第２日　Mountains　解説

解法

入力において，文字 'I' で始まる行（すなわち，設定の変更）の数を I で表すことにする．同様に，乗車の数を Q で表すことにする．

単純な解法

単純ではあるが最適ではない解法は，軌道の状況を次のようにベクトル A で表すことである．

> ベクトル A の i 番目の要素 $A[i]$ は i 番目のレールの終端の高さ
> を表すものとする．

1 回の設定の変更を行う（この操作を挿入と呼ぶ）の時間計算量は $O(n)$ である．乗車に関する 1 回の質問も $O(n)$ 時間かかるので，合計の時間計算量はで $O((I+Q)\cdot n)$ となる．その上，領域計算量も $O(n)$ となり，大きなテストケースにおいてはメモリ制限を超えずに実行するには大きすぎる．

他の単純なアプローチは，軌道の状況を同一区間に含まれるレールの高さの変化は全て同じである互いに素な区間のリストで表すものである．この方法の挿入と質問の時間計算量は $O((I+Q))\cdot I$ である．領域計算量は $O(I)$ である．

モデル解法

このモデル解法で用いるデータ構造は 2 分木である．各ノードはある整数 k と t に対する区間（すなわち，連続したレール）$J=\left[2^k t, 2^k(t+1)\right)$ に関する情報を記述する．各ノードに含まれる情報は

- $S_J=\sum_{j\in J} d_i$
- $H_J=\max\left(\{0\}\cup\left\{\sum_{2^k t\le i\le j} d_i : j\in J\right\}\right)$

である．全ての d_i が等しいならば，そのノードは葉である．この場合は S_j の計算も H_J の計算も簡単である．そうでない場合は，そのノードは 2 つの子を持ち，それぞれの子は，区間 $J_1=\left[2^{k-1}(2t), 2^{k-1}(2t+1)\right)$ と区間 $J_2=\left[2^{k-1}(2t+1), 2^{k-1}(2t+2)\right)$ に対応する．この場合は，S_J は $S_J=S_{J_1}+S_{J_2}$ として，H_J は $H_J=\max\{H_{J_1}, S_{J_1}+H_{J_2}\}$ として計算される．この木の根は区間 $\left[0, 2^{\lceil \log_2 n\rceil}\right)$ を表す．

中心になる操作は軌道の設定変更である．設定変更は高々 $2\lceil \log_2 n \rceil$ 個のノードを挿入あるいは変更で実現できる．よって，この解法の時間計算量は $O((I+Q)\cdot \log n)$ で，領域計算量は $O(I \cdot \log n)$ である．

さらに良い解法

入力を一行ずつ処理することは要求されていない．その代わりに，入力全体を読み込み処理前に軌道のどの部分が設定変更されるかを知ることができる．M を入力の I で始まる行に含まれる区間の端の整列されたベクトルとする．また，M の長さは 2 のべき乗であると仮定する.（もしそうでなければ，そうなるように M に大きな値をいくつか追加すればよい.）

この解法においても，前節で記述した木に似た木を用いる．違いは，各ノードは区間 $I = \left[M[2^k t], M[2^k(t+1)]\right)$ に対応することである．ヒープの標準的な実装のように，このような木を一つのベクトルで蓄えることができる．M のサイズは $O(I)$ なので，合計の時間計算量は $O((I+Q)\cdot \log I)$ で，領域計算量は $O(I+Q)$ に減らすことができる．