

Numbering (numbering)

Gegeben ein Wald mit N Knoten, eine Nummerierung des Waldes ist eine Zuordnung von positiven ganzen Zahlen zu jeder Kante des Waldes. Eine Nummerierung ist wunderschön, wenn für jeden Knoten seine Kanten die Zahlen $1, 2, \dots, d$ in irgendeiner Reihenfolge haben (wobei d der Grad des Knotens ist).


Gegeben sind N positive ganze Zahlen A_0, \dots, A_{N-1} . Bestimmen Sie, ob es einen Wald auf N Knoten gibt, so dass:

- für jedes $0 \leq i \leq N - 1$ gilt, der Grad des Knotens i ist A_i ;
- es lässt mindestens eine schöne Nummerierung zu.

Zusätzlich, wenn es einen solchen Wald gibt, konstruiere ein Beispiel.

Implementierung

Du musst eine einzige `.cpp`-Quelltextdatei einreichen.

 In den Anhängen zu dieser Aufgabe finden Sie eine Vorlage `numbering.cpp` mit einer Beispielimplementierung.

Sie müssen die folgende Funktion implementieren:

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

- Die ganze Zahl N steht für die Anzahl der Knoten.
- Das Array A , indiziert von 0 bis $N - 1$, enthält die Werte A_0, A_1, \dots, A_{N-1} , wobei A_i der Grad des i -ten Knotens ist.
- Die Funktion sollte entweder einen booleschen Wert oder ein Array von Paaren von ganzen Zahlen zurückgeben.
 - Wenn kein gültiger (die Bedingungen der Aufgabenstellung erfüllender) Wald existiert, sollte die Funktion `false` zurückgeben.
 - Wenn ein gültiger Wald existiert, haben Sie zwei Möglichkeiten:
 - * Um die volle Punktzahl zu erhalten, sollte die Prozedur ein Array von Paaren von Ganzzahlen zurückgeben, die die Kanten eines gültigen Waldes darstellen.
 - * Um eine Teilbewertung zu erhalten, sollte die Prozedur zurückgeben `true` oder ein Array mit ganzen Zahlen, die keinen gültigen Wald beschreiben.

Der Grader ruft die Funktion `find_numbering` auf und gibt folgendes aus in die Ausgabedatei:

- Wenn der Rückgabewert `false` ist, wird eine einzelne Zeile mit der Zeichenfolge `NO` gedruckt.
- Wenn der Rückgabewert `true` ist, wird eine einzelne Zeile mit der Zeichenfolge `YES`.
- Wenn der Rückgabewert ein Array von Paaren von ganzen Zahlen der Länge M ist, wird eine Zeile mit der Zeichenfolge `YES` gedruckt, gefolgt von einer Zeile mit M , gefolgt von M Zeilen mit den Paaren des Arrays.

Beispielgrader

Das Verzeichnis der Aufgabe enthält eine vereinfachte Version des Jury-Graders, die Sie verwenden können, um Ihre Lösung lokal zu testen. Der vereinfachte Grader liest die Eingabedaten aus `stdin`, ruft die Funktionen auf, die Sie implementieren müssen, und schreibt schliesslich die Ausgabe in `stdout`.

Die Eingabe besteht aus 2 Zeilen, die Folgendes enthalten:

- Zeile 1: die ganze Zahl N .
- Zeile 2: A_0, A_1, \dots, A_{N-1} .

Die Ausgabe besteht aus mehreren Zeilen, die die Werte enthalten, die von der Funktion `find_numbering`.

Einschränkungen

- $2 \leq N \leq 10^5$.
- $0 \leq A_i \leq N - 1$.

Punktevergabe

Dein Programm wird mit einer Reihe von Testfällen getestet, die nach Teilaufgaben gruppiert sind. Die Punktzahl, die einer Teilaufgabe zugeordnet ist, ist das Minimum der Punktzahlen, die der einzelnen Testfälle.

- **Teilaufgabe 1 [0 Punkte]**: Beispiel-Testfälle.
- **Teilaufgabe 2 [16 Punkte]**: $A_i \leq 2$.
- **Teilaufgabe 3 [12 Punkte]**: $A_i \leq 3$.
- **Teilaufgabe 4 [16 Punkte]**: Sei $\text{count}(i)$ die Anzahl der Vorkommen von i in A . Du hast die Garantie, dass $\text{count}(i) \geq \text{count}(i+1) + \text{count}(i+2) + \dots$ für alle $1 \leq i \leq N-1$.
- **Teilaufgabe 5 [10 Punkte]**: $N \leq 12$.
- **Teilaufgabe 6 [24 Punkte]**: $N \leq 500$.
- **Teilaufgabe 7 [22 Punkte]**: Keine zusätzlichen Beschränkungen.

Für jeden Testfall, in dem ein gültiger Wald existiert, erhält deine Lösung:

- hält die volle Punktzahl, wenn es einen gültigen Forest liefert.
- 50% der Punkte, wenn es `true` oder ein Array liefert, das nicht einen gültigen Wald beschreibt.
- sonst 0 Punkte.

Für jeden Testfall, in dem kein gültiger Wald existiert, erhält deine Lösung:

- die volle Punktzahl, wenn es `false` zurückgibt.
- sonst 0 Punkte.

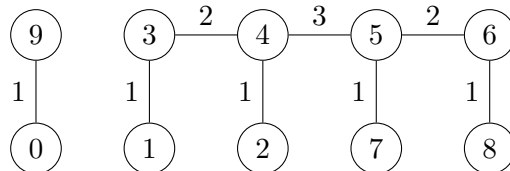
Beispiele

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

Erklärung

Im **ersten Testfall** wollen wir einen gültigen Wald mit 4 Knoten konstruieren: 3 mit Grad 1 und 1 mit Grad 2. Wir können zeigen, dass dies nicht möglich ist. Angenommen, ein solcher Wald existiert, dann müsste es eine Kante mit der Nummer 2 vom Knoten mit dem Grad 2 geben. Diese Kante verbindet sich mit einem anderen Knoten, der mindestens den Grad 2 haben sollte. Einen solchen Knoten gibt es jedoch nicht, da alle anderen Knoten den Grad 1 haben.

Im **zweiten Testfall** wollen wir einen gültigen Wald mit 10 Knoten konstruieren: 6 mit Grad 1, 2 mit Grad 2 und 2 mit Grad 3. Ein solcher Wald existiert, und die Ausgabe ist unten abgebildet:



Beachte, dass die Knoten 4 und 5 drei Kanten haben, die mit 1, 2 und 3 bezeichnet sind. Ausserdem haben die Knoten 3 und 6 zwei Kanten mit den Bezeichnungen 1 und 2. Schliesslich haben die Knoten 0, 1, 2, 7, 8 und 9 eine Kante mit der Bezeichnung 1.