ТРЕНИРОВЪЧНО СЪСТЕЗАНИЕ
МОМИЧЕТА
Пловдив, 12-14 юни 2026 г.
Група G

Задача GT4. ЛЕГОТА

⏳ 0.1 сек. 💾 256 MB

Като за последна задача тази година, Кирчо иска да направи нещо специално за Вас, затова днес ще строите лего!

Кирчо е събрал група от N човека, номерирани с числата $0, \ldots, N-1$, подредени в **кръг** в този ред по часовниковата стрелка, които се опитват да строят различни обекти от лего тухли. Нас единствено ни интересува бройката лего тухли, които даден човек е използвал до момента, която за човек с номер i ще означаваме с положителното число A_i. Първоначално всички обекти се състоят от една лего тухла, тоест $A_i = 1$. Строенето на обектите става по следния начин:

- За един ход точно един човек може да "копира" обекта на човека, съседен на него по часовниковата стрелка, и да го добави към своя обект. Формално се избира индекс i и към A_i се добавя A_{i+1}[1].

Кирчо се е разсеял и не е забелязал, че групата е започнала без него, като единствено вижда сегашното състояние на обектите, в частност и сегашното състояние на редицата $A_0, \ldots, A_{N-1}$. Сега той се опитва да възстанови поредицата от ходове, довела до това състояние, ако съществува такова[2]. Понеже Кирчо е зает, Ви моли да му помогнете, като напишете програмата **`legos`**, която решава задачата.

Детайли по имплементацията

Това е интерактивна задача, което означава, че информация между Вашата програма и програмата на журито няма да се обменя чрез стандартния вход и изход, а чрез специални функции.

Трябва да имплементирате следната функция:

```
std::pair <bool, std::vector <std::pair <int, int>>> solve(std::vector <
    int> A);
```

Функцията се извиква веднъж за всеки **подтест** в текущия тест. Функцията трябва да върне две неща:

- булева променлива, указваща дали е възможно да се достигне до даденото състояние от началното;
- поредицата от ходове, нужна за достигане до даденото състояние.

В случай, че е невъзможно да се достигне до даденото състояние, няма значение какво върнете като втори аргумент, стига първият да е `false`. В противен случай трябва да върне поредицата от ходове като вектор от двойки числа (u_j, v_j), където u_j е номерът на човека, който извършва ход, а v_j е броят пъти, които той последователно извършва хода преди всички следващи описани ходове във вектора.

Трябва да предадете към системата файл `lego.cpp`, който съдържа функцията `solve()`. Той може да съдържа и друг код и функции, необходими за работата на програмата Ви, но **не трябва да съдържа главната функция** `main()`. В началото си Вашият файл трябва да съдържа указание към предпроцесора:

```
#include "legos.h"
```

[1] Понеже хората са наредени в кръг, считаме, че $A_N = A_0$.

[2] За съжаление е възможно някой да не си е построил обекта по гореописаните правила.

ТРЕНИРОВЪЧНО СЪСТЕЗАНИЕ
МОМИЧЕТА
Пловдив, 12-14 юни 2026 г.
Група G

Всеки тест се състои от T подтеста. Тест се счита за верен, ако програмата Ви върне верни отговори и поредици от ходове за всички негови подтестове.

Ограничения

- $1 \leq T \leq 2 \times 10^5$
- $2 \leq N \leq 2 \times 10^5$
- $1 \leq A_i, v_j \leq 10^9$
- $0 \leq u_j < N$
- Гарантирано е, че сумата на N върху всички подтестове за даден тест не надвишава 2×10^5.

Подзадачи

Подзадача	**Точки**	**Необходими подзадачи**	N	T	**Допълнителни ограничения**
0	0	–	–	–	Примерните тестове.
1	13	0	≤ 6	–	$A_i \leq 6$
2	9	–	–	–	$A_0 = 1$
3	12	2	–	–	A_i са степени на 2
4	21	$0 - 1$	$\leq 2 \times 10^3$	≤ 3	–
5	19	$0 - 1, 4$	$\leq 4 \times 10^4$	–	–
6	26	$0 - 5$	–	–	–

Точките за дадена подзадача се получават само ако се преминат успешно всички тестове, предвидени за нея.

Примерна комуникация

№	**Действия на журито**	**Отговор на Вашата програма**
1	`solve({4, 3})`	`return {true, {{1, 2}, {0, 1}}};`

В примера $N = 2$ и следните ходове се прилагат в този ред:

- $i = 1$, тогава A_1 се увеличава с 1 и става равно на 2;
- $i = 1$, тогава A_1 се увеличава с 1 и става равно на 3;
- $i = 0$, тогава A_0 се увеличава с 3 и става равно на 4.

Локално тестване

Предоставени Ви са файловете `legos.h` и `Lgrader.cpp`, които можете да компилирате заедно с Вашата програма, за да я тествате на работния си компютър, както и команди за компилиране на двете програми.

При стартиране на локалния грейдър на първия ред на стандартния вход трябва да въведете цялото положително число T. Следва да въведете T теста. На първия ред на всеки тест трябва да въведете цялото положително число N, а на втория - целите положителни числа $A_0, \ldots, A_{N-1}$. Грейдърът ще изведе върнатите от Вашата програма отговори.