JAG夏合宿2017

K: Permutation Period

原案: darsein

問題文: tubo28

解答: darsein, tubo28

解説: darsein

問題概要

- 長さ N の恒等順列 p = [1, 2, ..., N] が与えられる
- Q 個のスワップを順に適用していき、各適用後の p の周期を mod $10^9 + 7$ で求めよ
    - 順列の周期: $p_i^k = p_{p_i}^{(k-1)}$ としたとき、$p_i = p_i^{(k+1)}$ $(1 \leq i \leq N)$ となる最小の k (>0)

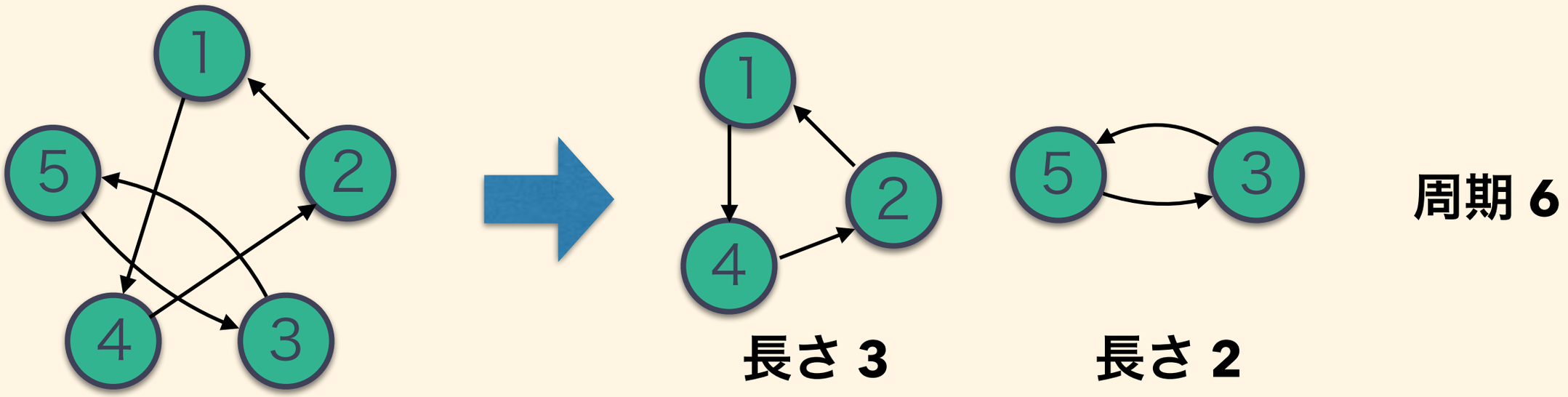
- 制約: $2 \leq N \leq 10^5$, $1 \leq Q \leq 10^5$

順列の周期の計算

- $p \to p_i$ を辺とする有向グラフを考えると、独立なサイクルの集合になる
- 周期が k ⇔ どのサイクルでも、サイクル上の x から k 回進むと x に戻っている

→ k はサイクル長の倍数

→ 最小の k は全サイクル長の最小公倍数 (LCM)

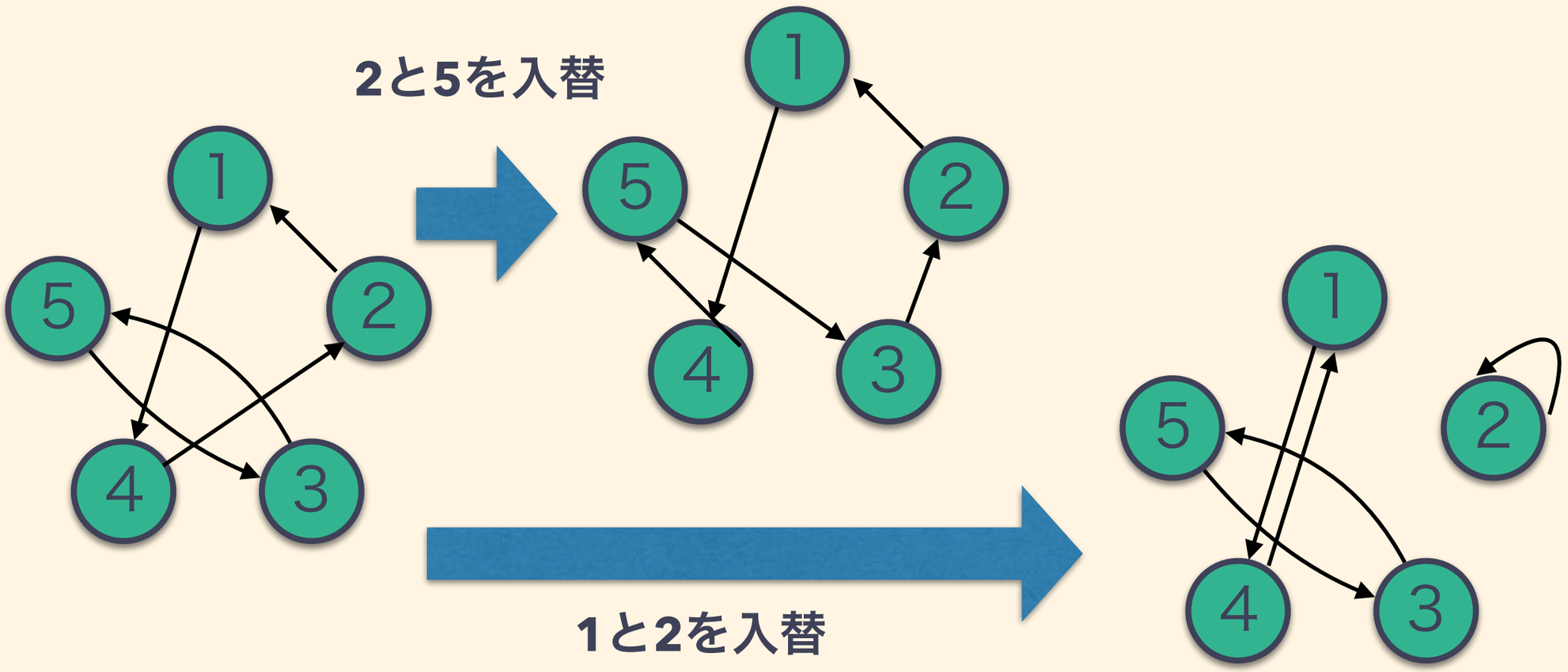


想定 TLE 解法

- Q回のクエリに対し、毎回 p の周期を計算
  - 周期の計算は O(N log N)
    - logN は LCM の計算時間
- 計算量: O(QNlogN) → TLE

考察: スワップの特徴

- スワップはサイクルの分離・併合しかしない
  - サイクル内の入替 → サイクルを2つに分離
  - サイクル間の入替 → サイクルを1つに併合



想定解法

- サイクルを管理し、クエリごとに長さが変わったサイクル (3つ) の長さを考えて LCM を更新する
  - サイクルの管理:

    サイクルを適当に切って列にして、平衡二分探索木で管理

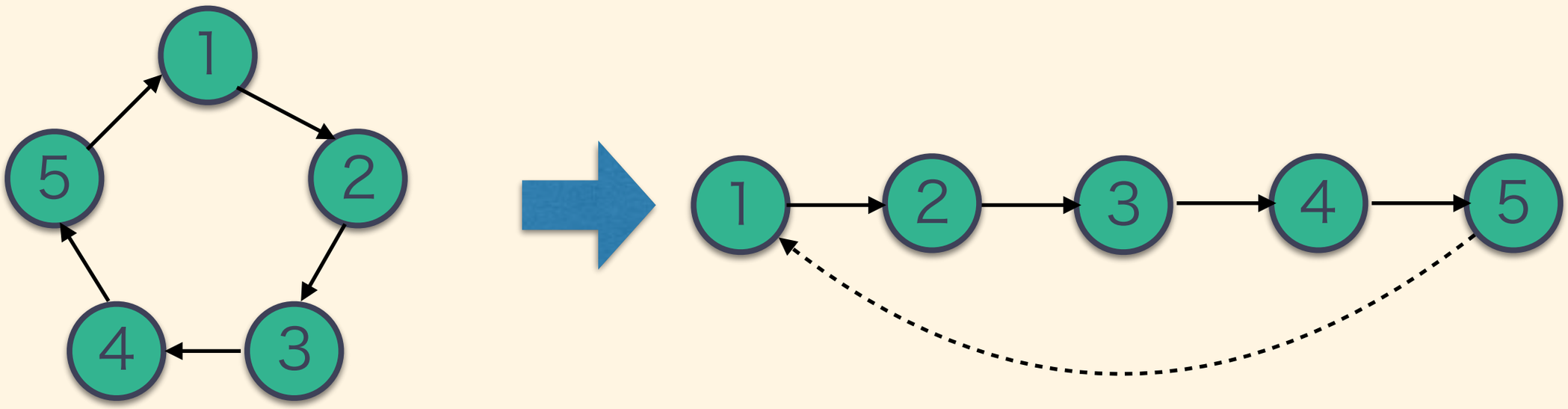
    計算量: O (logN)
  - LCMの計算:

    各サイクル長の素因数とその個数を素因数ごとに平衡二分探索木で管理。更新前後で個数最大が変わるときにLCMを更新

    計算量: $O(\log^2 N)$
- 全体で計算量 $O(Q \log^2 N)$

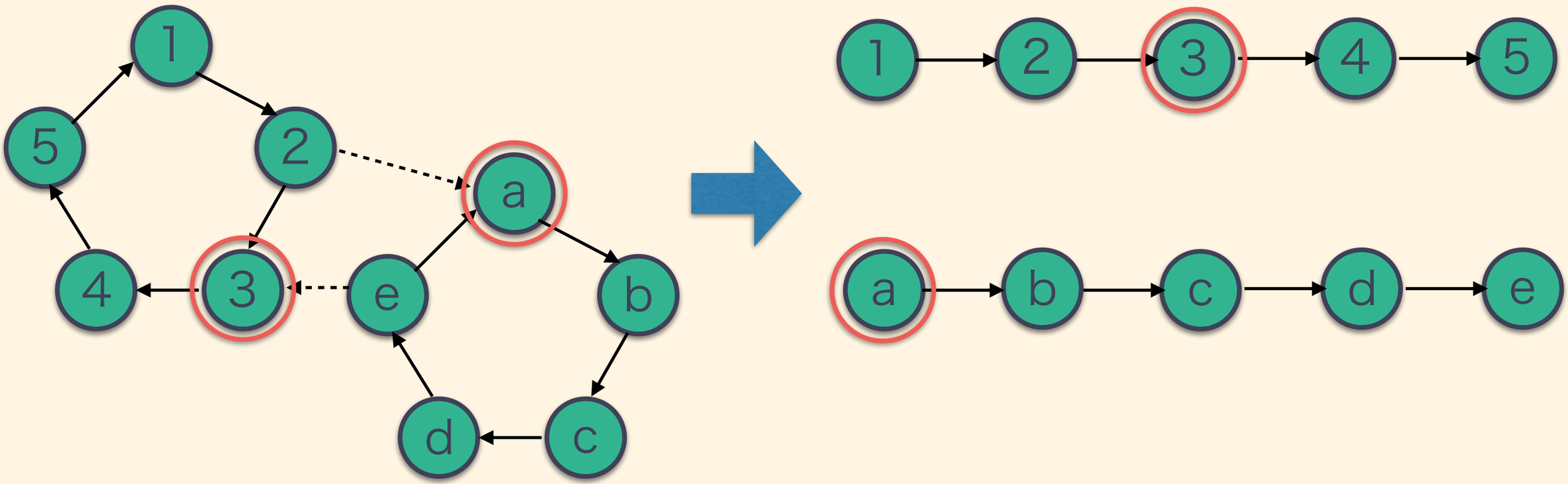
想定解法: サイクルの管理

- サイクルを適当なところで切り開いて列にする
- 列の高速な管理に定評のある平衡二分探索木に格納する
  - 列の merge/split 操作ができると楽 (後述)
  - c.f.) https://www.slideshare.net/iwiwi/2-12188757



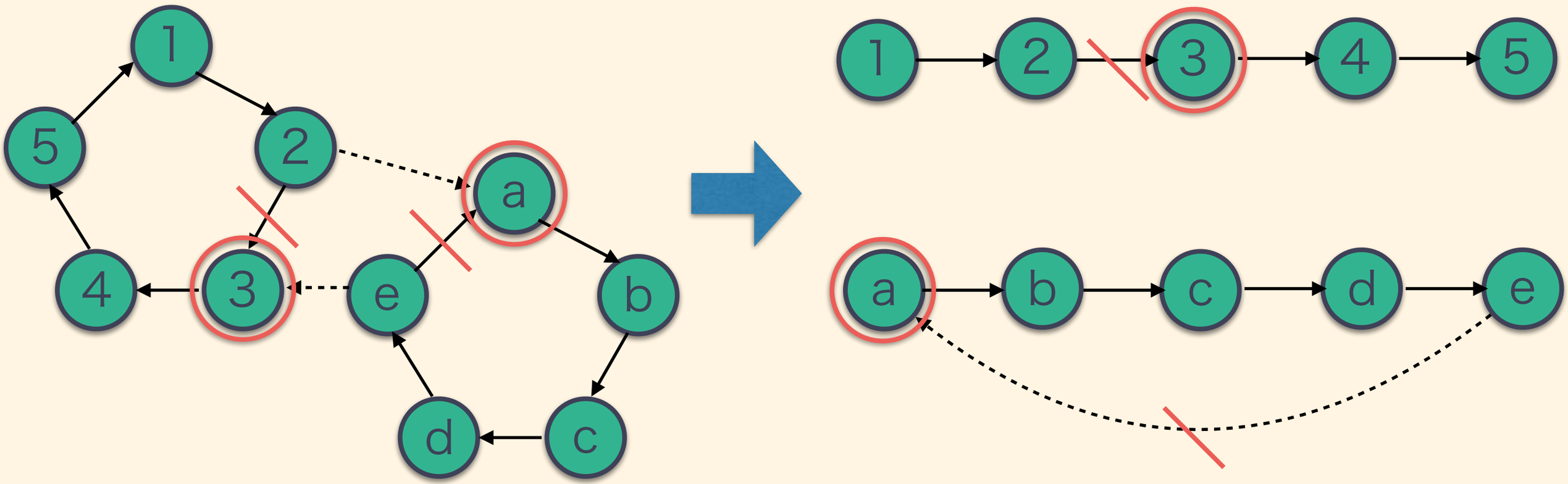
想定解法: サイクルの管理

- 併合の手順: (例: 3 と a のswap)
  1. **併合する要素の列状での位置を見つける**
  2. そこで split して、前後入れ替えて merge
  3. 2つの列を merge



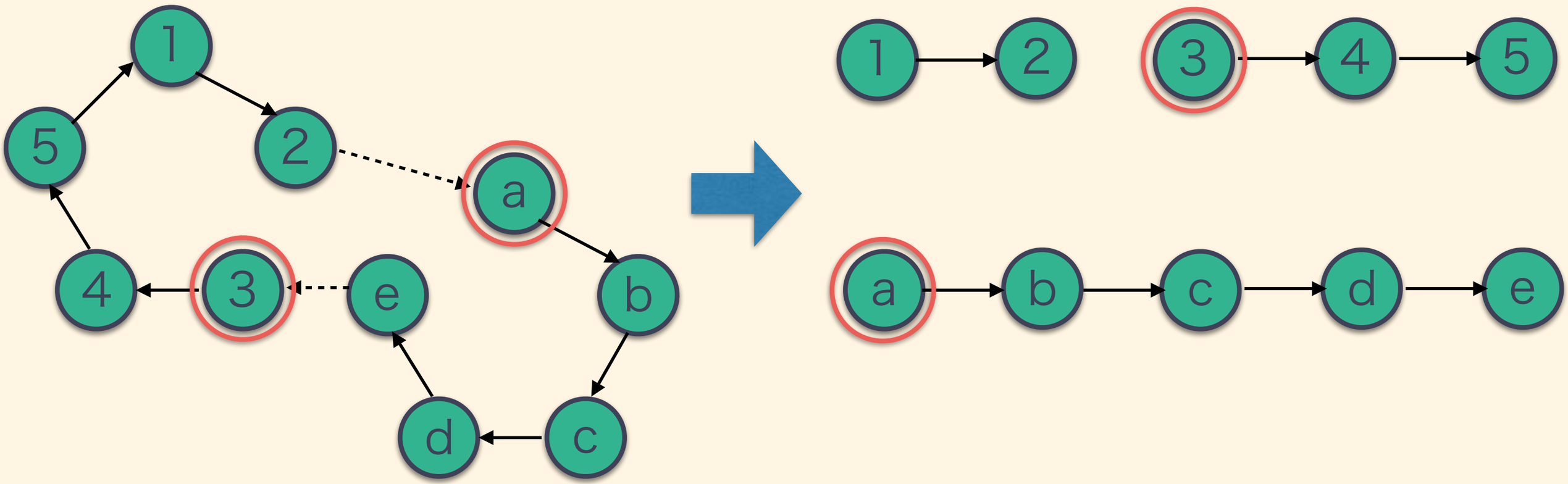
想定解法: サイクルの管理

- 併合の手順: (例: 3 と a のswap)
  1. 併合する要素の列状での位置を見つける
  2. **そこで split して**、前後入れ替えて merge
  3. 2つの列を merge



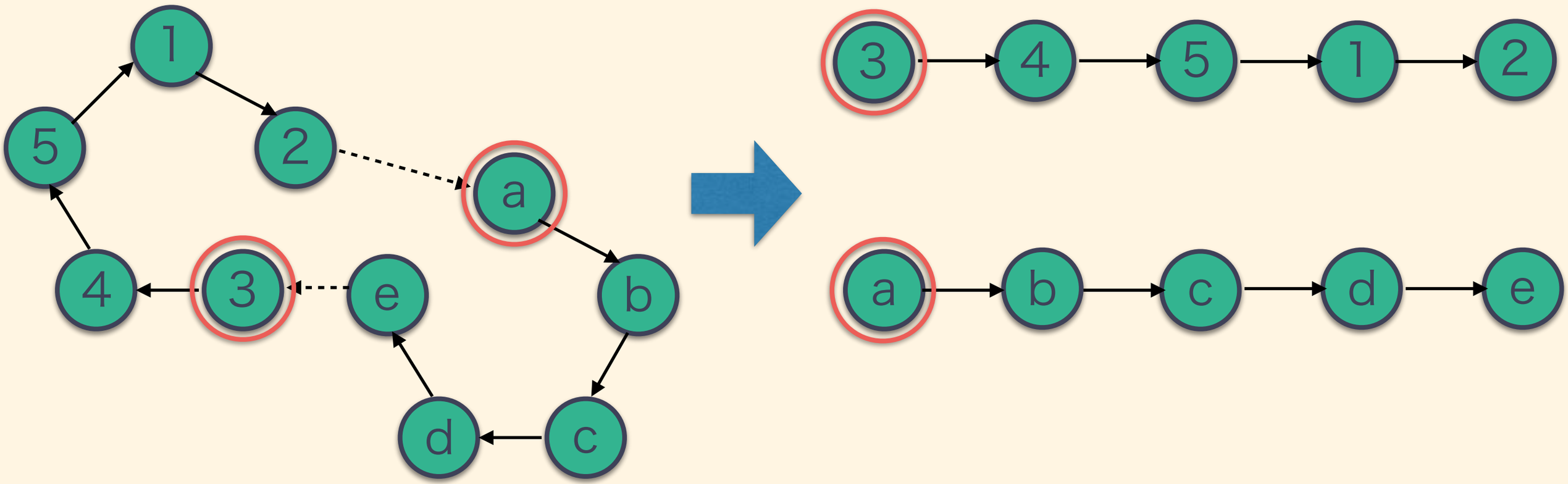
想定解法: サイクルの管理

- 併合の手順: (例: 3 と a のswap)
  1. 併合する要素の列状での位置を見つける
  2. **そこで split して**、前後入れ替えて merge
  3. 2つの列を merge



想定解法: サイクルの管理

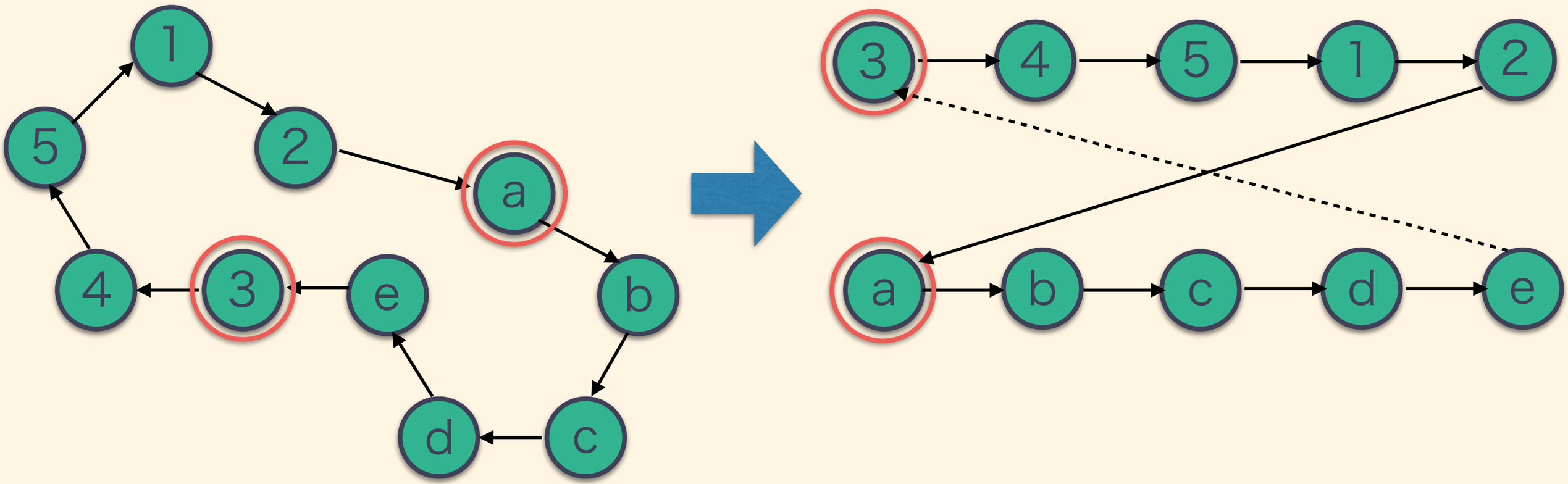
- 併合の手順: (例: 3 と a のswap)
  1. 併合する要素の列状での位置を見つける
  2. そこで split して、**前後入れ替えて merge**
  3. 2つの列を merge



想定解法: サイクルの管理

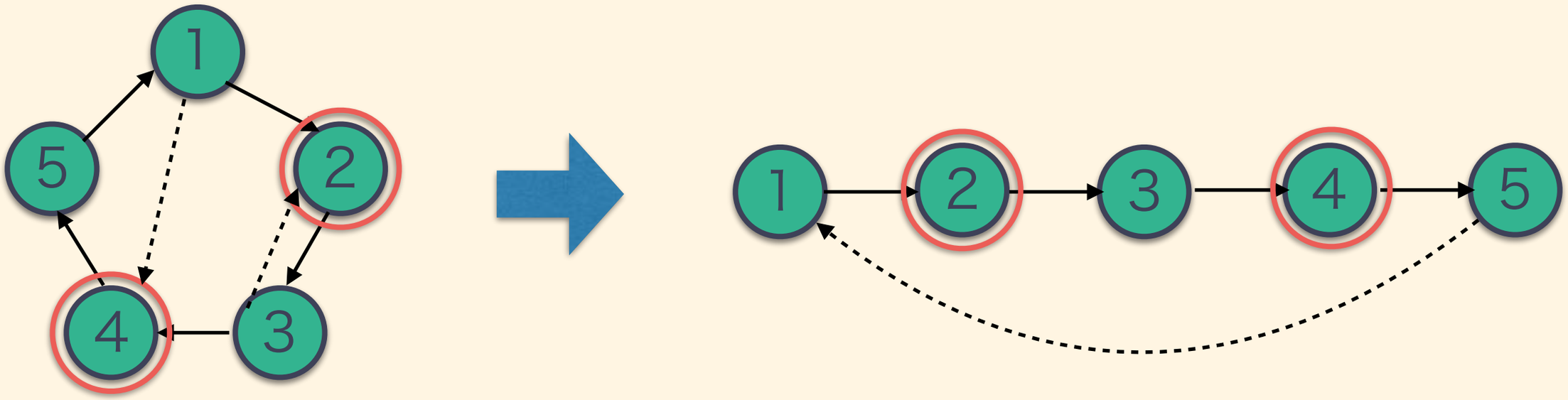
- 併合の手順: (例: 3 と a のswap)
  1. 併合する要素の列状での位置を見つける
  2. そこで split して、前後入れ替えて merge

3. 2つの列を merge



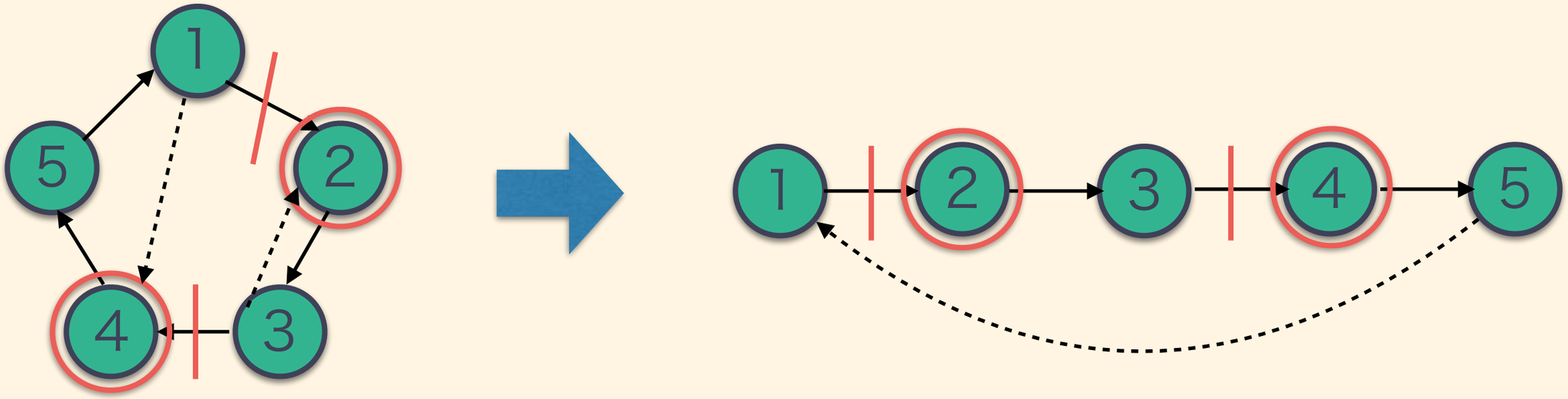
想定解法: サイクルの管理

- 分離の手順: (例: 2 と 4 のswap)
  1. **分離する要素の列状での位置を見つける**
  2. それらで split して3つに分ける
  3. 1つ目を3つ目の後に merge して 2つにする



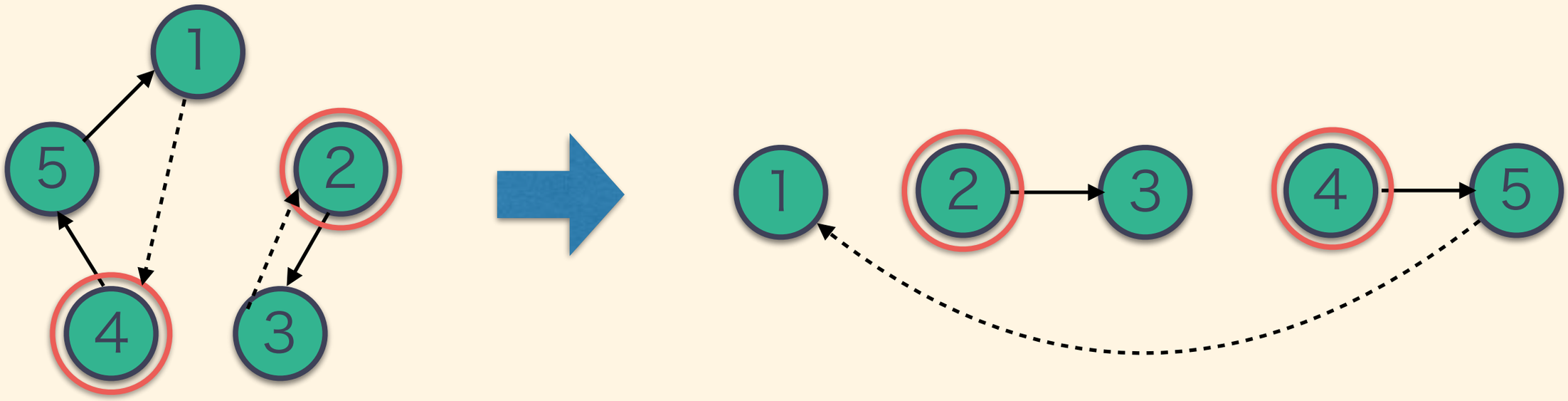
想定解法: サイクルの管理

- 分離の手順: (例: 2 と 4 のswap)
  1. 分離する要素の列状での位置を見つける
  2. **それらで split して3つに分ける**
  3. 1つ目を3つ目の後に merge して 2つにする



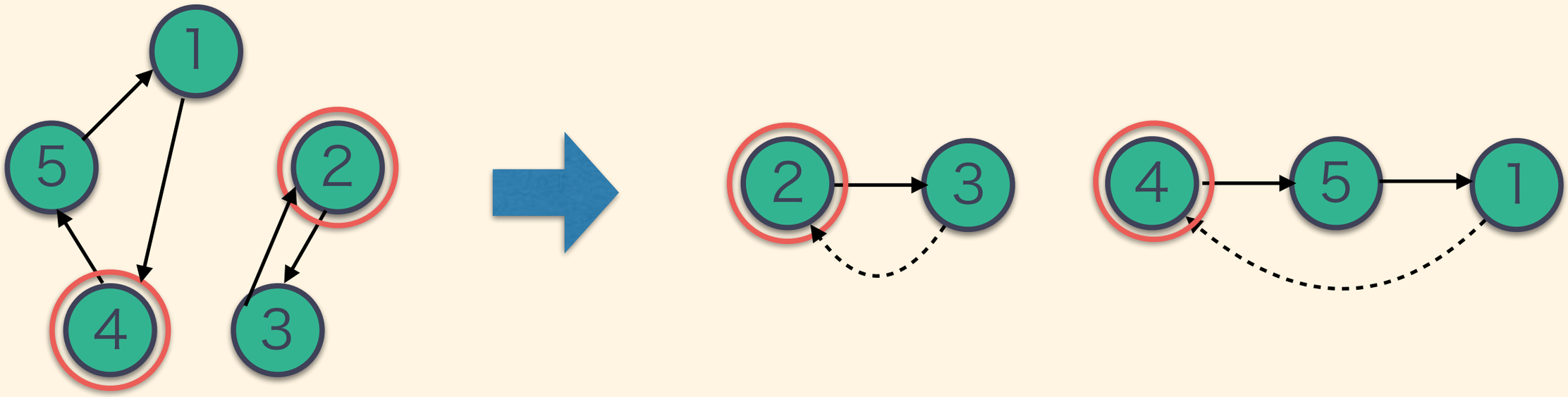
想定解法: サイクルの管理

- 分離の手順: (例: 2 と 4 のswap)
  1. 分離する要素の列状での位置を見つける
  2. **それらで split して3つに分ける**
  3. 1つ目を3つ目の後に merge して 2つにする



想定解法: サイクルの管理

- 分離の手順: (例: 2 と 4 のswap)
  1. 分離する要素の列状での位置を見つける
  2. それらで split して3つに分ける
  3. 1つ目を3つ目の後に merge して 2つにする



想定解法: サイクルの管理

- merge/split は O(logN)
- その他、さらに次の操作が必要
  - 2要素が同じ列に含まれるか？
    - 親を指すポインタも持つようにして、根まで辿って根が同じなら同じ列に含まれる (Union-Findっぽい) : O(logN)
  - 要素 x は列のどこにあるか？
    - 要素ごとのノードのポインタを覚えておき、そのノードが列の何番目の要素か親を辿りながら求める: O(logN)
- どれも O(logN) で処理できるので全体で O(logN)

想定解法: LCMの計算

- 解きたい問題:

  N以下の正整数の重複あり集合について、下記の(高速な) 演算をサポートせよ

  - 全要素の LCM (の mod 10^9+7) を求める
  - 正整数の追加
  - 正整数の削除
- これらがあれば、サイクルが減るときに削除、増える時に追加をして最後にLCMを求めればよい

想定解法: LCMの計算

- データ構造の概要:
  - 集合内の要素を素因数ごとに分けて保持する
    - 要素 k が素因数 p を x 個を持つ ($k \% p^x = 0$) の なら、バケット B_p に x を追加
- LCM = Π_p p $^{max\{Bp\}}$
  - B_p の管理には最大値の計算と整数の追加/削除が必要 → 平衡二分探索木
    - これは自作しなくても std::set などで十分

想定解法: LCMの計算

- データ構造の概要:
  - LCM の計算: 追加/削除時に更新して値を保持し続ける。保持している値を返すだけ: O(1)
  - x を追加:
    1. x を素因数分解をする。例えば osa_k法 (http://www.osak.jp/diary/diary_201310.html#20131017) で O(logN)
    2. 各素因数 p について B_p を更新、最大値を d 更新
       したなら LCM に $p^d \pmod{10^9+7}$ を掛ける:
       各更新が O(logN)、素因数が O(logN) 個で $O(\log^2 N)$
  - x を削除: 追加同様に $O(\log^2 N)$。ただし p の逆元が必要

想定解法

- サイクルを管理し、クエリごとに長さが変わったサイクル (3つ) の長さを考えて LCM を更新する
  - サイクルの管理: O (logN)
  - LCMの計算: O(log²N)
- 全体で計算量 O(Q log²N)

Writer 解

- darsein: 289 行 6169 bytes (C++)
- not: 185 行 4936 bytes (C++)
- tubo: 231 行 5469 bytes (C++)

統計情報

- AC / submissions
  - 3 / 6 (50.00%)
- First Acceptance
  - ainta (200:25)