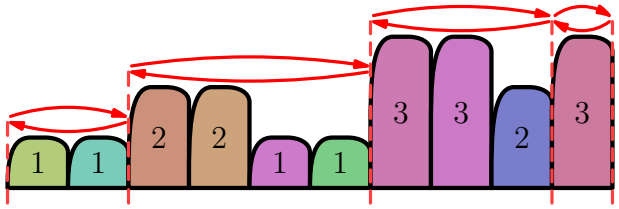
Vetrate colorate (`artemoderna`)

Per celebrare la nomina di Bergamo a capitale Italiana della Cultura, è appena arrivata un'installazione d'arte moderna ad abbellire l'entrata della stazione! L'opera è composta da N grandi vetrate colorate di diverse altezze, con il vetro opaco da un lato e lucido dall'altro. Per accogliere i partecipanti delle OII[1] nel migliore dei modi, l'amministrazione comunale vuole che tutte abbiano il lato lucido verso l'entrata, così che i nuovi arrivati possano subito ammirarle in tutta la loro bellezza.

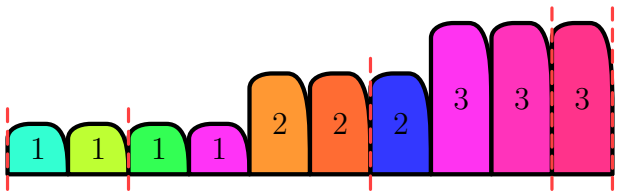
Figura 1: Vetrate colorate molto artistiche.

Al momento le vetrate sono allineate alla base, saldate l'una accanto all'altra, e tutte mostrano il lato opaco verso l'entrata. Giorgio, che è stato mandato dal comune a ruotare l'intera installazione, non è un grande estimatore dell'opera e ne critica molto l'asimmetria. Guardando le vetrate, si chiede se sia possibile ordinarle per altezza crescente mentre le gira dal lato giusto, a costo di rompere un po' l'opera.

Invece di ruotare tutto in un colpo solo, vuole spaccare l'installazione lungo alcune giunture tra vetri consecutivi e ruotare singolarmente tutti gli intervalli così formati, senza scambiarli di ordine e senza lasciare fuori nessuna vetrata. Vorrebbe che alla fine di queste operazioni le vetrate siano ordinate per altezza crescente da sinistra a destra. Per esempio se l'installazione è questa:



Spaccando lungo le linee tratteggiate e ruotando ogni intervallo si ottiene:



Aiuta Giorgio a capire se può ordinare l'installazione e come, riportando un po' d'ordine nell'arte moderna!

[1] *Oii Italiane di Informatica* (OII[1]).

Implementazione

Dovrai sottoporre un unico file, con estensione `.cpp`.

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

Dovrai implementare la seguente funzione:

C++
```
bool ordina(int N, vector<int> V, vector<int> &L);
```

- L'intero N è il numero di vetrate.
- L'array V, indicizzato da 0 a $N-1$, contiene le altezze, in ordine, delle vetrate.
- La funzione dovrà restituire `true` se V è ordinabile nel modo descritto dal testo, `false` altrimenti.
- Se il valore restituito è `true`, il vettore L, inizialmente vuoto, dovrà contenere le lunghezze degli intervalli in cui viene diviso l'array, in ordine da sinistra a destra.
  Attenzione! I valori contenuti in L devono essere tutti maggiori di zero e avere somma N.

Il grader chiamerà la funzione `ordina` e stamperà in output le informazioni restituite.

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 da `stdin`, chiama la funzione `ordina` e scrive su `stdout`, secondo il seguente formato.

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

- Riga 1: l'intero N, che indica la dimensione dell'array V;
- Riga 2: gli N interi $V[0], \ldots, V[N-1]$, separati da uno spazio.

Il file di output è composto da una prima riga, contenente `YES` o `NO` a seconda che `ordina` restituisca (rispettivamente) `true` o `false`. In caso di risposta positiva, `YES` viene seguito da due righe:

- Riga 1: l'intero M, che indica la dimensione di L;
- Riga 2: gli M interi $L[0], \ldots, L[M-1]$, separati da uno spazio.

Assunzioni

- $2 \leq N \leq 300\,000$.
- $0 \leq V[i] \leq 1\,000\,000\,000$ per $i = 0, \ldots, N-1$.

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 case che lo compongono.

- **Subtask 1 [0 punti]**: Casi d'esempio.
- **Subtask 2 [6 punti]**: I valori in V sono in ordine *crescente* oppure *decrescente*.
- **Subtask 3 [17 punti]**: In V compaiono solo i numeri 0 e 1.
- **Subtask 4 [12 punti]**: $2 \leq N \leq 200$.
- **Subtask 5 [14 punti]**: $2 \leq N \leq 5000$.
- **Subtask 6 [30 punti]**: Tutti i numeri in V sono distinti.
- **Subtask 7 [21 punti]**: Nessuna limitazione aggiuntiva.

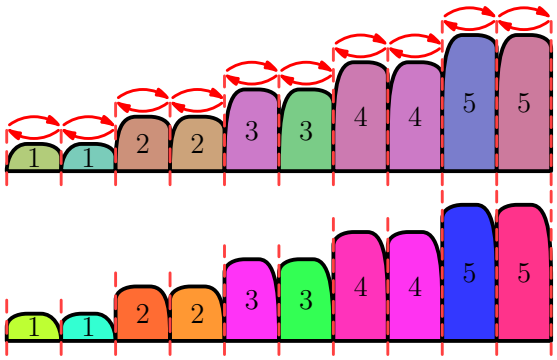
Esempi di input/output

stdin	stdout
10 1 1 2 2 1 1 3 3 2 3	YES 4 2 4 3 1
10 1 1 2 2 3 3 4 4 5 5	YES 10 1 1 1 1 1 1 1 1 1 1
5 50 40 30 20 10	YES 1 5
14 0 1 0 2 0 3 0 4 0 5 0 6 0 7	NO

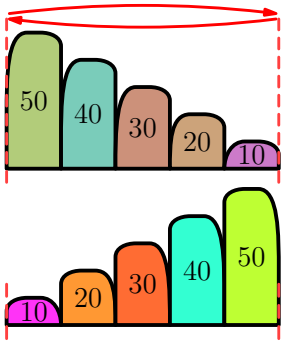
Spiegazione

Nel **primo caso d'esempio**, che corrisponde a quello descritto nel testo, una possibile soluzione consiste nel ribaltare gli intervalli di vetrate $\{0, 1\}, \{2, 3, 4, 5\}, \{6, 7, 8\}, \{9\}$ (nota che questa suddivisione copre tutta l'installazione). Le lunghezze di questi intervalli sono $2, 4, 3, 1$.

Nel **secondo caso d'esempio**, l'installazione è già ordinata. Suddividendola in intervalli da una sola vetrata (tenendo separato ogni singolo elemento) produce una soluzione valida, come visibile di seguito.



Nel **terzo caso d'esempio**, è sufficiente ribaltare l'intera installazione in un'unica mossa.



Nel **quarto caso d'esempio**, è possibile dimostrare che non esiste una suddivisione che permetta di ordinare l'installazione nel modo descritto dal testo.