バブルソート 2 (解説)

数列の i 番目の値に (微小量) $\times i$ を足して考えることにすると，数列の値が全て異なるとしてよい．

(自身より前にある自身より高い要素) が 1 つ以上ある要素は，一回の走査で 1 つだけ前に進む．よって，(自身より前にある自身より高い要素) の最大値が，バブルソートによる走査回数となる．ところで，この値が最大値を取りうるのは，数列の右下階段凸法に含まれる頂点である．これらの頂点に対しては，

$$(\text{自身より前にある自身より高い要素の個数}) = i - (A_i\text{より小さい要素の個数})$$

が成り立つ．また，それ以外の頂点に関しては，

$$(\text{自身より前にある自身より高い要素の個数}) > i - (A_i\text{より小さい要素の個数})$$

が成り立つ．よって，数列 A のバブルソートによる走査回数は，

$$\max\left\{ i - (A_i\text{より小さい要素の個数}) \,\middle|\, i = 0, 1, \cdots, N-1 \right\}$$

で求められる．

すべての i に対して

$$i - (A_i\text{より小さい要素の数})$$

を管理し，更新，最大値の取得が高速に行えれば良い．i の項は不変なので，クエリのたびに愚直に変更すれば良い．

$$-(A_i\text{より小さい要素の個数})$$

の項の更新は，クエリのたびに A_i がある範囲に含まれる要素に $+1$ または -1 をすれば良い．

以上の操作は Segment Tree を使えば 1 クエリあたり $O(\log(N+Q))$ で行うことができる．また，数列に登場する値の座標圧縮には，$O((N+Q)\log(N+Q))$ かかる．よって，全体で $O((N+Q)\log(N+Q))$ で，この問題を解くことができる．