Shop Tour (tour)

En Lineland hay N tiendas de galletas seguidas, numeradas de 0 a $N-1$. Baq quiere hacer un tour por las tiendas. Un tour de tiendas está determinado por N enteros **distintos** P_0, P_{N-1} entre 0 y $N-1$.

Para un tour de tiendas dado, Baq empezará en la tienda P_0. Para cada $i = 0, \ldots, N-1$, Baq pasará de la tienda P_i a la tienda P_{i+1} (decimos $P_N = P_0$) comprando una galleta de cada una de las tiendas entre P_i y P_{i+1}, ambas inclusive. Formalmente, si $L_i = \text{mín}(P_i, P_{i+1})$ y $R_i = \text{máx}(P_i, P_{i+1})$, entonces en el paso i-ésimo Baq comprará una galleta de cada una de las tiendas $L_i, L_i + 1, \ldots, R_i$.

Baq tiene ahora los números $A_0, \ldots, A_{N-1}$, donde A_i denota el número total de galletas compradas en la i-ésima tienda, pero no recuerda el tour de tiendas. Su tarea consiste en determinar si la información de el array A es coherente con un tour válido por la tienda y, si lo es, construir dicho tour válido. Además, para obtener una puntuación completa (ver la sección de puntuación para más detalles) el tour que construya debe ser el *lexicográficamente más pequeño*.

Decimos que un tour $P_0, \ldots, P_{N-1}$ es *lexicográficamente más pequeño* que otro tour $Q_0, \ldots, Q_{N-1}$. si existe un $0 \leq k \leq N-1$ tal que:

- $P_i = Q_i$ para todo $0 \leq i < k$.
- $P_k < Q_k$.

Un tour Q es el lexicográficamente más pequeño entre los consistentes con la información de el array A si no existe otro tour P con el mismo array A de galletas compradas en cada tienda que sea lexicográficamente menor que Q.

Implementación

Deberá presentar un único fichero fuente `.cpp`.

> ☞ Entre los archivos adjuntos de esta tarea encontrará una plantilla `tour.cpp` con una implementación de ejemplo.

Tiene que implementar la siguiente función:

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

- El entero N representa el número de tiendas.
- El array A, indexada de 0 a $N-1$, contiene los valores A_0, A_1, …, A_{N-1}, donde A_i es el número de galletas compradas en la tienda i-ésima.
- La función debe devolver un booleano o un array de enteros.
  - Si no existe ningún tour de tiendas válido que corresponda al array A, la función debe devolver `false`.
  - Si existe un tour de tienda válidos, tiene varias opciones:
    - Para obtener la puntuación completa, el procedimiento debe devolver un array de N enteros $P_0, \ldots, P_{N-1}$ que representan el tour de tiendas **lexicográficamente más pequeña** resultante en el array A.
    - Para obtener una puntuación parcial, el procedimiento debe devolver un array de N enteros $P_0, \ldots, P_{N-1}$ que representan cualquier tour de tiendas no-lexicográficamente-más pequeño que dé como resultado el array A.

- Para obtener una puntuación parcial menor, el procedimiento debe devolver `true` o cualquier matarrayriz de enteros que no describa un tour de tiendas válido resultando el array A.

El grader llamará a la función `tour` e imprimirá lo siguiente en el archivo de salida:

- Si el valor de retorno es `false`, imprimirá una sola línea con la cadena `NO`. Si el valor de retorno es `true` o un array de enteros de longitud no igual a N, se imprimirá una sola línea con la cadena `YES`.
- Si el valor de retorno es un array P de N enteros, se imprimirá una sola línea con la cadena `YES`, seguida de una línea con los N enteros $P_0, \ldots, P_{N-1}$ separados por espacios.

Sample Grader

El directorio de la tarea contiene una versión simplificada del grader del jurado, que puede utilizar para probar su solución localmente. El grader simplificado lee los datos de entrada de `stdin`, llama a las funciones que debe implementar, y finalmente escribe la salida a `stdout`.

La entrada se compone de dos líneas, que contienen:

- Línea 1: el entero N.
- Línea 2: los enteros A_i, separados por espacios.

La salida se compone de una o dos líneas, que contienen los valores devueltos por la función `tour`.

Restricciones

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

Puntuación

Tu programa será probado en un conjunto de casos de prueba agrupados por subtarea. La puntuación asociada a una subtarea será el mínimo de las puntuaciones obtenidas en cada uno de los casos de prueba.

- **Subtask 1 [0 puntos]**: Ejemplos de casos de prueba. número de subtarea (basado en 1).
- **Subtask 2 [8 puntos]**: $N \leq 8$.
- **Subtask 3 [32 puntos]**: $N \leq 2 \times 10^3$.
- **Subtask 4 [16 puntos]**: $A_i \leq 4$ para todo $i = 0, \ldots, N-1$.
- **Subtask 5 [20 puntos]**: Existe un $0 \leq j \leq N-1$ tal que $A_i \leq A_{i+1}$ para todo $0 \leq i < j$ y $A_i \geq A_{i+1}$ para todo $j \leq i \leq N-2$.
- **Subtask 6 [24 puntos]**: Sin restricciones adicionales.

Para cada caso de prueba en el que exista un tour de tiendas válido, su solución:

- obtiene todos los puntos si devuelve el tour lexicográficamente más pequeño de tiendas válida.
- obtiene 75 % de los puntos si devuelve un tour de tienda válido que no es el lexicográficamente más pequeño.
- obtiene 50 % de los puntos si devuelve `true` o un array que no no describe un tour de tiendas válido.
- obtiene 0 puntos en caso contrario.

Para cada caso de prueba en el que no exista un tour válido por la tienda, su solución:

- obtiene puntos completos si devuelve `false`.
- obtiene 0 puntos en caso contrario.

Ejemplos de entrada/salida

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

Explicación

En el **caso de la primera muestra**, el tour $P = [0, 2, 1, 3]$ genera el array $A = [2, 4, 4, 2]$ de la siguiente manera:

- Inicialmente, el número de galletas compradas en cada tienda es $[0, 0, 0, 0]$.
- Baq se mueve de la tienda $P_0 = 0$ a la tienda $P_1 = 2$, por lo que el array después de este movimiento es $[1, 1, 1, 0]$.
- Baq se mueve de la tienda $P_1 = 2$ a la tienda $P_2 = 1$, por lo que el array después de este movimiento es $[1, 2, 2, 0]$.
- Baq se mueve de la tienda $P_2 = 1$ a la tienda $P_3 = 3$, por lo que el array después de este movimiento es $[1, 3, 3, 1]$.
- Por último, Baq se mueve de la tienda $P_3 = 3$ a la tienda $P_0 = 0$, por lo que el array final es $[2, 4, 4, 2]$.

Se puede demostrar que $[0, 2, 1, 3]$ es el tour lexicográficamente más pequeño.

En el caso **segundo caso de muestra**, se puede demostrar que no existe un tour válido que resulte en el array $A = [2, 2, 2]$.