5 衝突 (Collision)

Author : 蜂矢 倫久 (Mitsubachi)

小課題 1

この小課題においては参加者の 1 ビョウあたりの速度は 1 か 2 であり，マラソン大会の時間は 1 ビョウである．したがって，重なりはゼッケン x を着用した 1 ビョウあたりの速度が 2 の人と，ゼッケン $x+1$ を着用した 1 ビョウあたりの速度が 1 の人が時刻 1 で同じ地点 $x+2$ にいる場合に限られる．

よって，人を追加したり削除する際の衝突の回数の変化は，その人がゼッケン A を着用しているとして，ゼッケン $A-1$ とゼッケン $A+1$ を着用している人がいるか，いる場合はその人の速度はいくつかが分かれば求められる．$A=0$ の場合はゼッケン $L-1$ に関する情報，$A=L-1$ の場合はゼッケン 0 に関する情報を参照するべきであることに注意せよ．

以上の操作は map などで可能である．$O(N)$ や $O(N \log N)$ などで答えが求まる．

小課題 2

ゼッケン A を着用した 1 ビョウあたりの速度が S の人 P と，ゼッケン A' を着用した 1 ビョウあたりの速度が S' の人 P' との間で起こる衝突の回数を考える．

まず，$S \leqq S'$ として一般性を失わないので，$S \leqq S'$ が成り立っていると仮定する．このとき，P' が P を 1 ビョウあたり $S'-S$ の速度で追いかけていると考えられる．すなわち，$A'-A+t(S'-S)$ が L の整数倍となるような時刻 t に 2 人は衝突する．

このような t の個数は $\left\lfloor \dfrac{A'-A+T(S'-S)}{L} \right\rfloor - \left\lfloor \dfrac{A'-A}{L} \right\rfloor$ となる．これは $O(1)$ で計算できる．

したがって，これを全ての 2 人組について計算することで，全体で $O(N^2)$ で答えが求まる．

小課題 3

小課題 2 の解法を変更のたびに行うと $O((N+Q)^2 Q)$ となる．これを高速化することを考える．差分更新という解法を用いると高速化することが可能である．

人を追加するとき，2 人組の候補として増えるのはその人を含む $O(N+Q)$ 通りである．また，削除するときも 2 人組の候補として減るのはその人を含む $O(N+Q)$ 通りである．

したがって，人の追加により増える，または人の削除により減る衝突の回数のみを求めれば，全体で $O((N+Q)^2)$ で答えが求まる．

小課題 4

小課題 2 の解法を高速化することを考える．参加者が 1 から n までの番号が付けられた n 人として，参加者 i がゼッケン A_i を着用し，1 ビョウあたりの速度は S_i とする．ここで $S_1 \leqq S_2 \leqq \cdots \leqq S_n$ が成立しているとして一般性を失わない．

このとき，人 i と人 j $(1 \leqq i < j \leqq n)$ の間で起こる衝突の回数は $\left\lfloor \frac{A_j - A_i + T(S_j - S_i)}{L} \right\rfloor - \left\lfloor \frac{A_j - A_i}{L} \right\rfloor$ と計算できる．これの総和を高速に計算したい．便宜的に衝突の回数を表す式を $\left\lfloor \frac{L + A_j - A_i + T(S_j - S_i)}{L} \right\rfloor - \left\lfloor \frac{L + A_j - A_i}{L} \right\rfloor$ とする．

まず，$\left\lfloor \frac{L + A_j - A_i}{L} \right\rfloor$ の総和を求めることを考える．$0 \leqq A_i, A_j \leqq L-1$ より $\left\lfloor \frac{L + A_j - A_i}{L} \right\rfloor$ は 0 か 1 である．また，1 となる条件は $A_i \leqq A_j$ であることである．したがって，転倒数を求めるのと同様のアルゴリズムで $O(N \log N)$ で計算できる．

次に，$\left\lfloor \frac{L + A_j - A_i + T(S_j - S_i)}{L} \right\rfloor$ の総和を求めることを考える．$A_i + S_i T = D_i L + E_i, 0 \leqq E_i \leqq L-1$ となるように D_i, E_i を定める．これは $O(1)$ で計算できる．このとき，$L + A_j - A_i + T(S_j - S_i) = L + D_j L + E_j - D_i L - E_i = L(D_j - D_i) + L + E_j - E_i$ となる．

したがって，$\left\lfloor \frac{L + A_j - A_i + T(S_j - S_i)}{L} \right\rfloor = D_j - D_i + \left\lfloor \frac{L + E_j - E_i}{L} \right\rfloor$ となる．$D_j - D_i$ の総和について，これは $D_i(2i - n - 1)$ を i について足し合わせたものである．

また，$\left\lfloor \frac{L + E_j - E_i}{L} \right\rfloor$ の総和について，$0 \leqq E_i, E_j \leqq L-1$ より $\left\lfloor \frac{L + A_j - A_i}{L} \right\rfloor$ の場合と同様にして求められる．

以上より，全体で $O(N \log N)$ で答えが求まる．

小課題 5，小課題 6

満点となるアルゴリズムをこれから説明するが，定数倍が悪い場合もしくは実行時間の遅い言語では小課題 5 までしか実行時間に間に合わない可能性がある．

小課題 4 の解法を人の変更に対応させることを考える．まず，X や Y を先読みすることで参加者の速度としてありうる値の一覧を作成しておく．

小課題 4 における $D_j - D_i$ の総和について考える．速度が s の人を追加するとき，それに伴う変化を求めることを考える．この人における D を d とする．また，速度が s 以下の人の人数とその人における D の総和をそれぞれ g, h とする．また，速度が $s+1$ 以上の人の人数とその人における D の総和をそれぞれ g', h'

とする．

このとき，$D_j - D_i$ の総和の変化は $h' - g'd + gd - h$ となる．ここまでの議論で用いた g, h の値を計算するまたはクエリの変更に対応できるようにするには，各速度ごとに人数と D の総和を管理する Segment Tree や Binary Indexed Tree を作成すればよい．参加者の速度としてありうる値の一覧を用いて座標圧縮しておくと便利である．

人を削除するときも同様に考えることができる．クエリあたり $O(\log(N+Q))$ の計算量で処理できる．

$\left\lfloor \dfrac{L + A_j - A_i}{L} \right\rfloor$ の総和についても同様のことを考える．ゼッケン a を着用し，速度が s の人を追加するとき，それに伴う変化を求めることを考える．

小課題 4 における考察より，速度が s 以下で，$a-1$ 以下のゼッケンを着用する人数や，速度が $s+1$ 以上で，$a+1$ 以上のゼッケンを着用する人数を求めることができれば良い．したがって，以下のような問題に帰着することができる．

2 次元平面上における以下のクエリを q 回処理せよ．それぞれのクエリは以下のパターン 1 またはパターン 2 のいずれかである．また，クエリの情報は前もって与えられるものとする．(このような問題をオフラインクエリと言う．)

- パターン 1: 2 次元平面上の点 (x, y) における重みを z 増加させる．
- パターン 2: 2 次元平面上の $x_l \leq x \leq x_r, y_l \leq y \leq y_r$ を満たす点 (x, y) における重みを足し合わせた値を求める．

この問題には 2 種類の解法が考えられる．事前にパターン 1 で登場する x の集合と y の集合を求め，それに伴い座標圧縮を行うものとする．また，i 番目のクエリをクエリ i と呼ぶ．

解法 1

クエリ分割統治と呼ばれるテクニックである．

クエリ 1 からクエリ $\left\lfloor \frac{q}{2} \right\rfloor$ におけるパターン 1 のクエリ全体がクエリ $\left\lfloor \frac{q}{2} \right\rfloor + 1$ からクエリ q におけるパターン 2 のクエリ全体に与える寄与を求めることを考える．

これは平面走査により $O(q \log q)$ で処理することができる．後は，問題のクエリを前半に限定したものと後半に限定したものを解けば元の問題を解くことができる．

したがって，q 個のクエリを処理するのに必要な計算量を $T(q)$ とすると，$T(q) = 2T\left(\frac{q}{2}\right) + O(q \log q)$ となる．したがって，$T(q) = O(q(\log q)^2)$ となるので，$O(q(\log q)^2)$ で解くことができる．

解法 2

平方分割と呼ばれるテクニックである．

座標圧縮に伴い，2 次元平面の x 座標と y 座標をそれぞれ $O(\sqrt{q})$ 個の区間に分割する．これにより，2 次元平面は $O(q)$ 個の部分長方形に分けることができる．また，クエリをクエリ 1 からクエリ B まで，クエリ $B+1$ からクエリ $2B$ まで，... と $O\left(\frac{q}{B}\right)$ 個の区間に分ける．各区間のクエリを処理し終わるたびに，部分長方形における重みの総和に関して累積和を取る．

パターン 2 のクエリを処理することを考える．自分より前の区間にあり，かつパターン 2 のクエリにおける長方形に包含されている部分長方形における重みの総和は累積和により求められる．それ以外のパターン 1 のクエリにおける寄与は愚直に計算することができる．

したがって，$O\left(\frac{q^2}{B}+qB\right)$ で解くことができる．$B=\Theta(\sqrt{q})$ とすることで，$O(q\sqrt{q})$ で解くことができる．

したがって，この問題は全体で $O((N+Q)(\log(N+Q))^2)$ または $O((N+Q)\sqrt{N+Q})$ で答えが求まる．実装の細かい部分については，解答例を参照せよ．

シラバス外の内容となるが，必要なところのみメモリを作る 2 次元動的 Segment Tree もしくは Binary Indexed Tree や，Wavelet Matrix などを用いて解く方法も考えられる．