Cambia i semafori (`semafori`)

Lavinia, dopo anni di studi e sacrifici, ha finalmente ottenuto il lavoro dei suoi sogni: fare il vigile urbano.

Lavinia vive nella metropoli di Correzzana e ha il compito di regolare il traffico su una strada con N semafori in fila. L'i-esimo semaforo è verde se il numero di secondi di attivazione S_i è pari e rosso se è dispari.

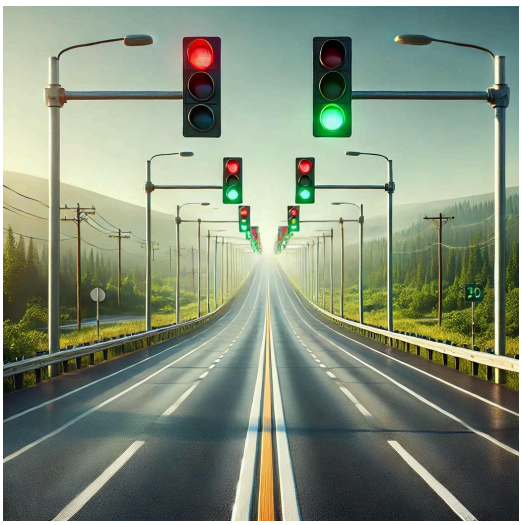

Figura 1: *Via Principale* a Correzzana.

Per garantire una circolazione fluida, Lavinia **non** vuole che ci siano due semafori consecutivi dello stesso colore.

Lavinia si ricorda i numeri di attivazione di tutti i semafori ma si accorge che la circolazione potrebbe non essere fluida come vorrebbe. Ultimamente è molto impegnata a impedire che le automobili a Correzzana percorrano le rotonde in senso opposto, quindi ha bisogno del tuo aiuto per rendere la circolazione fluida.

Modificare i secondi di attivazione di un semaforo è un'operazione delicata e per questo vorrebbe farlo il minor numero di volte possibile.

Il tuo compito è calcolare il numero minimo di modifiche necessarie affinché la circolazione sia fluida.

Implementazione

Dovrai sottoporre un unico file, con estensione `.cpp`, `.py`, `.cs` o `.java`.

> ⇨ Tra gli allegati di questo task troverai dei template `semafori.*` con un esempio di implementazione.

Dovrai implementare la seguente funzione:

C++	`int traffico(int N, vector<int> S);`
Python	`def traffico(N: int, S: list[int]) -> int:`

Java	`public static int traffico(int N, int[] S)`
C#	`public static int traffico(int N, int[] S)`

- L'intero N rappresenta il numero di semafori.
- L'array S, indicizzato da 0 a $N-1$, contiene i numeri di attivazione dei semafori.
- La funzione `traffico` dovrà restituire un intero: la risposta al problema, eventualmente 0 se la circolazione sia già fluida.

Grader di prova

Nella directory relativa a questo problema è presente una versione semplificata del grader usato durante la correzione, che puoi usare per testare le tue soluzioni in locale. Il grader di esempio legge i dati da `stdin`, chiama la funzione che devi implementare e scrive su `stdout`, secondo il seguente formato.

Il file di input è composto da 2 righe, contenenti:

- Riga 1: l'intero N.
- Riga 2: Gli interi $S_0, S_1, ..., S_{N-1}$.

Il file di output è composto da un'unica riga, contenente il valore restituito dalla funzione `traffico`.

Assunzioni

- $1 \leq N \leq 10^6$.
- $1 \leq S_i \leq 10^9$ per ogni $0 \leq i < N$.

Assegnazione del punteggio

Il tuo programma verrà testato su diversi test case raggruppati in subtask. Per ottenere il punteggio relativo ad un subtask, è necessario risolvere correttamente tutti i test che lo compongono.

- **Subtask 1 [0 punti]**: Casi d'esempio.
- **Subtask 2 [13 punti]**: S_i è pari per ogni $0 \leq i < N$
- **Subtask 3 [37 punti]**: $S_i = 1$ o $S_i = 2$ per ogni $0 \leq i < N$
- **Subtask 4 [50 punti]**: Nessuna limitazione aggiuntiva.

Esempi di input/output

stdin	stdout
5 1 2 4 2 1	1
8 2 4 6 8 10 12 14 16	4

Spiegazione

Nel **primo caso d'esempio**, è sufficiente cambiare i secondi di attivazione del semaforo in posizione 2 in un numero dispari, ad esempio 3.

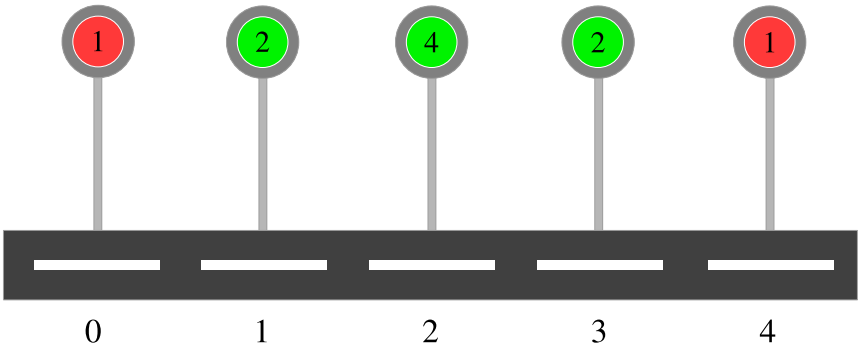

Figura 2: Cambiando il semaforo al centro da 2 a 3, la circolazione diventa fluida.

Nel **secondo caso d'esempio** è ottimale modificare i semafori in posizione $1, 3, 5$ e 7 in numeri dispari.