

# ABC303 B

## $x[j]--$ を図で理解する

# この資料のゴール

- 入力が 1-index, 配列添字が 0-index というズレを理解する.
- $x[j]--$  が何をしているかを具体例で理解する.
- なぜ変換しないと危険か（範囲外アクセス）を理解する.

# 1-index と 0-index の対応

問題文の人番号（1-index）と、配列添字（0-index）は次のように対応します。

人番号（入力）	1	2	3	4
添字（配列）	0	1	2	3

つまり「各値を 1 減らす」と対応が取れます。

$x[j]--$  は  $j$  番目の値だけを 1 減らす

```
for (int j = 0; j < n; j++) {  
    cin >> x[j]; // 例: 1 3 2 4  
    x[j]--; // 各要素を 1 減らして 0-index 化  
}
```

例として、入力が 1 3 2 4 の場合:

j	読み込み直後 $x[j]$	$x[j]--$ 後 $x[j]$	備考
0	1	0	1 番の人 → 添字 0
1	3	2	3 番の人 → 添字 2
2	2	1	2 番の人 → 添字 1
3	4	3	4 番の人 → 添字 3

最終的に  $x = [0, 2, 1, 3]$  になります。

# なぜ必要か: adjacent [a] [b] は 0-index 前提

## 変換あり (正しい)

- $x = [0, 2, 1, 3]$
- 隣接ペア:  $(0, 2), (2, 1), (1, 3)$
- $\text{adjacent}[a][b]$  の  $a, b$  はすべて  $0..n-1$  に収まる.

## 変換なし (危険)

- $x = [1, 3, 2, 4]$  のまま
- 例えば  $\text{adjacent}[2][4]$  が出る
- $n=4$  なら有効添字は  $0..3$
- 4 は範囲外アクセス.

# よくある誤解

「配列に入れたから 1-index になる」わけではありません。

- 1-index / 0-index は 入力仕様 で決まる。
- C++ の vector 添字は常に 0-index。
- 入力が  $1..n$  なら -1 変換が必要。
- 入力が最初から  $0..n-1$  なら不要。

// 入力仕様が 0-index の問題なら

```
cin >> x[j];
// x[j]--; は不要
```

## 確認問題

**Q.**  $n=5$ , 入力が 2 5 1 4 3 のとき,  $x[j]--$  後の配列は?

**A.** 1 4 0 3 2

これで adjacent[a][b] の添字は常に 0..4 に収まります.

覚えることは 1 行

入力が 1-index なら, 配列で使う前に 0-index に合わせる.