{r-1}{\arg\min}} B_k$ とする。$\sum_{k=l}^{m-1} A_k \geq \sum_{k=m}^{r-1} A_k$ ならば r を m に置き換え、そうでない場合 l を m に置き換える。

このステップにおいて、 $[l, r)$ に内包されて、パソコン力の総和が $\dfrac{x}{2}$ より大きいような極小な派閥区間 $[a, b)$ で $[l, r)$ を置き換えるステップを加えても問題ない！

小課題 3

- 初め区間 $[l, r) = [0, N)$ とする。以下を $r - l = 1$ になるまで繰り返す。
  - $[l, r)$ に内包されて、パソコン力の総和が $\dfrac{x}{2}$ より大きいような極小な派閥区間 $[a, b)$ で $[l, r)$ を置き換える。
  - $m = \underset{k=l}{\overset{r-1}{\arg\min}} B_k$ とする。$\displaystyle\sum_{k=l}^{m-1} A_k \geq \sum_{k=m}^{r-1} A_k$ ならば r を m に置き換え、そうでない場合 l を m に置き換える。

このアルゴリズムを考える。各ステップで $[l, r)$ 内のパソコン力の総和が半分以下になっているのでステップ数は $O(\log N + \log \max A)$ 回

小課題 3

ということは、

- $[l, r)$ に内包されて、パソコン力の総和が $\dfrac{x}{2}$ より大きいような極小な派閥区間 $[a, b)$

これを $f(N)$ 時間で求められたとしたら、クエリあたり $O(f(N)(\log N + \log \max A))$ 時間で答えを求められる。

結論から言うと、 A, B の更新があってもこれは $O(\log N)$ で求まる。この高速化の方針が小課題 3, 4, 5 で分かれているので、これを解説する。

小課題 3

- $\sum_{k=l}^{r-1} A_k = x$ とする。(再掲)

すると、以下が成り立つ。

- $l \leq a \leq b \leq r$ かつ $\sum_{k=a}^{b-1} A_k > \frac{x}{2}$ を満たすような全ての整数組 (a, b) について $a \leq m < b$ を満たすような m が存在する。

この m は二分探索で求まる。

小課題 3

よって、この問題は m を含むような派閥区間 $[a, b)$ $(l \leq a \leq m < b \leq r)$ であって $\sum_{k=a}^{r-b} A_k > \frac{x}{2}$ を満たす極小なものを求めることに帰着できた。

小課題 3

- m を含むような派閥区間 $[a, b)$ $(l \leq a \leq m < b \leq r)$ であって $\sum_{k=a}^{b-1} A_k > \frac{x}{2}$ を満たす極小なものを求める

「派閥区間」という条件が難しい。別の形式にならないだろうか？

小課題 3

- m を含むような **派閥区間** $[a, b)$ $(l \leq a \leq m < b \leq r)$ であって $\sum_{k=a}^{b-1} A_k > \frac{x}{2}$ を満たす極小なもの

これの代わりに以下を求めよう。

- m を含む、もしくは m が境界であるような **区間** $[a, b)$ $(l \leq (a - 1) \leq m \leq b \leq r)$ であって $\sum_{k=a}^{b-1} A_k \leq \frac{x}{2}$ を満たす極大なもののうち、$\min_{k=a}^{b-2} B_k$ を最大化するもの

小課題 3

これが求まったとする。このとき、$[a, b)$ を含むような極小な **派閥区間** は一意に定まり (セグ木二分探索などでできる) これが求めたいもの (以下に再掲) である。

- m を含むような **派閥区間** $[a, b)$ $(l \leq a \leq m < b \leq r)$ であって $\displaystyle\sum_{k=a}^{b-1} A_k > \frac{x}{2}$ を満たす極小なもの

小課題 3

結局これを高速に求めればよくなった。

- m を含む、もしくは m が境界であるような **区間** $[a, b)$ $(l \leq (a - 1) \leq m \leq b \leq r)$ であって $\sum_{k=a}^{b-1} A_k \leq \frac{x}{2}$ を満たす極小なもののうち、 $\min_{k=a}^{b-2} B_k$ を最大化するもの

小課題 3

例えば、これは簡単な二分探索で (判定の各ステップでセグ木二分探索をして) $O(\log^2 N)$ となる。

- これ BIT で高速化を頑張ると通るらしい？(tute7627 さんの提出)

高速化する。

小課題 3 の性質を考える。

- 変更クエリは B (仲良し度) だけ

小課題 3

変更クエリが B だけであることを用いてこれを用いて $O(\log N)$ にする。

求めるものを修正して、 x ではなく x 以上の最大の 2 冪を用いることにする。

A が更新されないので、各 2 冪の数 2^d $(0 \leq d \leq \log_2(N \max A))$ について以下を前計算できる。

- 各 l $(0 \leq l < N)$ について、 $\sum_{k=l}^{r-1} A_k \geq 2^d$ となる最小の r

小課題 3

これが前計算されていると、クエリの区間が数列全体であることと合わせて $O(\log N)$ で先の値を求められる。

これは十分高速。計算量は前計算 $O(N(\log N + \log \max A))$ で、クエリ毎 $O(\log N(\log N + \log \max A))$ となる。

600 点。

小課題 4

- 変更クエリが (事実上) なし、質問の区間が任意区間になる

質問のクエリが固定ではないと先の高速化はおそらく効かない。(もしかしたら効くかも)

実は二分探索をもっと賢くできる。

小課題 4

左右並列に二分探索を進めていく。

この二分探索は以下のようなステップからなる。

- $l_1 = l - 1, r_1 = m, l_2 = m, r_2 = r$ と初期化する。
- $r_1 - l_1 = 1$ かつ $r_2 - l_2 = 1$ が満たされるまで以下のステップを繰り返す。
  - $m_1 = \dfrac{l_1 + r_1}{2}, m_2 = \dfrac{l_2 + r_2}{2}$ とする。
  - $\displaystyle\sum_{k=m_1}^{m_2-1} A_k \leq \frac{x}{2}$ であれば後述の操作 1, そうでないなら操作 2 を行う。

小課題 4

操作 1 : $\min_{k=m_1}^{m-1} B_k > \min_{k=m}^{m_2-1} B_k$ である場合、r_1 を m_1 で置き換える。そうでない場合、l_2 を m_2 で置き換える。

操作 2 : $\min_{k=m_1}^{m-1} B_l < \min_{k=m}^{m_2-1} B_k$ である場合、l_1 を m_1 で置き換える。そうでない場合、r_2 を m_2 で置き換える。

以上は $O(\log(r-l))$ ステップで $r_1 - l_1 = 1, r_2 - l_2 = 1$ となる。このとき、$[r_1, l_2)$ が求めたいものである。

小課題 4 - 余談

この解法を一般化して、例えば以下の問題を解くことができる。

長さ N の単調非減少な正整数列 M 個 ($A_1, A_2, \cdots, A_M$ とする) と単調非増加な正整数列 M 個 ($B_1, B_2, \cdots, B_M$ とする) が隠されている。

整数 X が与えられるので、 index の列 $i_1, i_2, \cdots, i_M$ であって $A_1[i_1] + A_2[i_2] + \cdots + A_M[i_M] \geq X$ を満たすようなもののうち $\min(B_1[i_1], B_2[i_2], \cdots, B_M[i_M])$ を最大化するものを高々 $O(M \log N)$ 要素へのアクセスで 1 つ求めよ。

(余談終わり)

小課題 4

A, B に更新が来ないことから、累積和と Sparse Table を用いることで以上を $O(\log(r-l))$ 時間で実行できる。これは十分高速。

計算量は前計算 $O(N)$ のクエリ毎 $O(\log N(\log N + \log \max A))$ となる。

小課題 5

Segment Tree 上にこれを拡張できる。右端ノードと左端ノードをそれぞれ持っておき、適切に管理しながら両端を広げていく。

あるいは、バケット法でクエリごとにたくさん計算をしておくことで区間取得 $O(1)$ 時間を保つ。この場合、例えば Sparse Table を用いるとクエリ毎に $O(\sqrt{N}\log N)$ かかる。

前者の方法では、計算量は $O(N + Q(\log N(\log N + \log \max A)))$ となる。かなり定数倍に悪い実装でも 1400 ms だったので、十分に高速。

オンサイト

100 点 6 人

全体

900 点 1 人 (tute7627 さん)
100 点 16 人