Ordinamento a palett (`paletta`)

Romeo ha di recente assistito ad un'avvincente performance di un cuoco acrobatico, che gestiva una imponente grigliata di braciole ribaltandole a gruppi di tre per mezzo di una apposita paletta. Questo evento gli ha ispirato l'idea del *paletta-sort*, una nuova interessante procedura di ordinamento.

Dato un vettore `V` contenente gli interi da 0 a $N-1$ (indicizzato da 0 a $N-1$), l'unica operazione ammessa nel paletta-sort è l'operazione *ribalta*. Questa operazione sostituisce tre elementi A, B, C consecutivi di `V` con i corrispondenti ribaltati C, B, A. Aiuta Romeo a capire se è possibile ordinare il vettore `V`, e in caso affermativo quante e quali operazioni *ribalta* sono sufficienti!

Implementazione

Dovrai sottoporre esattamente un file con estensione `.cpp`.

> ☞ Tra gli allegati a questo task troverai un template (`paletta.cpp`) con un esempio di implementazione.

Dovrai implementare la seguente funzione:

Funzione `paletta_sort`

C++
```
long long paletta_sort(int N, int V[]);
```

- L'intero N rappresenta il numero di elementi da ordinare.
- Il vettore `V`, indicizzato da 0 a $N-1$, contiene la sequenza da ordinare.
- La funzione dovrà restituire il numero ribaltamenti effettuati per ordinare `V`, oppure -1 se non c'è modo di ordinare il vettore.

Il grader chiamerà la funzione `paletta_sort` e ne stamperà il valore restituito sul file di output.

Grader di prova

Nella directory relativa a questo problema è presente una versione semplificata del grader usato durante la correzione, che potete usare per testare le vostre soluzioni in locale. Il grader di esempio legge i dati di input dal file `input.txt`, chiama le funzioni che dovete implementare e scrive il file `output.txt`, secondo il seguente formato.

Il file `input.txt` è composto da due righe, contenenti:

- Riga 1: l'unico intero N.
- Riga 2: i valori `V[`i`]` per $i = 0, \ldots, N-1$.

Il file `output.txt` è composto da una riga, contenente:

- Riga 1: il valore R restituito dalla funzione `paletta_sort`.

Assunzioni

- $1 \leq N \leq 1\,500\,000$.
- $0 \leq$ `V[`i`]` $\leq N-1$ per ogni $i = 0, \ldots, N-1$.

Assegnazione del punteggio

Il tuo programma verrà testato su diversi test case raggruppati in subtask. Per ogni test case riceverai punteggio:

- 1: se riporti correttamente il numero minimo di ribaltamenti;
- 0.2: se il vettore `V` è ordinabile e restituisci un numero maggiore o uguale a 0 (anche se non corrisponde al numero minimo di ribaltamenti), cioè riesci a distinguere quando il vettore è ordinabile;
- 0: altrimenti.

Per ogni subtask riceverai un punteggio pari al valore del subtask moltiplicato per il peggior punteggio ottenuto su uno dei suoi test case.

- **Subtask 1** [**5 punti**]: Casi d'esempio.
- **Subtask 2** [**19 punti**]: $N \leq 100$.
- **Subtask 3** [**24 punti**]: $N \leq 5000$.
- **Subtask 4** [**21 punti**]: $N \leq 100\,000$ e $R \leq 100$ (oppure `V` non è ordinabile).
- **Subtask 5** [**25 punti**]: $N \leq 100\,000$.
- **Subtask 6** [**6 punti**]: Nessuna limitazione specifica.

Esempi di input/output

5 2 0 4 3 1	-1
6 2 3 0 5 4 1	3

Spiegazione

Nel **primo caso di esempio**, non è possibile ordinare il vettore dato.

Nel **secondo caso di esempio**, la soluzione proposta produce la seguente sequenza di ribaltamenti:

2	3	0	5	4	1

2	3	0	1	4	5

0	3	2	1	4	5

0	1	2	3	4	5