USACO 2017 December Contest

Silver, Problem 3: The Bovine Shuffle

Prof. Edson Alves

Faculdade UnB Gama

Convinced that happy cows generate more milk, Farmer John has installed a giant disco ball in his barn and plans to teach his cows to dance!

Looking up popular cow dances, Farmer John decides to teach his cows the "Bovine Shuffle". The Bovine Shuffle consists of his N cows $(1 \leq N \leq 100,000)$ lining up in a row in some order, then performing successive "shuffles", each of which potentially re-orders the cows. To make it easier for his cows to locate themselves, Farmer John marks the locations for his line of cows with positions $1 \ldots N$, so the first cow in the lineup will be in position 1, the next in position 2, and so on, up to position N.

Convencido que vacas felizes dão mais leite, o fazendeiro John instalou uma bola de discoteca gigante no celeiro e planeja ensinar suas vacas a dançar!

Procurando por danças bovinas populares, o fazendeiro John decidiu ensinar a suas vacas a "Misturada Bovina". A Misturada Bovina consiste em alinhar suas N vacas $(1 \leq N \leq 100.000)$ em uma linha, em alguma ordem. Então elas executam sucessivas "misturas", cada uma delas potencialmente reordenando as vacas. Para que as vacas possam se localizar com mais facilidade, o fazendeiro John marcou posições na linha com números de $1 \ldots N$, de modo que a primeira vaca se alinhe na posição 1, a próxima na posição 2, e assim por diante, até a posição N.

A shuffle is described with N numbers, $a_1 \ldots a_N$, where a cow in position i moves to position a_i during the shuffle (and so, each a_i is in the range $1 \ldots N$). Every cow moves to its new location during the shuffle. Unfortunately, all the a_i's are not necessarily distinct, so multiple cows might try to move to the same position during a shuffle, after which they will move together for all remaining shuffles.

Farmer John notices that some positions in his lineup contain cows in them no matter how many shuffles take place. Please help him count the number of such positions.

Uma mistura é descrita por N números, $a_1 \dots a_N$, onde a vaca que ocupa a posição i se move para a posição a_i durante a mistura (e assim por diante, cada a_i está no intervalo $1 \dots N$). Cada vaca se move para sua nova localização durante a mistura. Infelizmente, os a_i's não são necessariamente distintos, de modo que múltiplas vacas podem tentar se mover para a mesma posição durante a mistura, e após isso elas deve se mover juntas durante todas as misturas restantes.

O fazendeiro John notou que algumas posições na linha sempre tinham vacas sobre elas independentemente do número de misturas feitas. Por favor o ajude a contar o número de tais posições.

Input

The first line of input contains N, the number of cows. The next line contains the N integers $a_1 \ldots a_N$.

Output

Please output the number of positions that will always contain cows, no matter how many shuffles take place.

Entrada

A primeira linha da entrada contém N, o número de vacas. A próxima linha contém os N inteiros $a_1 \ldots a_N$.

Saída

Por favor imprima o número de posições que sempre terão vacas, independentemente de quantas misturas sejam feitas.

Exemplo de entrada e saída

Exemplo de entrada e saída

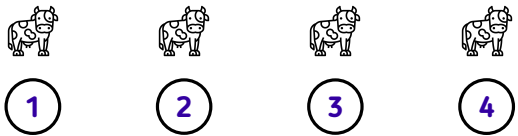
```
4
```

Exemplo de entrada e saída

4

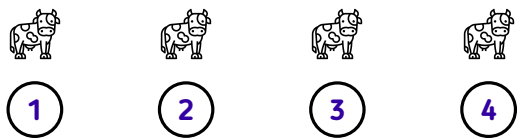
Exemplo de entrada e saída

```
4
```



Exemplo de entrada e saída

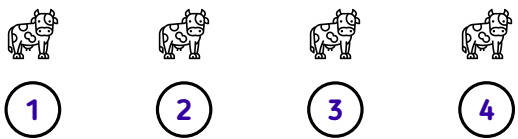
```
4
3 2 1 3
```



Exemplo de entrada e saída

```
4
3 2 1 3
```

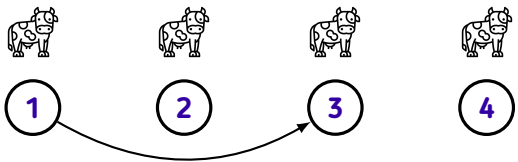
a_1



Exemplo de entrada e saída

```
4
3 2 1 3
```

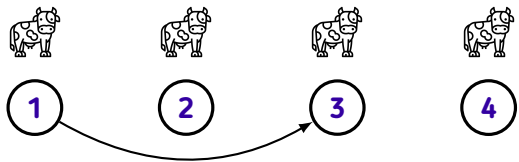
a_1



Exemplo de entrada e saída

```
4
3 2 1 3
```

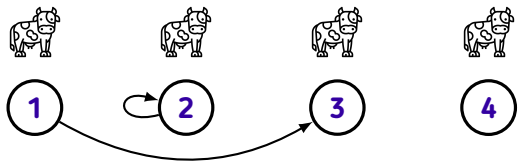
a_2



Exemplo de entrada e saída

```
4
3 2 1 3
```

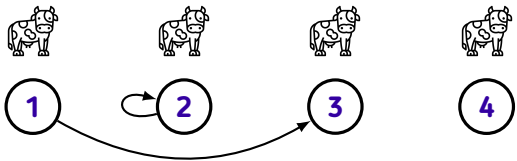
a_2



Exemplo de entrada e saída

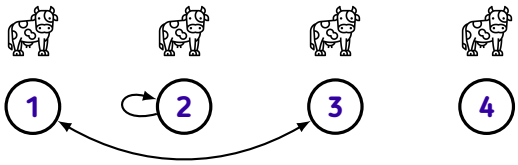
```
4
3 2 1 3
```

a_3



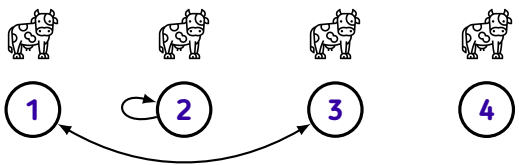
Exemplo de entrada e saída

```
4
3 2 1 3
```



Exemplo de entrada e saída

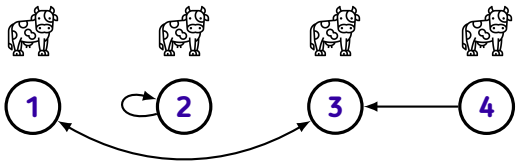
```
4
3 2 1 3
```



Exemplo de entrada e saída

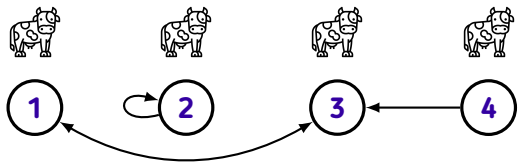
```
4
3 2 1 3
```

a_4



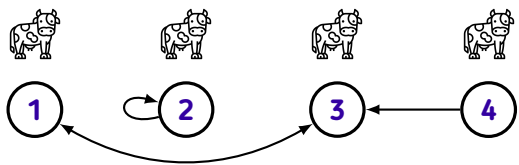
Exemplo de entrada e saída

```
4
3 2 1 3
```



Exemplo de entrada e saída

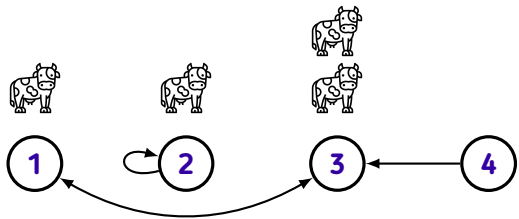
```
4
3 2 1 3
```



Mistura #1

Exemplo de entrada e saída

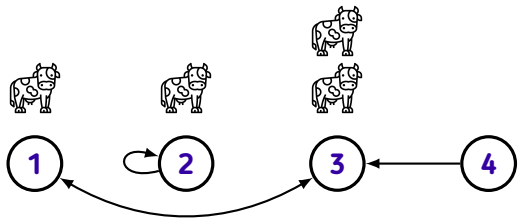
```
4
3 2 1 3
```



Mistura #1

Exemplo de entrada e saída

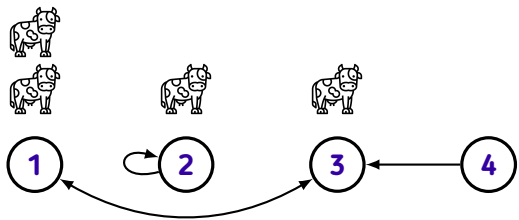
```
4
3 2 1 3
```



Mistura #2

Exemplo de entrada e saída

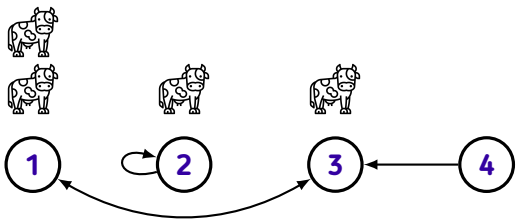
```
4
3 2 1 3
```



Mistura #2

Exemplo de entrada e saída

```
4
3 2 1 3
```



Mistura #2

Solução

Solução

No problema, as posições são os vértices e as movimentações as arestas de um grafo G

Solução

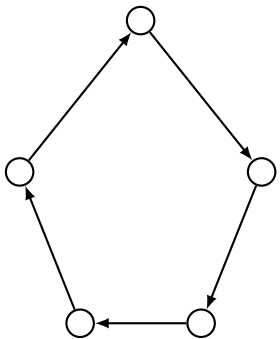
Este grafo G é um grafo de sucessores

Solução

Assim, ele tem um mais um componentes, onde cada componente tem um ciclo e um ou mais caminhos que levam a este ciclo

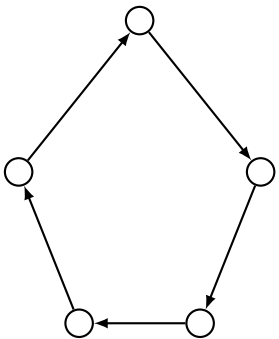
Solução

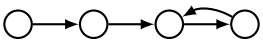
Assim, ele tem um mais um componentes, onde cada componente tem um ciclo e um ou mais caminhos que levam a este ciclo



Solução

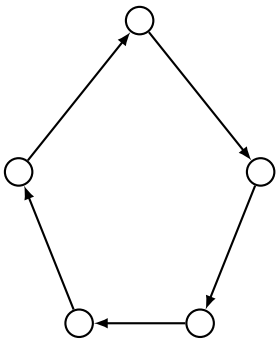
Assim, ele tem um mais um componentes, onde cada componente tem um ciclo e um ou mais caminhos que levam a este ciclo

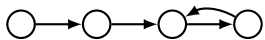


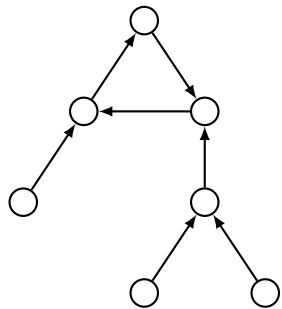


Solução

Assim, ele tem um mais um componentes, onde cada componente tem um ciclo e um ou mais caminhos que levam a este ciclo

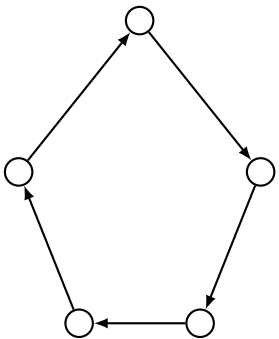


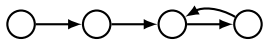


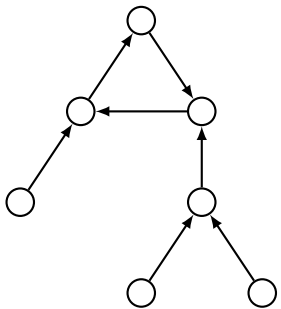


Solução

As vacas convergem, a cada mistura, para os ciclos

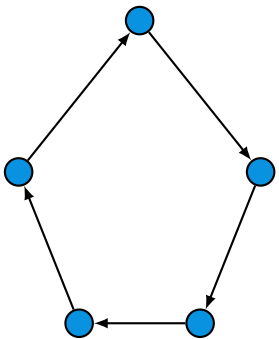


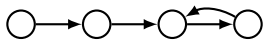


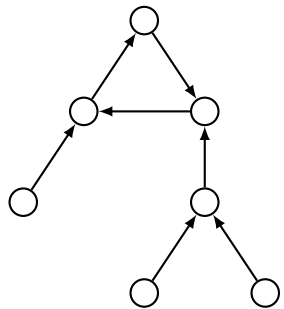


Solução

As vacas convergem, a cada mistura, para os ciclos

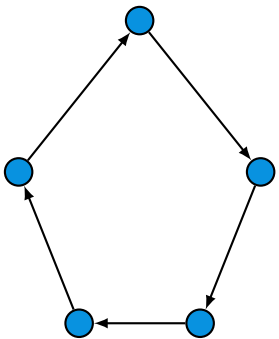


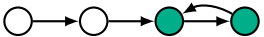


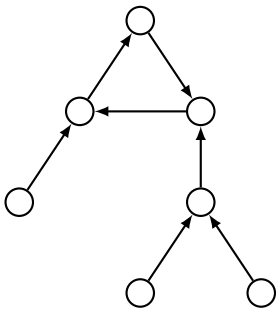


Solução

As vacas convergem, a cada mistura, para os ciclos

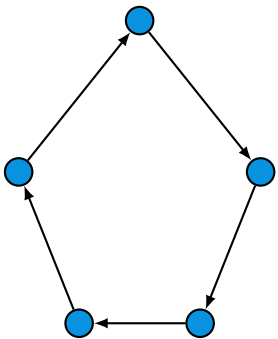


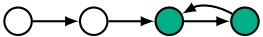


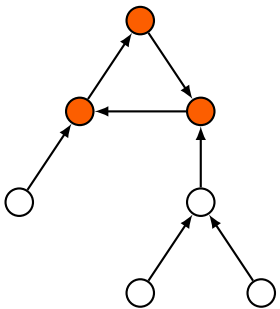


Solução

As vacas convergem, a cada mistura, para os ciclos

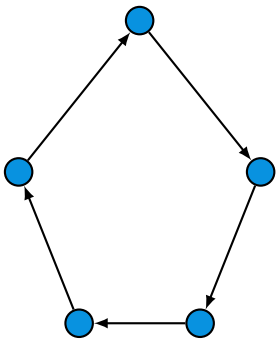


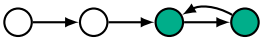


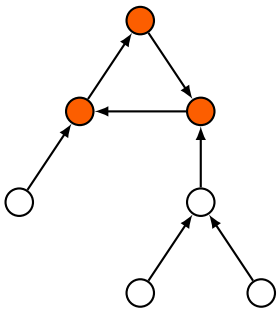


Solução

Assim, a solução é a soma dos comprimentos destes ciclos







```
auto solve(int N, const vector<int>& as)
{
    for (int u = 1; u <= N; ++u)
        ++in_degree[as[u]];

    priority_queue<ii, vector<ii>, greater<ii>> pq;

    for (int u = 1; u <= N; ++u)
        pq.emplace(in_degree[u], u);

    auto ans = 0;

    while (not pq.empty())
    {
        auto [_, u] = pq.top();
        pq.pop();

        if (found[u])
            continue;
```

```
        auto next = 1, len = 0;
        set<int> s;

        while (num[u] == 0)
        {
            num[u] = next++;
            found[u] = true;

            s.insert(u);
            u = as[u];
        }

        if (s.count(u))
            len = next - num[u];

        ans += len;
    }

    return ans;
}
```

Créditos

Cow icon made by surang from *www.flaticon.com*.