Zadanie: BST
Bardzo skomplikowany test [B]

POTYCZKI ALGORYTMICZNE

Potyczki Algorytmiczne 2020, runda piąta. Limity: 512 MB, 3 s. *11.12.2020*

Bajtek właśnie przystąpił do egzaminu ustnego z *Algorytmów i Struktur Danych*. Nie uczył się do niego za długo, nie szło mu zatem najlepiej. Po kilku minutach rozmowy załamany wykładowca postanowił dać chłopcu ostatnią szansę.

– Wiesz chociaż chłopcze, czym są drzewa *BST*? – zapytał profesor.

Bajtek uśmiechnął się pod nosem, słysząc to, bowiem podczas jego drzemki na wykładzie trochę teorii zapadło mu w pamięć.

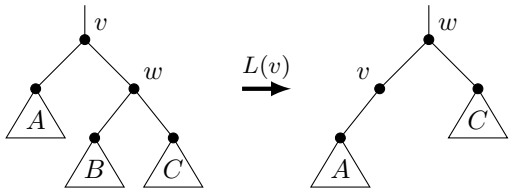
– Tak, drzewo BST o rozmiarze n to ukorzenione drzewo, którego wierzchołki numerowane są liczbami całkowitymi od 1 do n. Każdy wierzchołek może mieć co najwyżej dwójkę dzieci; może mieć lewego (co najwyżej jednego) syna oraz prawego (co najwyżej jednego) syna. Dodatkowo, numer każdego wierzchołka musi być większy niż numery wszystkich wierzchołków w jego lewym poddrzewie oraz mniejszy niż numery wszystkich wierzchołków w jego prawym poddrzewie – odpowiedział Bajtek, sięgając do głębin swojej podświadomości.

– Dobrze. Sprawdzimy, czy pamiętasz, jak się wykonuje na nich rotacje – odpowiedział, siedzący dotąd profesor, po czym wstał i ruszył w stronę tablicy.

Bajtka zalał zimny pot. Momentalnie stracił pewność siebie, nie pamiętał bowiem, jak dokładnie działają rotacje (zapewne w tym momencie wykładu przewracał się na drugi bok, i to musiało zagłuszyć słowa wykładowcy). Egzaminator narysował na tablicy dwa drzewa BST tego samego rozmiaru i nakazał Bajtkowi za pomocą poprawnych rotacji przeobrazić pierwsze z nich w drugie.

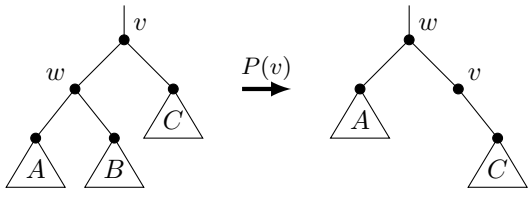
Bajtek pomyślał chwilę i założył, że lewa rotacja to wybranie pewnego wierzchołka v i jego prawego syna w oraz sprawienie, by to w był ojcem v. Intuicję Bajtka formalizuje następujący pseudokod:

```
if v.Ojciec != null then
    if v.Ojciec.PrawySyn == v then
        v.Ojciec.PrawySyn := w
    else
        v.Ojciec.LewySyn := w
w.Ojciec := v.Ojciec
v.Ojciec := w
w.LewySyn := v
v.PrawySyn := null
```



Analogicznie Bajtek rozumie prawą rotację, gdzie w jest lewym synem wierzchołka v:

```
if v.Ojciec != null then
    if v.Ojciec.PrawySyn == v then
        v.Ojciec.PrawySyn := w
    else
        v.Ojciec.LewySyn := w
w.Ojciec := v.Ojciec
v.Ojciec := w
w.PrawySyn := v
v.LewySyn := null
```



Jednak chłopiec szybko zauważył, że coś jest nie tak. Jeśli podczas lewej rotacji wierzchołek w miał jakieś lewe poddrzewo, to zostanie ono zgubione! To samo może stać się z ewentualnym prawym poddrzewem wierzchołka w podczas prawej rotacji.

– Pospiesz się chłopcze, nie tylko Ty chciałbyś zdać ten egzamin – ponaglił go zniecierpliwiony profesor.

Nie mając zbyt wiele czasu do namysłu, Bajtek założył, że rotację może wykonać tylko wtedy, gdy owe problematyczne poddrzewo jest puste, tzn. gdy nie zgubi żadnego wierzchołka, a drzewo pozostanie spójne.

Chcąc jak najszybciej zakończyć swoje udręki, postanowił, że wykona minimalnie wiele rotacji, które pozwolą mu przeistoczyć pierwsze drzewo w drugie. Dasz radę stwierdzić, czy jest to możliwe, i jeśli tak, to ile rotacji będzie musiał wykonać? Ponieważ liczba ta może być dość duża, wystarczy jak podasz jej resztę z dzielenia przez $10^9 + 7$.

Wejście

W pierwszym wierszu wejścia znajduje się jedna liczba całkowita n ($1 \leq n \leq 5 \cdot 10^5$) oznaczająca rozmiar drzew narysowanych przez profesora.

W następnych dwóch wierszach znajdują się opisy tych drzew. Opis drzewa składa się z ciągu n liczb całkowitych $a_1, a_2, \ldots, a_n$ ($-1 \leq a_i \leq n$, $a_i \neq 0$); wartość $a_i \geq 1$ oznacza, że ojcem i-tego wierzchołka jest wierzchołek a_i; wartość $a_i = -1$ oznacza, że i-ty wierzchołek jest korzeniem całego drzewa.

Możesz założyć, że oba drzewa są poprawnymi drzewami BST, tzn. nie ma w nich cykli, jest dokładnie jeden korzeń, a każdy wierzchołek ma co najwyżej jednego syna mniejszego od siebie i jednego syna większego od siebie (oraz zachodzi warunek z porównywaniem wierzchołka z jego poddrzewami).

Wyjście

Na wyjściu powinna znaleźć się jedna liczba całkowita, oznaczająca resztę z dzielenia przez $10^9 + 7$ minimalnej możliwej liczby rotacji (w rozumieniu Bajtka) potrzebnych do przekształcenia pierwszego drzewa w drugie, lub -1, jeśli takie przekształcenie nie jest możliwe.

Przykład

Dla danych wejściowych:

```
4
3 1 -1 3
2 -1 4 2
```

poprawnym wynikiem jest:

```
3
```

Natomiast dla danych wejściowych:

```
8
2 4 2 7 4 5 -1 7
2 3 6 5 3 -1 6 7
```

poprawnym wynikiem jest:

```
7
```

Wyjaśnienie pierwszego przykładu: Na poniższym rysunku przedstawiono minimalną liczbę rotacji przekształcającą drzewa: