Shop Tour (tour)

Em Lineland há N lojas de bolachas em fila, numeradas de 0 a $N-1$. O Baq quer fazer uma *percurso de compras* pelas lojas. Um percurso é determinado por N inteiros **distintos** $P_0, \ldots, P_{N-1}$ entre 0 e $N-1$.

Para um determinado percurso, Baq começa na loja P_0. Para cada $i = 0, \ldots, N-1$, Baq deslocar-se-á da loja P_i para a loja P_{i+1} (aqui dizemos que $P_N = P_0$) comprando uma bolacha de cada uma das lojas entre P_i e P_{i+1}, inclusive. Formalmente, se $L_i = \min(P_i, P_{i+1})$ e $R_i = \max(P_i, P_{i+1})$, então no i-ésimo passo Baq comprará uma bolacha em cada uma das lojas $L_i, L_i + 1, \ldots, R_i$.

Baq tem agora os números $A_0, \ldots, A_{N-1}$, em que A_i representa o número total de bolachas compradas na i-ésima loja, mas não se lembra do percurso. A tua tarefa é determinar se a informação na matriz A é consistente com um percurso e, se for, construir esse percurso válido. Além disso, para obter uma pontuação máxima (ver a secção de pontuação para mais detalhes) o percurso que construires tem de ser o *lexicograficamente mais pequeno* dos percursos.

Dizemos que um percurso $P_0, \ldots, P_{N-1}$ é *lexicograficamente mais pequeno* do que um percurso diferente $Q_0, \ldots, Q_{N-1}$ se existir um $0 \leq k \leq N-1$ tal que:

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

Um percurso Q é o mais pequeno lexicograficamente entre os que são consistentes com a informação do conjunto A se não existir um percurso diferente P com o mesmo conjunto A de bolachas compradas em cada loja que seja lexicograficamente mais pequeno que Q.

Implementação

Deves submeter um único ficheiro de código `.cpp`.

> ☞ Entre os ficheiros do problema encontrarás um template `tour.cpp` com um exemplo de implementação.

Tens de implementar a seguinte função:

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

- O inteiro N representa o número de lojas.
- O array A, indexado de 0 até $N-1$, contém os valores A_0, A_1, …, A_{N-1}, onde A_i é o número de bolachas compradas na i-ésima loja.
- A função deve devolver um valor booleano ou um array de inteiros.
  - Se não existir nenhum percurso válido que corresponda ao array A, a função deve devolver `false`.
  - Se um percurso válido existir, tens múltiplas opções:
    - Para receber pontuação máxima, a função deve devolver um array de N inteiros $P_0, \ldots, P_{N-1}$ representando o percurso **lexicograficamente mais pequeno** que resulta bo array A.

    * Para receber uma pontuação parcial, a função deve devolver um array de N inteiros $P_0, \ldots, P_{N-1}$ representando um qualquer percurso que não seja o lexicograficamente menor e resulte no array A.
    * Para receber uma pontuação parcial menor, a função deve devolver `true` ou um qualquer array de inteiros que não descreva um percurso que resulta no array A.

O avaliador irá chamar a função `find_tour` e irá escrever o seguinte no ficheiro de output:

- Se o valor devolvido for `false`, irá escrever uma única linha com a string `NO`.
- Se o valor devolvido for `true` ou um array de inteiros com tamanho diferente de N, irá escrever uma única linha com a string `YES`.
- Se o valor devolvido for um array P de N integers, irá escrever uma linha com a string `YES`, seguida de uma linha com N inteiros $P_0, \ldots, P_{N-1}$ separados por espaços.

Avaliador Padrão

O diretório do problema contém uma versão simplificada do avaliador oficial, que podes usar para testar o teu problema localmente. O avaliador exemplo lê os dados de input de `stdin`, chama as funções que deves implementar, e finalmente escreve o output para `stdout`.

O input é feito de duas linhas, contendo:

- Linha 1: o inteiro N.
- Linha 2: os inteiros A_i, separados por espaços.

O output é feito de uma ou duas linhas, contendo os valores devolvidos pela função `find_tour`.

Restrições

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

Pontuação

O teu programa será testado num conjunto de casos de teste agrupados por subtarefa. A pontuação de uma subtarefa será o mínimo das pontuações obtidas em cada um dos casos de teste.

- **Subtarefa 1 [0 pontos]**: Casos de exemplo.
- **Subtarefa 2 [8 pontos]**: $N \leq 8$.
- **Subtarefa 3 [32 pontos]**: $N \leq 2 \times 10^3$.
- **Subtarefa 4 [16 pontos]**: $A_i \leq 4$ para $i = 0, \ldots, N-1$.
- **Subtarefa 5 [20 pontos]**: Existe um $0 \leq j \leq N-1$ tal que $A_i \leq A_{i+1}$ para $0 \leq i < j$ e $A_i \geq A_{i+1}$ para $j \leq i \leq N-2$.
- **Subtarefa 6 [24 pontos]**: Nenhuma restrição adicional.

Para cada caso de teste onde exista um percurso válido, a tua solução:

- obtém pontuação máxima se devolver um percurso válido que seja o lexocograficamente menor.
- obtém 75% dos pontos se devolver um percurso válido que não seja o lexicograficamente menor.
- obtém 50% dos pontos se devolver `true` um array que não descreva um percurso válido.

- obtém 0 pontos caso contrário.

Para cada caso de teste em que não exista um percurso válido, a tua solução:

- obtém pontuação máxima se devolver `false`.
- obtém 0 pontos caso contrário.

Exemplos

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

Explicação

No **primeiro caso de exemplo**, o percurso $P = [0, 2, 1, 3]$ gera o array $A = [2, 4, 4, 2]$ tal como a seguir descrito:

- Inicialmente, o número de bolachas comprado em cada loja é $[0, 0, 0, 0]$.
- Baq move-se da loja $P_0 = 0$ para a loja $P_1 = 2$, e portanto o array depois deste movimento fica $[1, 1, 1, 0]$.
- Baq move-se da loja $P_1 = 2$ para a loja $P_2 = 1$, e portanto o array depois deste movimento fica $[1, 2, 2, 0]$.
- Baq move-se da loja $P_2 = 1$ para a loja $P_3 = 3$, e portanto o array depois deste movimento fica $[1, 3, 3, 1]$.
- Finalmente, Baq move-se da loja $P_3 = 3$ para a loja $P_0 = 0$, e portanto o array final é $[2, 4, 4, 2]$.

Pode ser mostrado que $[0, 2, 1, 3]$ o percurso que é lexicograficamente menor.

No **segundo caso de exemplo**, pode ser mostrado que não existe um percurso válido resultando no array $A = [2, 2, 2]$.