

Shop Tour (tour)

In Lineland gibt es N Keksläden in einer Reihe, nummeriert von 0 bis $N-1$. Baq möchte eine *Einkaufstour* durch die Läden machen. Eine Einkaufstour wird durch N **unterschiedliche** ganze Zahlen P_0, \dots, P_{N-1} zwischen 0 und $N-1$ bestimmt.

Bei einer gegebenen Einkaufstour beginnt Baq im Geschäft P_0 . Für jedes $i = 0, \dots, N-1$, bewegt sich Baq vom Geschäft P_i zum Geschäft P_{i+1} (hier sagen wir $P_N = P_0$) und kauft einen Keks von jedem der Geschäfte zwischen P_i und P_{i+1} , einschliesslich. Formal, wenn $L_i = \min(P_i, P_{i+1})$ und $R_i = \max(P_i, P_{i+1})$, dann kauft Baq im i -ten Schritt jeweils einen Keks von jedem der Geschäfte L_i, L_{i+1}, \dots, R_i .

Baq hat nun die Zahlen A_0, \dots, A_{N-1} , wobei A_i die Gesamtzahl der gekauften Kekse bezeichnet, die er im i -ten Geschäft gekauft hat, erinnert sich aber nicht an die Tour durch die Geschäfte. Deine Aufgabe ist es, festzustellen, ob die Informationen im Array A mit einer gültigen Einkaufstour übereinstimmen, und wenn ja, dann konstruiere eine solche gültige Tour. Um eine volle Punktzahl zu erhalten (siehe den Abschnitt über die Punktevergabe für Details) muss die von dir konstruierte Tour die *lexikographisch kleinste* solche Tour sein.


Wir sagen, dass eine Tour P_0, \dots, P_{N-1} *lexikographisch kleiner* ist als eine andere Tour Q_0, \dots, Q_{N-1} , wenn es ein $0 \leq k \leq N-1$ gibt, so dass:

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

Eine Tour Q ist die lexikographisch kleinste unter denen, die mit den Informationen in der Matrix A übereinstimmen wenn es keine andere Tour P mit demselben Array A von gekauften Keksen in jedem Geschäft gibt die lexikografisch kleiner ist als Q .

Implementierung

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

 In den Anhängen zu dieser Aufgabe findest du eine Vorlage `tour.cpp` mit einer Beispielimplementierung.

Du musst die folgende Funktion implementieren:

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

- Die ganze Zahl N steht für die Anzahl der Geschäfte.
- Das Array A , indiziert von 0 bis $N-1$, enthält die Werte A_0, A_1, \dots, A_{N-1} , wobei A_i die Anzahl der Kekse ist die im i -ten Geschäft gekauft wurden.
- Die Funktion sollte entweder einen booleschen Wert oder ein Array von ganzen Zahlen zurückgeben.
 - Wenn keine gültige Einkaufstour existiert, die dem Array A entspricht, sollte die Funktion `false` zurückgeben.
 - Wenn eine gültige Einkaufstour existiert, hast du mehrere Möglichkeiten:
 - * Um die volle Punktzahl zu erhalten, sollte die Funktion ein Array von N ganzen Zahlen P_0, \dots, P_{N-1} zurückgeben, die die **lexikographisch kleinste** Einkaufstour, die zu dem Array A .

- * Um eine Teilbewertung zu erhalten, soll die Prozedur ein Array von N ganzen Zahlen P_0, \dots, P_{N-1} zurückgeben, die eine beliebige nicht-lexikografisch-kleinste Ladentour, die das Array A .
- * Um eine kleinere Teilpunktzahl zu erhalten, sollte die Funktion `true` oder ein beliebiges Array von ganzen Zahlen, die keine gültige Einkaufstour beschreiben was zu dem Array A führt zurückgeben.

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

- Wenn der Rückgabewert `false` ist, wird eine einzelne Zeile mit der Zeichenfolge `NO` ausgegeben.
- Wenn der Rückgabewert `true` oder ein Array von ganzen Zahlen mit einer Länge ungleich N ist, wird eine einzelne Zeile mit der Zeichenkette `YES` ausgegeben.
- Wenn der Rückgabewert ein Array P mit N ganzen Zahlen ist, wird eine einzelne Zeile mit der Zeichenkette `YES` gedruckt, gefolgt von einer Zeile mit den N ganzen Zahlen P_0, \dots, P_{N-1} , getrennt durch Leerzeichen.

Beispielgrader

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

Die Eingabe besteht aus zwei Zeilen, die Folgendes enthalten:

- Zeile 1: die ganze Zahl N .
- Zeile 2: die ganzen Zahlen A_i , getrennt durch Leerzeichen.

Die Ausgabe besteht aus einer oder zwei Zeilen, die die Werte enthalten, die von der Funktion `tour` zurückgegeben werden.

Einschränkungen

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

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 [8 Punkte]:** $N \leq 8$.
- **Teilaufgabe 3 [32 Punkte]:** $N \leq 2 \times 10^3$.
- **Teilaufgabe 4 [16 Punkte]:** $A_i \leq 4$ für alle $i = 0, \dots, N - 1$.
- **Teilaufgabe 5 [20 Punkte]:** Es gibt ein $0 \leq j \leq N - 1$, so dass $A_i \leq A_{i+1}$ für alle $0 \leq i < j$ und $A_i \geq A_{i+1}$ für alle $j \leq i \leq N - 2$.
- **Teilaufgabe 6 [24 Punkte]:** Keine zusätzlichen Beschränkungen.

Für jeden Testfall, in dem eine gültige Einkaufstour existiert, erhält deine Lösung:

- die volle Punktzahl, wenn es die lexikographisch kleinste gültige Shoptour liefert.

- 75% der Punkte, wenn es eine gültige Einkaufstour liefert, die nicht nicht die lexikografisch kleinste ist.
- 50% der Punkte, wenn es `true` oder ein Array zurückgibt, das keine gültige Einkaufstour beschreibt.
- bekommt sonst 0 Punkte.

Für jeden Testfall, in dem keine gültige Einkaufstour existiert, erhält deine Lösung:

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

Beispiele

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

template.output2.txt

Erklärung

Im **ersten Testfall** erzeugt die Tour $P = [0, 2, 1, 3]$ das Feld $A = [2, 4, 4, 2]$ wie folgt:

- Zu Beginn ist die Anzahl der Kekse, die in jedem Geschäft gekauft wurden, $[0, 0, 0, 0]$.
- Baq bewegt sich vom Laden $P_0 = 0$ zum Laden $P_1 = 2$, so dass die Anordnung nach dieser Bewegung $[1, 1, 1, 0]$ ist.
- Baq zieht von Geschäft $P_1 = 2$ nach Geschäft $P_2 = 1$, so dass das Array nach diesem Zug $[1, 2, 2, 0]$ ist.
- Baq bewegt sich von Geschäft $P_2 = 1$ nach Geschäft $P_3 = 3$, so dass die Anordnung nach diesem Zug $[1, 3, 3, 1]$ ist.
- Schliesslich bewegt sich Baq von Laden $P_3 = 3$ zu Laden $P_0 = 0$, so dass die endgültige Anordnung $[2, 4, 4, 2]$ ist.

Es kann gezeigt werden, dass $[0, 2, 1, 3]$ die lexikografisch kleinste solche Tour ist.

Für den **zweiten Testfall** kann gezeigt werden, dass es keine gültige Tour gibt die zu der Anordnung $A = [2, 2, 2]$ führt.