Shop Tour (tour)

A Linealandia ci sono N negozi di biscotti in fila, numerati da 0 a $N-1$. Baq vuole fare un *tour dei negozi* visitando tutti i negozi esattamente una volta. Un tour dei negozi è determinato da N interi **distinti** $P_0, \ldots, P_{N-1}$ tra 0 e $N-1$.

Per un dato tour dei negozi, Baq inizierà dal negozio P_0. Per ogni $i = 0, \ldots, N-1$, Baq si sposterà dal negozio P_i al negozio P_{i+1} (qui diciamo $P_N = P_0$) comprando un biscotto da ciascuno dei negozi tra P_i e P_{i+1}, inclusi. Formalmente, se $L_i = \min(P_i, P_{i+1})$ e $R_i = \max(P_i, P_{i+1})$, allora nel passo i-esimo Baq comprerà un biscotto da ciascuno dei negozi $L_i, L_i + 1, \ldots, R_i$.

Baq ha acquistato A_i biscotti nel negozio i-esimo ma non ricorda il suo tour dei negozi. Il tuo compito è determinare se le informazioni nell'array A sono coerenti con un tour dei negozi valido, e se lo sono, costruire un tale tour valido. Inoltre, per ottenere un punteggio pieno (vedi la sezione di punteggio per i dettagli), il tour che costruisci deve essere il tour *lessicograficamente minimo* tra quelli coerenti con le informazioni nell'array A.

Diciamo che un tour $P_0, \ldots, P_{N-1}$ è *lessicograficamente minore* di un tour $Q_0, \ldots, Q_{N-1}$ se esiste un $0 \le k \le N-1$ tale che:

- $P_i = Q_i$ per tutti $0 \le i < k$.
- $P_k < Q_k$.

Un tour Q è *lessicograficamente minimo* tra quelli coerenti con le informazioni nell'array A se non esiste un tour P con lo stesso array A di biscotti acquistati in ogni negozio che sia lessicograficamente minore di Q.

Implementazione

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

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

Dovrai implementare la seguente funzione:

C++
```
variant<bool, vector<int>> find_tour(int N, vector<int> A);
```

- L'intero N rappresenta il numero di negozi.
- L'array A, indicizzato da 0 a $N-1$, contenente i valori A_0, A_1, …, A_{N-1}, dove A_i è il numero di biscotti acquistati nel negozio i-esimo.
- La funzione deve restituire un booleano o un array di interi.
  - Se non esiste un tour dei negozi valido che corrisponde all'array A, la funzione deve restituire `false`.
  - Se esiste un tour dei negozi valido, hai più opzioni:
    - Per ottenere il punteggio pieno, la procedura deve restituire un array di N interi $P_0, \ldots, P_{N-1}$ rappresentanti il **tour dei negozi lessicograficamente minimo** risultante dall'array A.

    * Per ottenere un punteggio parziale, la procedura può restituire un array di N interi $P_0, \ldots, P_{N-1}$ rappresentanti un qualunque tour non lessicograficamente minimo risultante dall'array A.
    * Per ottenere un punteggio parziale ancora più basso, la procedura può restituire `true` o un qualunque array di interi non rappresentante un tour valido risultante dall'array A.

Il grader chiamerà la funzione `tour` e ne stamperà il valore di ritorno sul file di output:

- Se il valore di ritorno è `false`, stamperà una riga contenente la stringa `NO`.
- Se il valore di ritorno è `true` o un array di interi di lunghezza diversa da N, stamperà una riga contenente la stringa `YES`.
- Se il valore di ritorno è un array P di N interi, stamperà una riga contenente la stringa `YES`, seguita da una riga contenente gli N interi $P_0, \ldots, P_{N-1}$ separati da spazio.

Grader di prova

La directory relativa a questo problema contiene 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 `stdin`, chiama la funzione che dovete implementare e scrive il risultato sul file `stdout`.

L'input è composto da due righe, contenenti:

- Riga 1: l'intero N.
- Riga 2: gli interi A_i, separati da spazio.

L'output è composto da una o due righe, contenenti il valore di ritorno della funzione `tour`.

Assunzioni

- $2 \leq N \leq 10^6$.
- $0 \leq A_i \leq 10^6$.

Assegnazione del punteggio

Il tuo programma verrà testato su diversi test case raggruppati in subtask. Il punteggio associato ad un subtask è il minimo dei punteggi ottenuti nei singoli test case.

- **Subtask 1 [0 punti]**: Casi d'esempio.
- **Subtask 2 [8 punti]**: $N \leq 8$.
- **Subtask 3 [32 punti]**: $N \leq 2 \times 10^3$.
- **Subtask 4 [16 punti]**: $A_i \leq 4$ per ogni $i = 0, \ldots, N-1$.
- **Subtask 5 [20 punti]**: Esiste un $0 \leq j \leq N-1$ tale che $A_i \leq A_{i+1}$ per ogni $0 \leq i < j$ e $A_i \geq A_{i+1}$ per ogni $j \leq i \leq N-2$.
- **Subtask 6 [24 punti]**: Nessuna limitazione aggiuntiva.

Per ogni test case in cui esiste un tour valido, la tua soluzione:

- ottiene il punteggio pieno se restituisce il tour dei negozi lessicograficamente minimo.
- ottiene il 75% dei punti se restituisce un tour valido che non è quello lessicograficamente minimo.
- ottiene il 50% dei punti se restituisce `true` o un array che non rappresenta un tour valido.

- ottiene 0 punti negli altri casi.

Per ogni test case in cui non esiste un tour valido, la tua soluzione:

- ottiene il punteggio pieno se restituisce `false`.
- ottiene 0 punti negli altri casi.

Esempi di input/output

stdin	stdout
4 2 4 4 2	YES 0 2 1 3
3 2 2 2	NO

Spiegazione

Nel **primo caso d'esempio**, il tour $P = [0, 2, 1, 3]$ genera l'array $A = [1, 2, 3, 4]$ come segue:

- Inizialmente, il numero di biscotti comprati da ogni negozio è $[0, 0, 0, 0]$.
- Baq si sposta dal negozio $P_0 = 0$ al negozio $P_1 = 2$, quindi l'array dopo questo spostamento è $[1, 1, 1, 0]$.
- Baq si sposta dal negozio $P_1 = 2$ al negozio $P_2 = 1$, quindi l'array dopo questo spostamento è $[1, 2, 2, 0]$.
- Baq si sposta dal negozio $P_2 = 1$ al negozio $P_3 = 3$, quindi l'array dopo questo spostamento è $[1, 3, 3, 1]$.
- Infine, Baq si sposta dal negozio $P_3 = 3$ al negozio $P_0 = 0$, quindi l'array finale è $[2, 4, 4, 2]$.

Può essere dimostrato che il tour $[0, 2, 1, 3]$ è lessicograficamente minimo.

Nel **secondo caso d'esempio**, può essere che non esista un tour valido che genera l'array $A = [2, 2, 2]$.