4 最悪の記者 5 (Worst Reporter 5)

Author: 平木 康傑 (sheyasutaka)

0 問題概要

$(1, 2, \ldots, N)$ の順列 Q が valid であるとは，ある $(1, \ldots, N)$ の順列 $(p_1, p_2, \ldots, p_N)$ を初期配置として，$i = 1, 2, \ldots, M$ の昇順に以下の操作を失敗なく行うことができ，かつ操作後の順列を Q に一致させられることをいう．

- その時点の順列において A_i と B_i が隣り合っている場合，それらの位置を入れ替える．そうでない場合，操作に失敗する．

valid な順列が存在するか判定し，あれば辞書順最小のものを求めよ．

1 小課題 1 ($N \leqq 8$, $M \leqq 8$)

$N!$ 通りの初期配置を全探索することで valid な順列をすべて求め，そのうち辞書順最小のものを求めればよい．時間計算量は $O(N! \cdot NM)$.

2 小課題 2 ($A_1, A_2, \ldots, A_M, B_1, B_2, \ldots, B_M$ は相異なる)

この制約下において，順列 Q は以下の条件を満たすとき，またそのときに限り valid である．

- Q において A_i, B_i は隣接する位置にある．$(1 \leq i \leq M)$

これは，Q が以下の計 $N - M$ 個の数列をつなげてできることと等価である．

- **2 要素の連続部分列** (M 個): 各 $i = 1, 2, \ldots, M$ について，(A_i, B_i) または (B_i, A_i)
- **1 要素の連続部分列** ($N - 2M$ 個): $v \notin \{A_1, A_2, \ldots, A_M, B_1, B_2, \ldots, B_M\}$ を満たす $1 \leq v \leq N$ について，(v)

このような順列は必ず存在する．

辞書順最小の Q を作るという目標において，以下の 2 つの主張が成り立つ．

1. 2 要素の連続部分列は $(A_i, B_i), (B_i, A_i)$ のうち辞書順で小さいほう (この場合は必ず (A_i, B_i)) を選ぶのが最適である．
2. 上で決定した $N - M$ 個の列をつなげるとき，先頭要素が昇順になるように並べるのが最適である．

したがって，$N - M$ 個の数列を先頭要素の昇順でソートして，順につなげればよい．

3　小課題 3 ($N \leqq 1\,000$, $M \leqq 1\,000$)

valid な順列が存在する場合, 順列 Q が valid である条件は, $1, 2, \ldots, N$ を全体でちょうど 1 つずつ含むような数列の集合 $R = \{R_1, R_2, \ldots, R_k\}$ を用いて以下のように言い換えられる．

- Q は $R'_1, R'_2, \ldots, R'_k$ を任意の並びでつなげて得られる数列である．ここでそれぞれの R'_j は，R_j に等しいか，R_j の要素の順序を反転させたものに等しい．

操作が 0 個である状態では，$R = \{(1), (2), \ldots, (N)\}$ とすれば言い換えが成立する．

ここから操作を 1 つずつ追加していくときの R の変化を管理することを考える．

操作 $(A_i, B_i) = (a, b)$ が追加されたとする．

- **a, b が同じ列 R_u に属している場合**: R_u において a, b が隣接する位置にある場合，単にその位置を入れ替える．そうでない場合，valid な順列は存在しない．
- **a, b が異なる列 R_s, R_t に属している場合**: a が R_s の先頭または末尾にあり，かつ b が R_t の先頭または末尾にある場合，R_s, R_t を反転を許してつないだ新しい列 R_u で置き換えた後，a, b の位置を入れ替える．そうでない場合，valid な順列は存在しない．
  - **新しい列 R_u の構成**: a, b が隣り合うように，R'_s, R'_t を任意の順につなげればよい．たとえば，a が R_s の先頭，b が R_t の先頭だった場合，$R_u := rev(R_t) + R_s$ のように構成できる．

たとえば，$R = \{(1, 2, 3), (4, 5, 6), (7, 8, 9)\}$ の状態から操作 $(3, 6)$ が追加されたとき，新しい R は $R = \{(1, 2, 6, 3, 5, 4), (7, 8, 9)\}$ となる．

最終的な R を得たあとは，以下にしたがって列を並べればよい．

1. 各列の向きは，$R_j, rev(R_j)$ のうち辞書順で小さいほうを選ぶのが最適である．
2. 決定した列をつなげるとき，先頭要素が昇順になるように並べるのが最適である．

これらは愚直に実装しても時間計算量 $O(NM)$ で行える．

4　小課題 4 (i 回目 ($2 \leqq i \leqq M$) の順位変動において，A_i と B_i のうち少なくとも片方は $A_1, A_2, \ldots, A_{i-1}, B_1, B_2, \ldots, B_{i-1}$ の中に現れない値である)

この制約下において，操作で与えられる 2 点のうち少なくとも片方は，必ず長さ 1 の列に属する．

追加する操作 (a, b) において b が長さ 1 の列に属するとする．このとき，a が列の先頭であれば先頭から，末尾であれば末尾から以下の操作を行えばよい．

1. a を削除する．
2. b を追加する．
3. a を追加する．

たとえば，$R_j = (1, 2, 3, 4)$ に対して操作 $(1, 6)$ を追加するとき，新しい R_j は $R_j = (1, 6, 2, 3, 4)$ となる．

これは両端キュー (deque) と呼ばれるデータ構造によって実現できる．1 つの deque は 2 つの queue の組合せで実装できる．

ある値がどの列に属するかを管理することで，時間計算量を $O(N + M)$ にできる．

5　小課題 5 (追加の制約なし)

5.1　解法 1: 重軽マージ

小課題 3 の解法を deque で実装することを考えると，列 R_s, R_t をつなぐ操作は，列 R_s に R_t の各要素を push することで実現できるので，計算量は R_t 側の要素数のみで抑えられる．そこで，常に要素数が少ないほうを push する側 (R_t) におくと，実は全体での push 回数は $O(N \log N)$ のオーダーに改善される．

証明: 任意の要素 x について，x が push されるごとに x の属する列の長さは 2 倍以上になっている．したがって，x が push される回数は $\log_2 N$ 以下である．

ある値がどの列のどの位置に属するかを管理することで，時間計算量を $O(M + N \log N)$ にできる．

5.2　解法 2: 双方向リスト

双方向リストを用いて同様の解法を実装すると，列をつなげる操作が定数時間で実現できる．

2 つの値が同じ列に属するかを判定するために，追加の情報を持つ必要があることに注意が必要である．これは Union-Find 木によって実現できる．

時間計算量は $O(N + M\alpha(N))$ である．ここで，$\alpha(N)$ はアッカーマンの逆関数である．