Parentesi k-bilanciate

Definizione

Data una sequenza di parentesi aperte `(` e chiuse `)`, essa è detta **k-bilanciata** se, aggiungendo `k` parentesi aperte all'inizio e `k` parentesi chiuse alla fine della sequenza, si ottiene una sequenza bilanciata.
Ricordiamo che una sequenza di parentesi è bilanciata se ogni parentesi aperta trova una corrispondente parentesi chiusa ed è correttamente annidata.

Esempi

- La sequenza `)()(` è 1-bilanciata: aggiungendo `(` all'inizio e `)` alla fine si ottiene `()()()`, che è bilanciata.
- La sequenza `)()())((` è 2-bilanciata ma non 1-bilanciata.
- La sequenza `()())` non è k-bilanciata per alcun $k \geq 0$.

Obiettivo

Progettare un algoritmo che, data una sequenza lunga n di parentesi, calcoli il minimo valore di k per cui la sequenza è k-bilanciata, o restituisca $+\infty$ se non è possibile bilanciarla per nessun k.

Idea della Soluzione

Interpretiamo le parentesi come incrementi e decrementi di un contatore:

- `(` incrementa il bilancio di 1.
- `)` decrementa il bilancio di 1.

Definiamo:

- `balance(i)` come il bilancio dopo aver letto i primi i simboli.
- `balance(n)` come il bilancio alla fine della sequenza.

Osservazioni

1. Se `balance(n) ≠ 0`, allora non sarà possibile bilanciare la sequenza aggiungendo lo stesso numero di `(` e `)` (poiché l'aggiunta di pari quantità di entrambe non modifica lo sbilanciamento totale). In questo caso, il risultato è $+\infty$.

2. Se `balance(n) = 0`, la sequenza contiene lo stesso numero di `(` e `)`. Potrebbe però essere "sottobilanciata" in certi punti, cioè potrebbe andare in negativo durante la scansione. Consideriamo `min_balance` = il valore minimo raggiunto da `balance(i)` durante la scansione.
    - Se `min_balance ≥ 0`, la sequenza non ha mai avuto deficit di parentesi aperte: è già 0-bilanciata.
    - Se `min_balance < 0`, per evitare che la sequenza "scenda sotto zero", dobbiamo aggiungere almeno `-min_balance` parentesi aperte all'inizio (e di conseguenza `-min_balance` parentesi chiuse alla fine). Quindi `k = -min_balance`.

Algoritmo

1. Inizializza due variabili:
    - `balance = 0`
    - `min_balance = 0`
2. Scansiona la sequenza di parentesi da sinistra a destra:
    - Per ogni simbolo:
        - Se è `(`, `balance++`.
        - Se è `)`, `balance--`.
    - Aggiorna `min_balance = min(min_balance, balance)`.
3. Alla fine della scansione:
    - Se `balance ≠ 0`, restituisci $+\infty$.
    - Altrimenti, se `min_balance ≥ 0`, restituisci `0`.
    - Altrimenti, restituisci `-min_balance`.

Complessità

- Tempo: $O(n)$ per scansionare la sequenza.
- Spazio: $O(1)$ in quanto utilizza un numero costante di variabili ausiliarie.

Correttezza

- La condizione `balance(n) = 0` è necessaria per la bilanciabilità.
- Il valore `min_balance` misura quanto "in negativo" la sequenza va: se non va mai sotto zero, è già bilanciata.
- Se va sotto zero, aggiungere `-min_balance` parentesi iniziali la "rialza" di quel tanto da non andare mai sotto zero, e aggiungere lo stesso numero di parentesi chiuse alla fine mantiene l'equilibrio globale.
- Se `balance(n) ≠ 0`, nessun `k` può mai equilibrare la differenza.

In questo modo, l'algoritmo determina correttamente il minimo k per cui la sequenza è k-bilanciata, o conclude che non è possibile bilanciarla.s