Schuhe anordnen

Adnan besitzt das grösste Schuhgeschäft in Baku. Im Geschäft ist soeben eine Kiste mit n Paaren von Schuhen angekommen. Jedes Paar besteht aus zwei Schuhen gleicher Grösse: ein linker und ein rechter Schuh. Adnan hat alle $2n$ Schuhe in einer Reihe aufgestellt und zwar mit $2n$ **Positionen**, die von 0 bis $2n-1$ von links nach rechts durchnummeriert sind.

Adnan möchte die Schuhe in einer **gültigen Anordnung** neu aufstellen. Eine Anordnung ist genau dann gültig, wenn für jedes i $(0 \leq i \leq n-1)$ die folgenden Bedingungen gelten:

- Die Schuhe an den Positionen $2i$ and $2i+1$ haben dieselbe Grösse.
- Der Schuh an der Position $2i$ ist ein linker Schuh.
- Der Schuh an der Position $2i+1$ ist ein rechter Schuh.

Um dieses Ziel zu erreichen, kann Adnan eine Folge von Vertauschungen durchführen. Bei jeder Vertauschung wählt er zwei Schuhe, die zu diesem Zeitpunkt **benachbart** sind, und vertauscht sie (das heisst: Er hebt sie auf und stellt jeden Schuh auf die vorherige Position des anderen). Zwei Schuhe sind benachbart, wenn sich ihre Positionen um eins unterscheiden.

Bestimme die minimale Anzahl von Vertauschungen, die Adnan benötigt, um die Schuhe in einer gültigen Anordnung aufzustellen.

Implementierung

Implementiere die folgende Funktion:

```
int64 count_swaps(int[] S)
```

- S: ist ein Array von $2n$ Integers. Für jedes i $(0 \leq i \leq 2n-1)$, ist $|S[i]|$ ein Wert ungleich null, der die Schuhgrösse jenes Schuhs beschreibt, welcher zu Beginn an der Position i plaziert ist. Die Schuhgrösse übersteigt n nicht. Wenn $S[i] < 0$ ist, dann ist der Schuh an Position i ein linker Schuh, sonst ist er ein rechter Schuh.
- Diese Funktion soll die minimale Anzahl von Vertauschungen (benachbarter Schuhe) zurückgeben, die nötig ist, um eine gültige Anordnung zu erreichen.

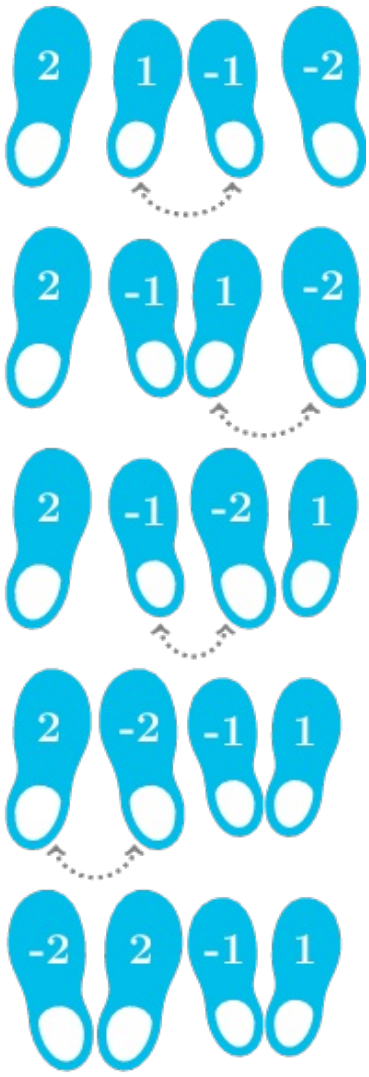
Beispiele

Beispiel 1

```
count_swaps([2, 1, -1, -2])
```

Adnan kann eine gültige Anordnung mit 4 Vertauschungen erreichen.

Zum Beispiel: Er kann zuerst die Schuhe 1 und -1 vertauschen, dann 1 und -2, dann -1 und -2 und zum Schluss 2 and -2. Dann erreicht er die folgende gültige Anordnung: $[-2, 2, -1, 1]$. Es ist nicht möglich, irgendeine gültige Anordnung mit weniger als 4 Vertauschungen zu erreichen. Daher muss die Funktion 4 zurückgeben.



Beispiel 2

Im folgenden Beispiel haben alle Schuhe die gleiche Grösse:

```
count_swaps([-2, 2, 2, -2, -2, 2])
```

Adnan kann die Schuhe an den Positionen 2 und 3 vertauschen, um die gültige

Anordnung $[-2, 2, -2, 2, -2, 2]$ zu erreichen. Die Funktion sollte daher 1 zurückgeben.

Beschränkungen

- $1 \leq n \leq 100\,000$
- Für jedes i ($0 \leq i \leq 2n-1$), $1 \leq |S[i]| \leq n$. Hier bezeichnet $|x|$ den Absolutwert von x.
- Es gibt immer eine Folge von Vertauschungen mit der eine gültige Anordnung von Schuhen erreicht wird.

Subtasks

1. (10 Punkte) $n = 1$
2. (20 Punkte) $n \leq 8$
3. (20 Punkte) Alle Schuhe haben die gleiche Grösse.
4. (15 Punkte) Alle Schuhe an den Positionen $0, \ldots, n-1$ sind linke Schuhe, und alle Schuhe an den Positionen $n, \ldots, 2n-1$ sind rechte Schuhe. Zusätzlich haben für jedes i ($0 \leq i \leq n-1$) die Schuhe an den Positionen i und $i+n$ dieselbe Grösse.
5. (20 Punkte) $n \leq 1000$
6. (15 Punkte) keine zusätzlichen Beschränkungen.

Beispiel-Grader

Der Beispiel-Grader liest die Eingaben im folgenden Format:

- Zeile 1: n
- Zeile 2: $S[0]$ $S[1]$ $S[2]$ $\ldots$ $S[2n-1]$

Der Beispiel-Grader gibt eine einzige Zeile mit dem Wert von `count_swaps` zurück.