Numbering (numbering)

Étant une forêt de N nœuds, une numérotation de celle-ci est une assignation d'un entier strictement positif à chacune de ses arêtes. Une numérotation est jolie si, pour tous les nœuds, ses arêtes ont les numéro $1, 2, \dots d$, dans un ordre quelconque (où d est le degré du nœud).

Étant donnés N entiers strictement positifs $A_0, \dots, A_{N-1}$, déterminez s'il existe une forêt de N nœuds qui respecte les conditions suivantes :

— quelque soit $0 \leq i \leq N-1$, le degré du nœud i est A_i ;
— la forêt admet une numérotation jolie

De plus, s'il existe une telle forêt, alors construisez en un exemple.

Implémentation

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

> ☞ Parmi les fichiers joints du problème, vous trouverez un squelette de code `numbering.cpp` avec un exemple d'implémentation.

Vous devez implémenter la fonction suivante :

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

— L'entier N représente le nombre de nœuds.
— Le tableau A, indexé de 0 à $N-1$, contient les valeurs A_0, A_1, …, A_{N-1}, où A_i est le degré du i-ème nœuds.
— La fonction doit renvoyer soit un booléen, soit un table de paires d'entiers.
  — Si aucune forêt (satisfaisant les conditions de l'énoncé) n'existe, la fonction doit retourner `false`.
  — Si une forêt valide existe, vous avez deux options :
    — Pour recevoir le total des points, la fonction doit retourner un tableau de paires d'entiers, qui représente une forêt valide.
    — Pour recevoir un score partiel, la fonction doit retourner `true` ou n'importe quel tableau d'entier qui ne représente pas une forêt valide.

L'évaluateur va appeler la fonction `find_numbering` et va écrire sa valeur de retour dans le fichier de sortie :

— Si la valeur de retour est `false`, il va écrire une seule ligne contenant la chaîne de caractères `NO`.
— Si la valeur de retour est `true`, il va écrire une seule ligne contenant la chaîne de caractères `YES`.
— Si la valeur de retour est un tableau de paires d'entier de taille M, il va écrire une ligne avec la chaîne de caractères `YES`, suivi d'une ligne contenant M, suivi de M lignes avec les paires du tableau.

Évaluateur

Le dossier du problème contient une version simplifiée de l'évaluateur du jury que vous pouvez utiliser pour tester votre solution localement. Cet évaluateur simplifié lit les données en entrée du fichier `stdin`, appelle les fonction que vous devez implémenter et enfin écrit la sortie dans `stdout`.

L'entrée est constituée de 2 lignes, contenant :

— Ligne 1 : l'entier N.

— Ligne 2 : $A_0, A_1, \ldots, A_{N-1}$.

La sortie est constituée de plusieurs lignes, contenant les valeurs retournées par la fonction `find_numbering`.

Contraintes

— $2 \leq N \leq 10^5$.

— $0 \leq A_i \leq N - 1$.

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]**: Exemples de l'énoncé.

— **Sous-tâche 2 [16 points]**: $A_i \leq 2$.

— **Sous-tâche 3 [12 points]**: $A_i \leq 3$.

— **Sous-tâche 4 [16 points]**: Soit $\texttt{count}(i)$ le nombre d'occurrences de i dans A. Il est garanti que $\texttt{count}(i) \geq \texttt{count}(i+1) + \texttt{count}(i+2) + \ldots$ pour tout $1 \leq i \leq N - 1$.

— **Sous-tâche 5 [10 points]**: $N \leq 12$.

— **Sous-tâche 6 [24 points]**: $N \leq 500$.

— **Sous-tâche 7 [22 points]**: Pas de contraintes supplémentaires.

Pour chaque test pour lequel une forêt valide existe, votre solution :

— reçoit tous les points si elle renvoie une forêt valide.

— reçoit 50% des points si elle renvoie `true` ou un tableau qui ne représente pas une forêt valide

— reçoit 0 points dans les autres cas.

Pour chaque test pour lequel une forêt valide n'existe pas, votre solution :

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

— reçoit 0 points sinon.

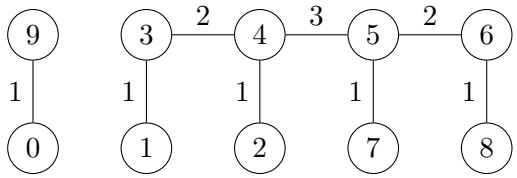
Exemples

stdin	stdout
4 1 1 2 1	NO
10 1 1 1 2 3 3 2 1 1 1	YES 8 0 9 1 3 2 4 3 4 4 5 5 6 5 7 6 8

Explication

Dans le **premier exemple**, une forêt de 4 nœuds doit être construite : 3 nœuds de degré 1 et 1 de degré 3. On peut montrer qu'une forêt valide n'existe pas. Supposons qu'une telle forêt existe, alors il doit exister une arête à laquelle on a attribué le nombre 2 et qui relie un nœud de degré 2 à un autre nœud de degré au moins 2. Ce premier nœud n'existe pas ici, puisqu'on n'a que des nœuds de degré 1

Dans le **deuxième exemple**, une forêt de 10 nœuds doit être construite : 6 nœuds de degré 1, 2 nœuds de degré 2 et 2 nœuds de degré 3. Une forêt valide avec ces degrés existe et est dessiné ci-dessous.



Notez que les nœuds 4 et 5 ont bien trois arêtes numérotées $1, 2$ et 3.
De plus, les nœuds 3 et 6 ont deux arêtes numérotées 1 et 2.
Enfin, les nœuds $0, 1, 2, 7, 8$ et 9 ont une arête numérotée 1.