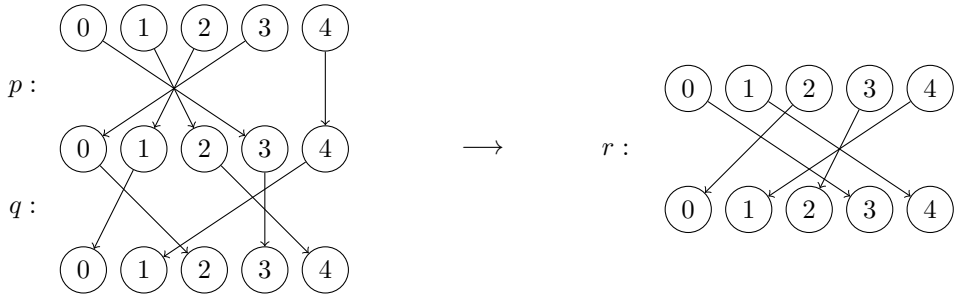
Zadanie: PER
Permutacje

Olimpiada Informatyczna

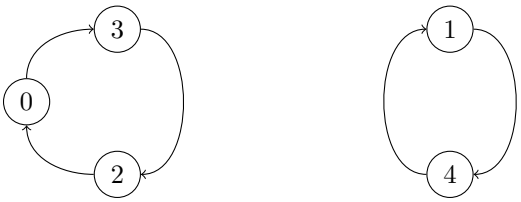
XXXI OI, etap III, dzień drugi. Plik źródłowy `per.*` Dostępna pamięć: 256 MB. *11.04.2024*

Bajtosz dostał się do finału teleturnieju "Jeden z n!". W tym teleturnieju zawodnik ma odgadnąć ukrytą permutację p liczb $0, 1, 2, \ldots, n-1$. W każdej rundzie zawodnik zadaje pytanie opisane przez wybraną przez niego permutację q liczb $0, 1, 2, \ldots, n-1$ i liczby $a, b \in \{0, 1, \ldots, n-1\}$. Prowadzący zaczyna od wyznaczenia permutacji r będącej nałożeniem permutacji q na permutację p. Tak więc dla każdego $i \in \{0, 1, \ldots, n-1\}$, w permutacji r liczba i przechodzi na liczbę $q(p(i))$. Następnie, prowadzący informuje zawodnika, czy liczby a oraz b znajdują się na tym samym cyklu w permutacji r. Mówiąc formalnie, cyklem zawierającym a w permutacji r jest ciąg $a, r(a), r(r(a)), r(r(r(a))), \ldots$ (ciąg możemy zakończyć, gdy po raz drugi pojawi się w nim liczba a). Pytamy się o to, czy w takim ciągu znajduje się liczba b.

Przykładowo, rozważmy permutację $p = [3, 2, 1, 0, 4]$. Ten zapis oznacza, że w permutacji p liczba 0 przechodzi na liczbę 3, liczba 1 na liczbę 2, liczba 2 na liczbę 1, liczba 3 na liczbę 0, a liczba 4 na liczbę 4. Nakładając na nią permutację $q = [2, 0, 4, 3, 1]$, otrzymamy permutację $r = [3, 4, 0, 2, 1]$, co obrazuje poniższy rysunek.



Cykle otrzymanej permutacji r wyglądają tak jak na poniższym rysunku:



Przykładowo, liczby 2 i 3 leżą na tym samym cyklu w permutacji r, a liczby 2 i 4 nie.

Celem zawodnika jest odgadnięcie permutacji p przy użyciu ograniczonej liczby zapytań. Pomóż Bajtoszowi grać tak, aby zdobyć jak najwyższą wygraną w teleturnieju.

Interakcja

To zadanie jest interaktywne. Należy napisać program, który będzie grał w teleturnieju, używając do tego dostarczonej biblioteki. Aby użyć biblioteki, należy wpisać w swoim programie:

- C++: `#include "perlib.h"`
- Python: `from perlib import *`

Twój program zostanie skompilowany/uruchomiony za pomocą komendy:

- C++: `g++ -O3 -static per.cpp perlib.cpp -std=c++20`
- Python: `python3 per.py`

Następująca funkcja daje w wyniku długość permutacji n ($1 \leq n \leq 500$):

- C++: `int daj_n()`
- Python: `daj_n() -> int`

Kolejna funkcja daje w wyniku maksymalną liczbę zapytań z. Jeśli Twój program zada więcej zapytań, to otrzyma werdykt *Błędna odpowiedź*.

- C++: `int daj_z()`
- Python: `daj_z() -> int`

Olimpiada Informatyczna finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2022/2023, 2023/2024, 2024/2025".

Poniższa funkcja daje w wyniku numer podzadania (patrz sekcja *Ocenianie*). W przypadku testu przykładowego jej wynikiem jest 1.

- C++: `int daj_podzadanie()`
- Python: `daj_podzadanie() -> int`

Poniższej funkcji możesz używać, aby dowiedzieć się, czy a i b są na tym samym cyklu w permutacji r będącej nałożeniem permutacji q na (nieznaną Ci) permutację p. Wynikiem funkcji jest `true` (`True` w Pythonie) jeśli a i b są w tym samym cyklu, a `false` (`False` w Pythonie) w przeciwnym przypadku.

- C++: `bool zapytaj(std::vector<int> q, int a, int b)`
- Python: `zapytaj(q: list[int], a: int, b: int) -> bool`

Ostatnia z funkcji podaje permutację p jako odpowiedź.

- C++: `void odpowiedz(std::vector<int> p)`
- Python: `odpowiedz(p: list[int]) -> None`

W języku C++ funkcja ta kończy działanie programu. W języku **Python** natomiast funkcja ta **nie** kończy działania programu. Użytkownik powinien skończyć działanie programu samodzielnie, od razu po wywołaniu tej funkcji.

Reprezentacja permutacji w funkcjach jest następująca:

- C++: Permutacje q i p są podawane jako argumenty funkcji w postaci wektorów (`std::vector`) długości n. Liczba na pozycji i ($0 \leq i < n$) powinna być równa liczbie, na którą przechodzi i w opisywanej permutacji.
- Python: Permutacje q i p są podawane jako listy elementów typu `int` długości n. Liczba na pozycji i ($0 \leq i < n$) powinna być równa liczbie, na którą przechodzi i w opisywanej permutacji. Nie należy przekazywać generatorów albo tablic `numpy`. Generator można zamienić na listę za pomocą instrukcji `list(zmienna)`.

Twój program nie może otwierać żadnych plików ani używać standardowego wejścia i wyjścia. Może on korzystać ze standardowego wyjścia diagnostycznego (`stderr`), jednak pamiętaj, że zużywa to cenny czas.

Przykładowy przebieg programu

Permutacja $p = [3, 2, 1, 0, 4]$, a także permutacje $q = [2, 0, 4, 3, 1]$ i $r = [3, 4, 0, 2, 1]$ występujące w części pytań w poniższym przykładzie zostały przedstawione na rysunkach na pierwszej stronie treści.

Przykładowo, obiekt typu `std::vector` reprezentujący permutację $[3, 2, 1, 0, 4]$ zapisuje się jako `{3,2,1,0,4}`.

Wywołana funkcja	**Wynik**	**Opis**
`daj_n()`	5	$n = 5$, ukryta permutacja p to $[3, 2, 1, 0, 4]$
`zapytaj({0,1,2,3,4}, 0, 1)`	`false`	0 i 1 nie są na tym samym cyklu
`zapytaj({0,1,2,3,4}, 1, 2)`	`true`	1 i 2 są na tym samym cyklu
`zapytaj({0,1,2,3,4}, 2, 3)`	`false`	2 i 3 nie są na tym samym cyklu
`zapytaj({0,1,2,3,4}, 0, 3)`	`true`	0 i 3 są na tym samym cyklu
`zapytaj({2,0,4,3,1}, 4, 1)`	`true`	4 i 1 są na tym samym cyklu w permutacji $r = [3, 4, 0, 2, 1]$
`zapytaj({2,0,4,3,1}, 4, 0)`	`false`	4 i 0 nie są na tym samym cyklu w permutacji $r = [3, 4, 0, 2, 1]$
`zapytaj({2,0,4,3,1}, 2, 4)`	`false`	2 i 4 nie są na tym samym cyklu w permutacji $r = [3, 4, 0, 2, 1]$
`zapytaj({2,0,4,3,1}, 4, 3)`	`false`	4 i 3 nie są na tym samym cyklu w permutacji $r = [3, 4, 0, 2, 1]$
`zapytaj({2,0,4,3,1}, 0, 3)`	`true`	0 i 3 są na tym samym cyklu w permutacji $r = [3, 4, 0, 2, 1]$
`odpowiedz({3,2,1,0,4})`	—	tylko ta permutacja jest zgodna z powyższymi odpowiedziami

Olimpiada Informatyczna finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2022/2023, 2023/2024, 2024/2025”.

Ocenianie

Zestaw testów dzieli się na następujące podzadania. Testy do każdego podzadania składają się z jednej lub większej liczby osobnych grup testów.

Podzadanie	Ograniczenia	Punkty
1	$n \leq 6$, $z = 100\,000$	10
2	$n \leq 30$, $z = 100\,000$	20
3	$n \leq 100$, $z = 15\,000$	20
4	$n \leq 500$, $z = 15\,000$	10
5	$n \leq 500$, $z = 7000$, p jest jednym cyklem	10
6	$n \leq 500$, $z = 7000$	20
7	$n \leq 500$, $z = 5000$	10

Biblioteka ustala na samym początku permutację p, a następnie odpowiada na kolejne pytania.

Eksperymenty

W sekcji *Pliki i testy* w SIO możesz znaleźć przykładową implementację biblioteki oraz test przykładowy w formacie przyjmowanym przez tę bibliotekę.

Przykładowa biblioteka w pierwszym wierszu wejścia wczytuje kolejno n oraz z, a w drugim wierszu n liczb oznaczających kolejne wartości permutacji p: $p(0), p(1), \ldots, p(n-1)$. Funkcja `daj_podzadanie()` daje w wyniku -1.

Implementacja biblioteki używanej do oceny Twoich rozwiązań może się różnić od przykładowej.

Uwaga: W tym zadaniu nie są dostępne uruchomienia próbne w SIO ani skrypt `ocen` na komputerach.