HCPC 新歓2024
Box and Ball(Diff: 981)

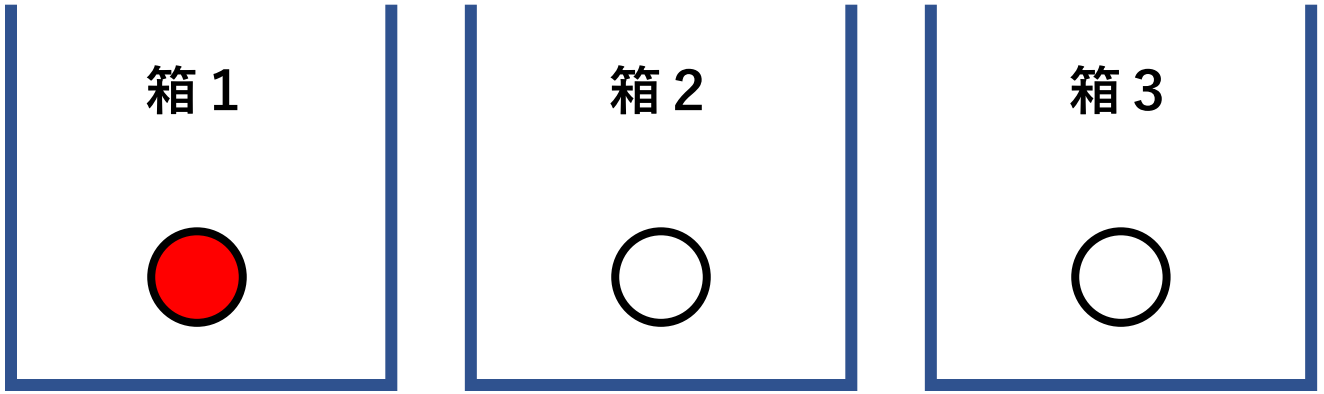
解説：Slephy

問題文（引用）

- N 個の箱があります。 箱は 1 から N まで番号が振られています。
  最初、1 番目の箱には赤いボールが 1 個入っています。
  また、2〜N 番目の箱には白いボールが 1 個ずつ入っています。
- 高橋君は M 回の操作を順に行います。 i 回目の操作では、
  x_i 番目の箱から適当なボールを 1 個選び、それを y_i 番目の箱へ移します。
- 高橋君がすべての操作を終えた後、赤いボールが入っている可能性のある箱は何個か求めてください。

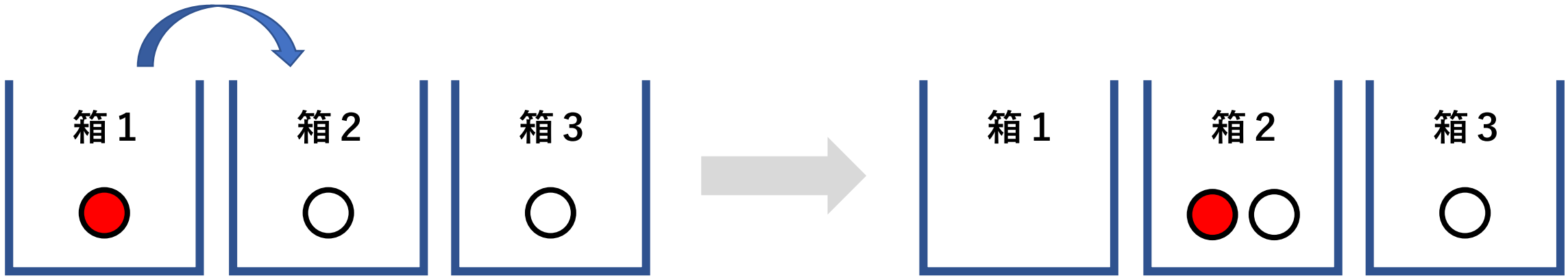
入力例１を考えてみよう

入力例１
３２
１２
２３



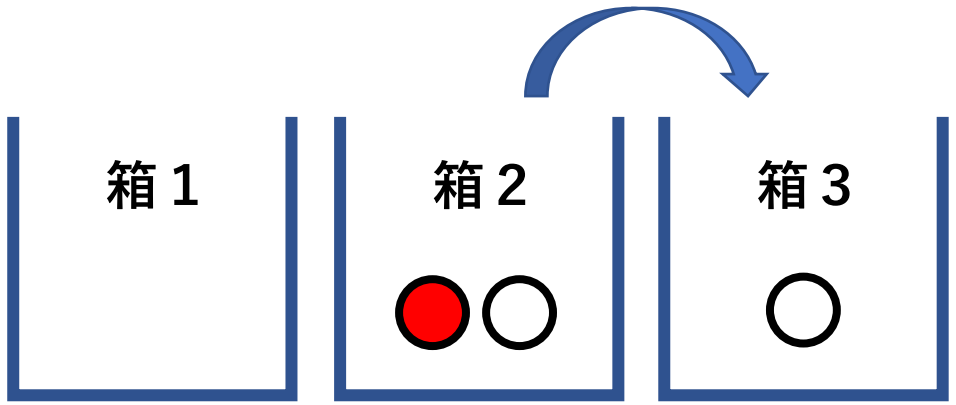
移動1回目

箱1→箱2

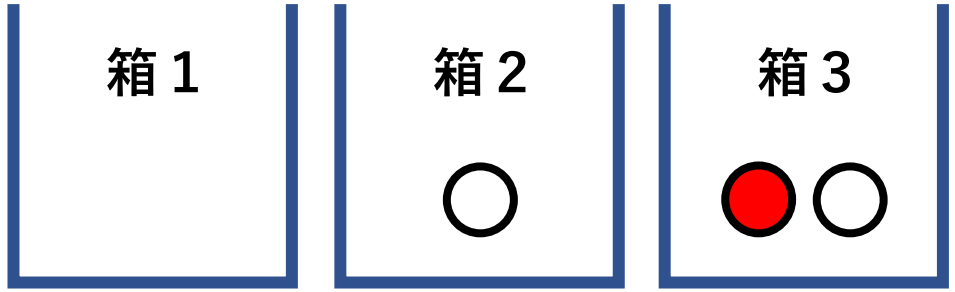


移動2回目

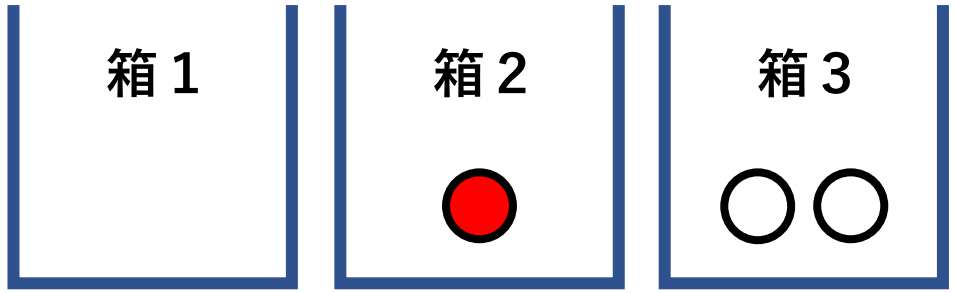
箱2→箱3



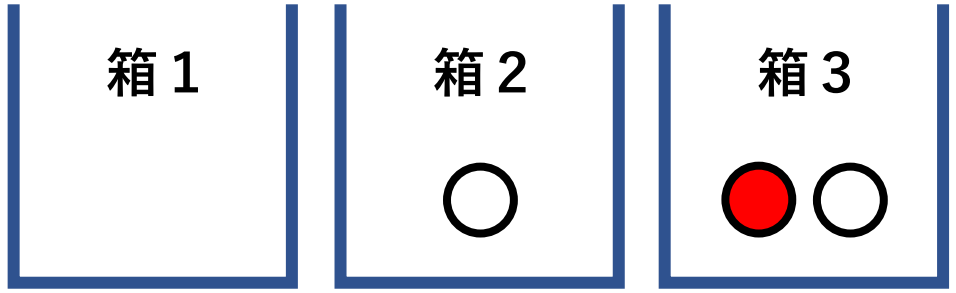
赤を移動したとき

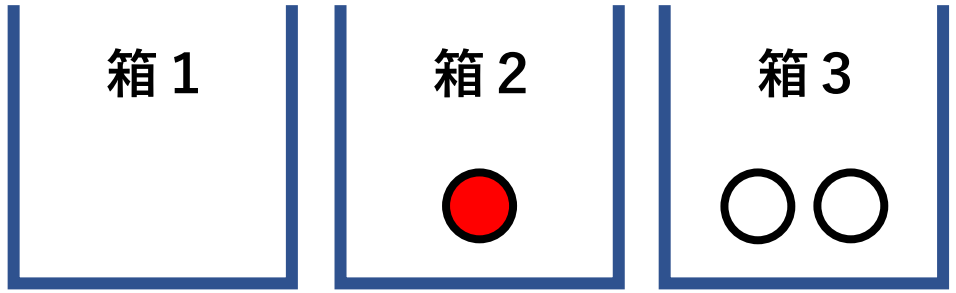


白を移動したとき



入力例１の答え





最終的に考えられる状態は左に示す2つのみ。
赤いボールが入っている可能性がある箱は
箱2と箱3の2つ

$$\therefore \mathrm{Answer} = 2$$

ここからが本題

この問題の難しさ

制約が大きい！

$$2 \leq N \leq 10^5, \quad 2 \leq M \leq 10^5$$

先ほどのように終了状態を全通り列挙しては**到底間に合わない**

これでは**指数オーダー $O(2^N)$**かかってしまう

解法のポイント

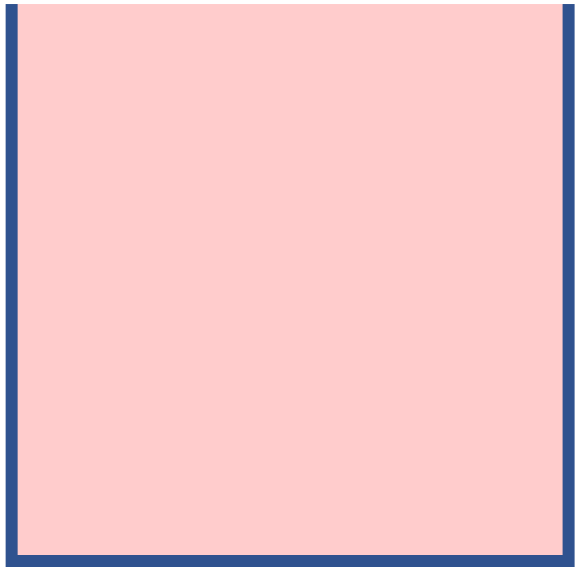
- 最終的に問題になるのは
  「その箱に赤いボールが入っている可能性があるか否か」

- 具体的にどの箱に赤いボールが入っているかを気にせず、
  「赤いボールが入っている可能性」の遷移に着目して考える

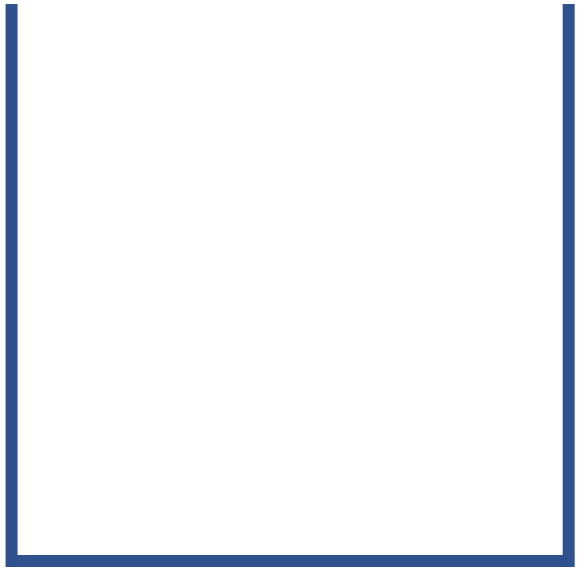
- これにより、i 回操作が終わった時点で有り得るすべての
  状態をまとめて表現できる（うれしい）

まずは定義

赤いボールが入っている可能性がある箱

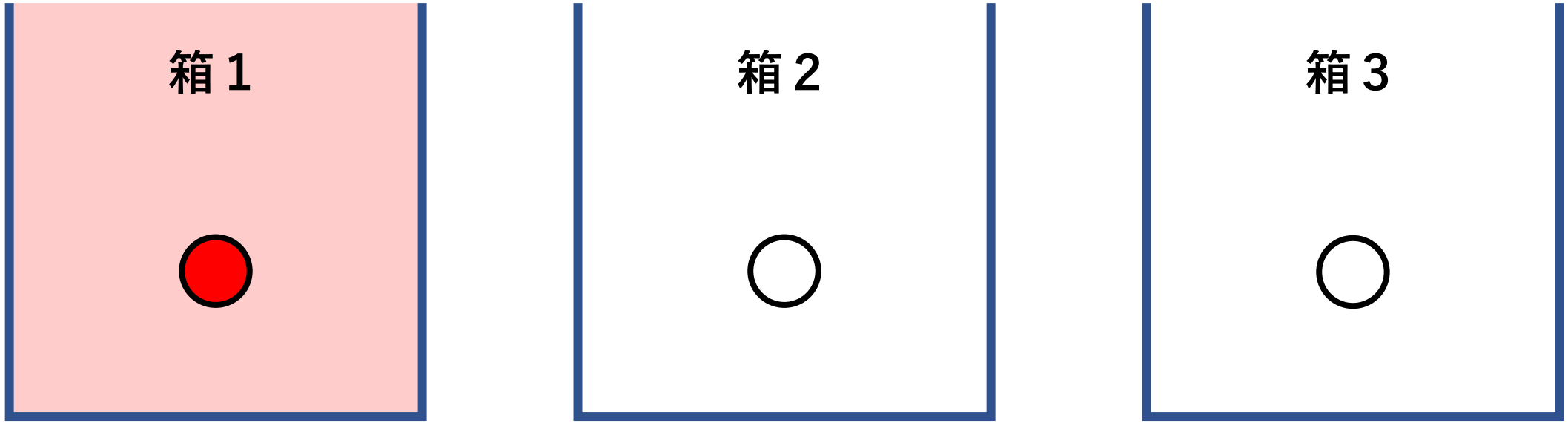


〃可能性がない箱



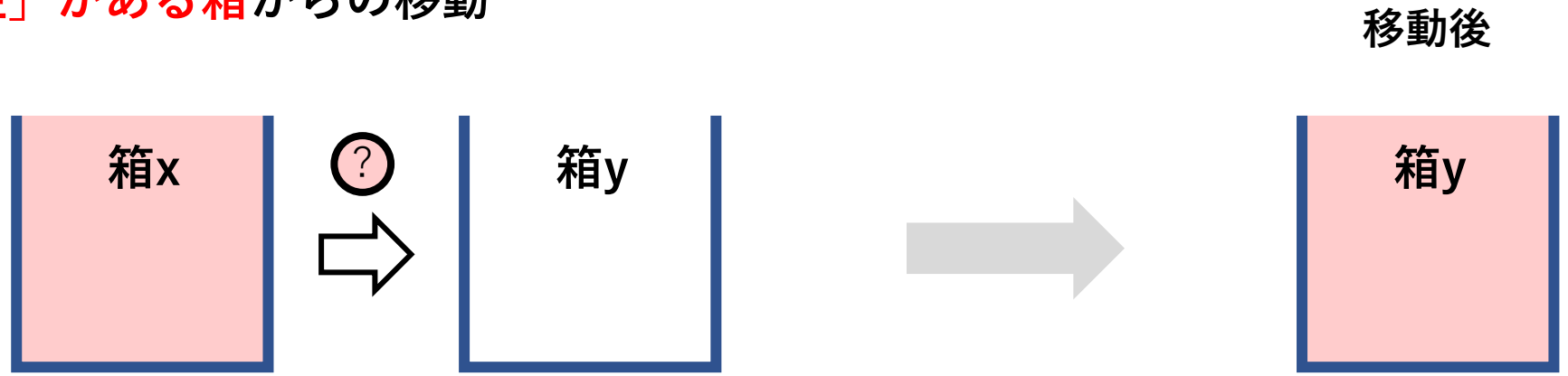
以下、「赤いボールが入っている可能性」を単に「可能性」と書きます。

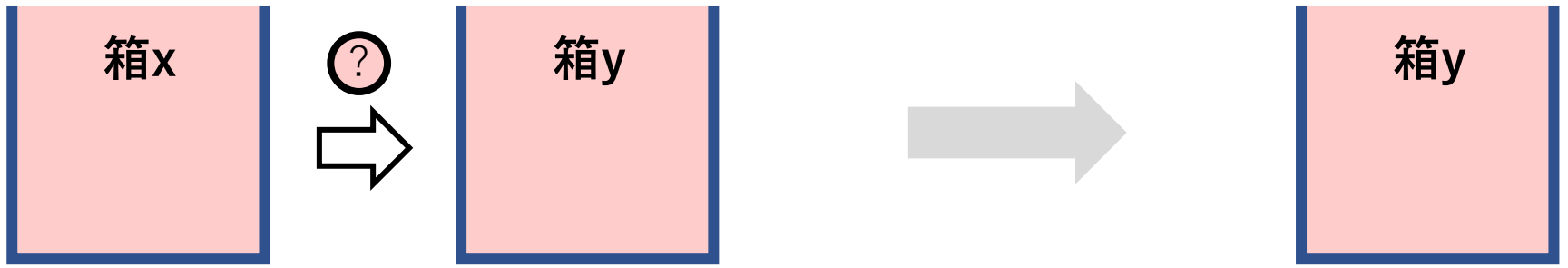
初期状態における可能性



可能性の遷移（移動先の箱①）

「可能性」がある箱からの移動



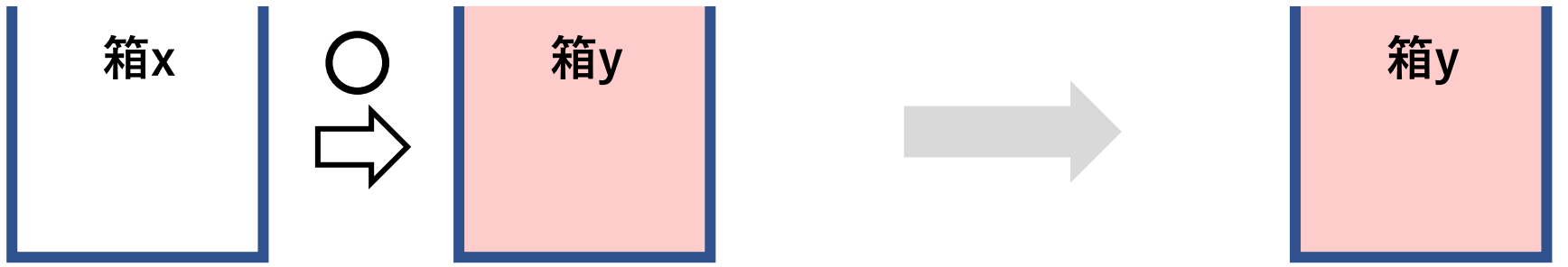


可能性が
伝播する

可能性の遷移（移動先の箱②）

「可能性」がない箱からの移動

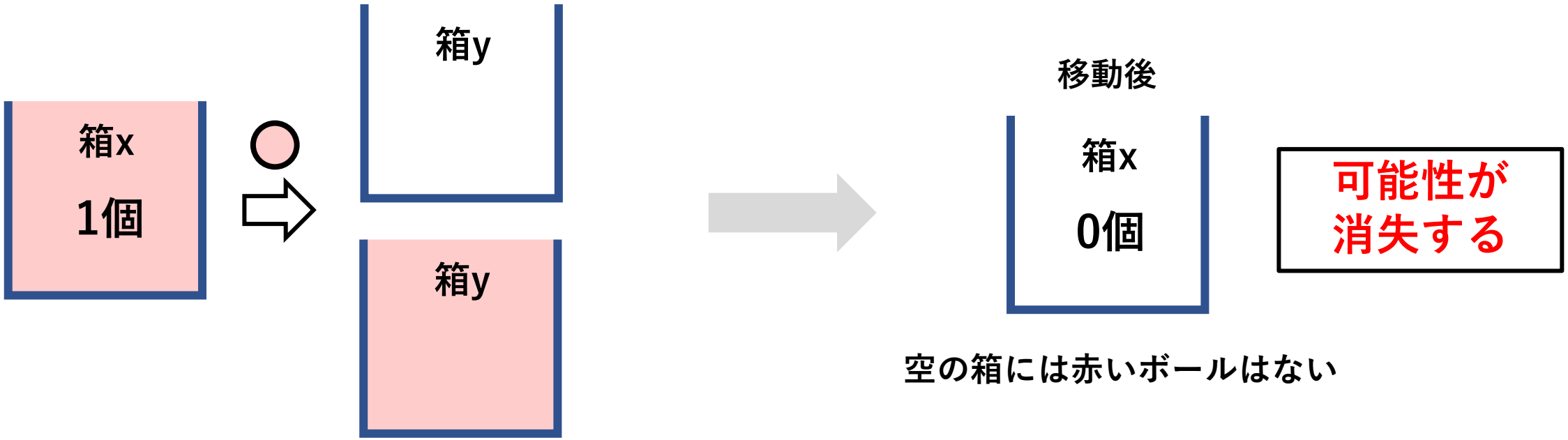
可能性が
伝播しない



可能性の遷移（移動元の箱）

移動元の箱の状態は**基本的に不変**

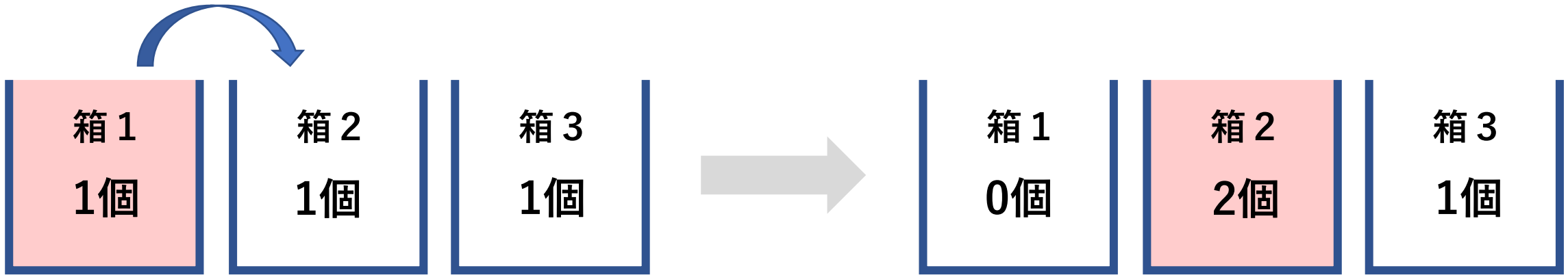
移動元の箱に**ボールが１個**のときだけ状態が変化し得る



実際に適用してみよう！

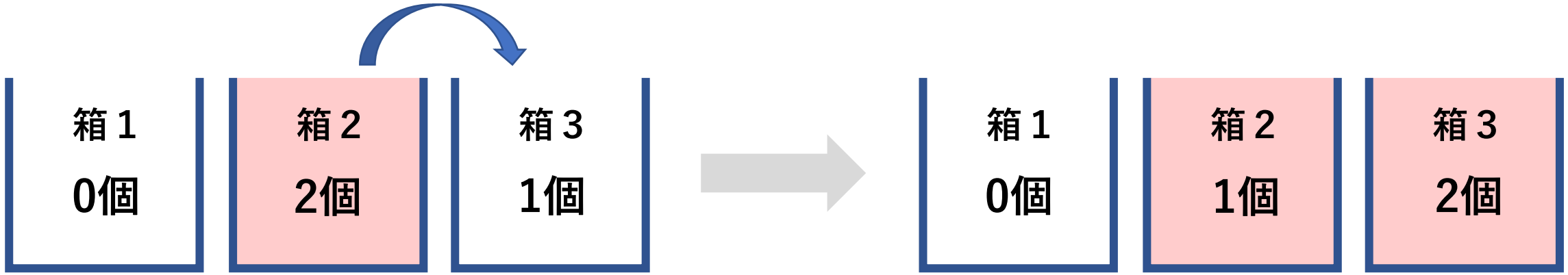
入力例１（想定解）

箱1→箱2



入力例１（想定解）

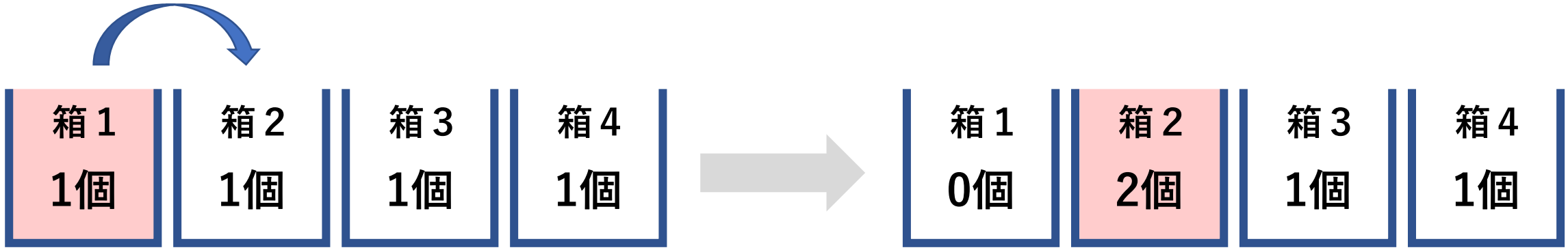
箱2→箱3



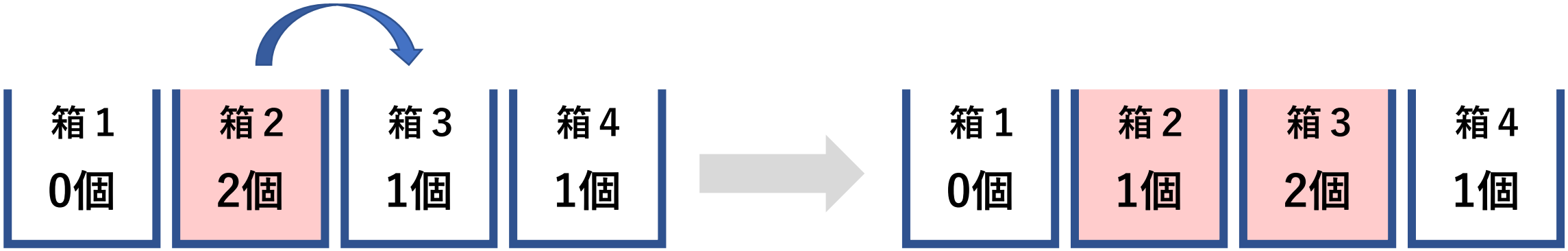
答えは２

入力例３（想定解）

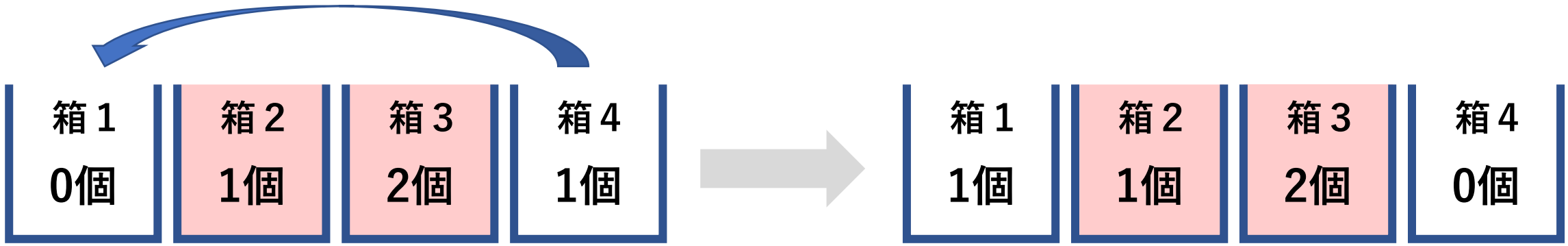
入力例３も同様にやってみよう



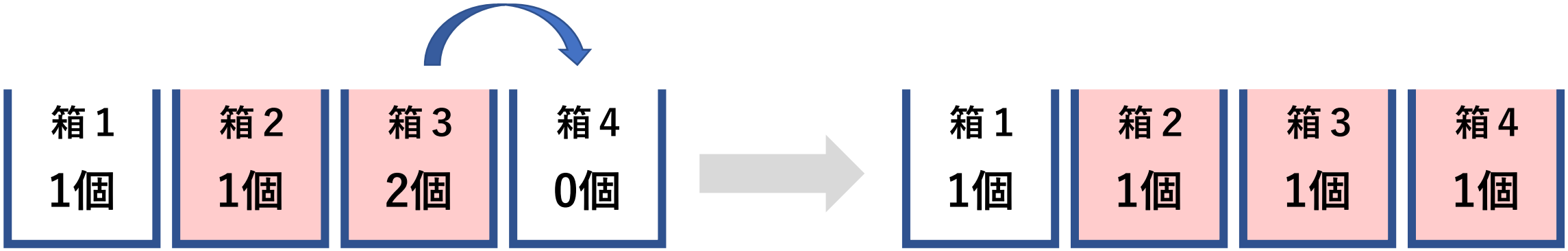
入力例３（想定解）



入力例３（想定解）



入力例３（想定解）



答えは３

このアルゴリズムの計算量

- 1回の操作は定数時間で終わる。$O(1)$
- それがM回あるので、$O(M)$
- 配列の初期化に$O(N)$かかるので、合わせて $O(N+M)$
- $2 \leq N \leq 10^5$, $2 \leq M \leq 10^5$ なので間に合う！

実装例（Python）

```python
# 入力
n, m = map(int, input().split())

# 配列の初期化(0-indexed)
ballNum = [1] * n
mayRed = [False] * n
mayRed[0] = True

for i in range(m):
    x, y = map(int, input().split())
    # 0-indexedに変換
    x -= 1
    y -= 1

    # 移動先の箱へ可能性を伝播させる
    if mayRed[x]: mayRed[y] = True

    # ボールの個数を変更
    ballNum[x] -= 1
    ballNum[y] += 1
    # 移動元の箱が空になったら、可能性をfalseにする
    if ballNum[x] == 0: mayRed[x] = False

# 赤いボールが入っている可能性がある箱の数を出力する
print(mayRed.count(True))
```

参考提出

- **Python(220ms)**
  https://atcoder.jp/contests/agc002/submissions/33767850
- **C++(49ms)**
  https://atcoder.jp/contests/agc002/submissions/33780398