Задача AB23. Триъгълник

⏳10 сек. 💾 1024 МБ

Ръководството на националната комисия, преставлявана от отговорниците на групи A и B, желае да следва стриктно конспекта на задачите, чрез които се избира националния отбор. За да изпълнят това, те ще дадат геометрична задача. И не само – бройката на интерактивните задачи, дадени досега, не изпълнява изискванията. За да разреши кризата в подбора за `IOI`, Сашка реши да даде задача, която е както интерактивна, така и геометрична. Условието е както следва:

Журито е скрило пермутация $P_0, P_1, \ldots, P_{N-1}$ на $\{1, 2, 3, \ldots, N\}$. Трябва да откриете пермутацията. За тази цел, можете да задавате следния въпрос на журито: 'Възможно ли е да се създаде триъгълник с положително лице и страни с дължини P_A, P_B, P_C?'

Известно е (от неравенството на триъгълника), че това е възможно тогава и само тогава, когато:

$$P_A + P_B > P_C,\ P_A + P_C > P_B \quad \text{и} \quad P_B + P_C > P_A.$$

Напишете програма **`triangle`**, съдържаща функция `solve`, която ще бъде компилирана заедно с програмата на журито и ще комуникира с нея, задавайки въпроси от гореописания вид. Накрая на комуникацията трябва да може да определите пермутацията.

Детайли по имплементацията

Трябва да имплементирате функцията `solve`:

```
std::vector<int> solve(int N)
```

- N: дължината на пермутацията.

Функцията ще бъде извикана T пъти в един тест – веднъж за всеки подтест, които имат равни N и трябва да върне скритата пермутация за дадения подтест. За да постигне това, Вашата програма може да вика функцията `query`, имплементирана от журито:

```
bool query(int A, int B, int C)
```

- A, B, C: индексите на страните P_A, P_B, P_C.

Функцията ще върне `true`, ако съществува триъгълник със страни с дължини P_A, P_B, P_C и `false`, в противен случай.

Ограничения

- $N = T = 1\,000$

Подзадачи

Подзадача	Точки	Описание
1	100	Подзадачата съдържа 1 тест с $T = 1\,000$ произволно генерирани пермутации, всяка с по $N = 1\,000$ елемента.

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

Оценяване

Нека $Q_{contestant}$ да бъде средното аритметично на бройката заявки, необходими на вашата програма за едно извикване на `solve` в единствения тест, и нека $Q_{target} = 8770$. Тогава резултатът Ви за задачата ще бъде:

$$\begin{cases} 0 & \text{if } Q_{contestant} > 2 \times 10^6, \\ 100 & \text{if } Q_{contestant} \le Q_{target}, \\ 100 \times \max\left(0.15,\ 1 - \sqrt{1 - \left(\frac{Q_{target}}{Q_{contestant}}\right)^{0.55}}\right) & \text{if } Q_{contestant} > Q_{target}. \end{cases}$$

Примерно взаимодействие

За примерната интеракция $N = 4$, $T = 2$.

Действие на състезателя	Действие на журито
	`solve(4)`
`query(0, 0, 0)`	`true`
`query(0, 1, 2)`	`false`
`query(0, 1, 3)`	`false`
`query(0, 2, 3)`	`true`
`query(1, 2, 3)`	`false`
`return {3, 1, 2, 4};`	
	`solve(4)`
`query(0, 1, 2)`	`false`
`query(0, 1, 3)`	`true`
`query(0, 2, 3)`	`false`
`query(1, 2, 3)`	`false`
`return {4, 2, 1, 3};`	

Формат на грейдъра

Формат на входа:

- ред 1: три цели числа T, N и R – размерът на пермутациите, броят на подтестовете и режимът на работа. Ако $R = 1$, локалният гредът ще генерира произволни пермутации и ще очаква допълнително число – S, което ще бъде параметъра (сийда) за генератора. Ако $R = 2$, входът продължава, както следва:
- ред 2 до $(1 + T)$: пермутация на числата $\{1, 2, \ldots, N\}$.

Формат на изхода:

- ред 1: съобщение за грешка или средното аритметично на броя заявки за всички подтестове, ако всички пермутации са коректно намерени.