Shop Tour (tour)

À Lineland, il y a N boutiques de cookies, numérotés de 0 à $N-1$. Baq veut faire une *visite des boutiques*. Une visite est déterminée par N entiers distincts $P_0, \ldots, P_{N-1}$ entre 0 et $N-1$.

Pour une visite donnée, Baq va commencer à la boutique P_0. Pour chaque $i = 0, \ldots, N-1$, Baq va bouger de la boutique P_i à la boutique P_{i+1} (on considère que $P_N = P_0$) en achetant un cookie de chacune des boutiques entre P_i et P_{i+1}, inclus. Formellement, si $L_i = \min(P_i, P_{i+1})$ et $R_i = \max(P_i, P_{i+1}$, alors à la i-ème étape Baq va acheter un cookie à chacune des boutiques $L_i, L_i + 1, \ldots, R_i$.

Baq dispose maintenant des nombres $A_0, \ldots, A_{N-1}$, où A_i correspond au nombre total de cookies achetés à la i-ème boutique, mais ne se souvient plus de sa visite des boutiques. Vous devez déterminer si l'information dans le tableau A est cohérente avec une visite des boutiques valide, et si oui, construire une visite valide. De plus, afin d'obtenir le total des points (voir la section score pour plus de détails), la visite doit être celle qui est la *plus petite lexicographiquement*.

On dit qu'une visite $P_0, \ldots, P_{N-1}$ est (strictement) *plus petite lexicographiquement* qu'une autre visite $Q_0, \ldots, Q_{N-1}$ s'il existe un $0 \leq k \leq N-1$ tel que :

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

Une visite Q est est la plus petite lexicographiquement parmi celle cohérente avec l'information dans le tableau A s'il n'existe pas de visite différente P qui corresponde au même tableau A de cookies achetés dans chaque boutique et qui soit (strictement) plus petite lexicographiquement que Q.

Implémentation

Vous devrez soumettre un unique fichier source `.cpp`.

> ☞ Parmi les pièces jointes à ce problème, vous trouverez un squelette `tour.cpp` avec un exemple d'implémentation.

Vous devez implémenter la fonction suivante :

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

- L'entier N représente le nombre de boutiques.
- Le tableau A, indexé de 0 à $N-1$, contient les valeurs A_0, A_1, …, A_{N-1}, où A_i est le nombre de cookies achetés à la i-ème boutique.
- La fonction doit renvoyer un booléen ou un tableau d'entiers.
  - S'il n'y a pas de visite des boutiques valide qui corresponde au tableau A, la fonction doit renvoyer `false`.
  - Si une visite des boutiques valide existe, vous avez plusieurs options :
    - Pour recevoir le total des points, la fonction doit renvoyer un tableau de N entiers $P_0, \ldots, P_{N-1}$ représentant la visite la **plus petite lexicographiquement** correspondant au tableau A.
    - Pour recevoir un score partiel, la fonction doit renvoyer un tableau de N entiers $P_0, \ldots, P_{N-1}$ représentant n'importe quelle visite correspondant au tableau A.

— Pour recevoir un score partiel plus petit, la fonction doit renvoyer `true` ou n'importe quel tableau d'entiers qui ne décrit pas une visite de boutiques valide correspondant au tableau A.

L'évaluateur va appeler la fonction `tour` et écrire ce qui suit dans le fichier de sortie :

— Si la valeur de retour est `false`, il va écrire une seule ligne avec la chaîne de caractères `NO`.
— Si la valeur de retour est `true` ou un tableau d'entiers de longueur différente de N, il va écrire une unique ligne avec la chaîne de caractère `YES`.
— Si la valeur de retour est un tableau P de N entiers, il va écrire une ligne avec la chaîne de caractère `YES`, suivie d'une ligne avec les N entiers $P_0, \ldots, P_{N-1}$ séparés par des espaces.

Évaluateur

Le dossier du problème contient une version simplifiée de l'évaluateur du jury, que vous pouvez utiliser pour tester votre solution localement. L'évaluateur simplifié lit l'entrée depuis `stdin`, appelle les fonctions à implémenter, et enfin écrit la sortie dans `stdout`.

L'entrée est constituée de deux lignes, contenant :

— Ligne 1 : l'entier N.
— Ligne 2 : les entiers A_i, séparés par des espaces.

La sortie est constituée d'une ou deux lignes, contenant les valeurs renvoyés par la fonction `tour`.

Contraintes

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

Score

Votre programme sera testé sur un ensemble de tests groupés par sous-tâche. Le score associé à une sous-tâche sera le minimum des scores obtenus dans chacun des tests.

— **Sous-tâche 1 [0 points]**: Sample test cases.
— **Sous-tâche 2 [8 points]**: $N \leq 8$.
— **Sous-tâche 3 [32 points]**: $N \leq 2 \times 10^3$.
— **Sous-tâche 4 [16 points]**: $A_i \leq 4$ pour tout $i = 0, \ldots, N-1$.
— **Sous-tâche 5 [20 points]**: Il existe $0 \leq j \leq N-1$ tel que $A_i \leq A_{i+1}$ pour tout $0 \leq i < j$ et $A_i \geq A_{i+1}$ pour tout $j \leq i \leq N-2$.
— **Sous-tâche 6 [24 points]**: Pas de contraintes supplémentaires.

Pour chaque test pour lequel il existe une visite des boutiques valide, votre solution :

— reçoit tous les points si elle renvoie la visite valide la plus petite lexicographiquement.
— reçoit 75% des points si elle renvoie une autre visite valide.
— reçoit 50% des points si elle renvoie `true` ou un tableau qui ne décrit pas une visite valide.
— reçoit 0 points dans les autres cas.

Pour chaque test pour lequel il n'y a pas de visite valide, votre solution :

— reçoit tous les points si elle renvoie `false`.

— reçoit 0 points sinon.

Exemples

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

Explication

Dans le **premier exemple**, la visite $P = [0, 2, 1, 3]$ génère le tableau $A = [2, 4, 4, 2]$ comme expliqué ci-dessous :

— Initialement, le nombre de cookies acheté depuis chaque boutique est $[0, 0, 0, 0]$.
— Baq bouge de la boutique $P_0 = 0$ à la boutique $P_1 = 2$, de sorte que le tableau après cette étape est $[1, 1, 1, 0]$.
— Baq bouge de la boutique $P_1 = 2$ à la boutique $P_2 = 1$, de sorte que le tableau après cette étape est $[1, 2, 2, 0]$.
— Baq bouge de la boutique $P_2 = 1$ à la boutique $P_3 = 3$, de sorte que le tableau après cette étape est $[1, 3, 3, 1]$.
— Enfin, Baq bouge de la boutique $P_3 = 3$ à la boutique $P_0 = 0$, de sorte que le tableau final est $[2, 4, 4, 2]$.

On peut montrer que $[0, 2, 1, 3]$ est la visite la plus petite lexicographiquement.

Dans le **deuxième exemple**, on peut montrer qu'il n'existe pas de visite valide correspondant au tableau $A = [2, 2, 2]$.