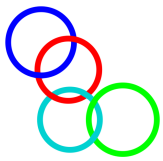
Gyűrűk

Régen Leonardo írta le az első ejtőernyő változatot.

Összekapcsolt gyűrűk

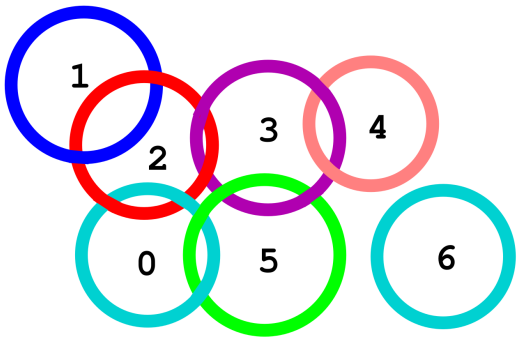
A modern ejtőernyőkben gyűrűket kapcsolnak össze (a gyűrűk kinyithatók, majd összekapcsolás után visszazárhatók). Speciális összekapcsolás a lánc, olyan gyűrűk sorozata, ahol minden gyűrű maximum 2 szomszédjával van összekapcsolva, az ábrának megfelelően. Az egyetlen gyűrű is láncnak számít.



Más elrendezés is lehetséges, ahol van olyan gyűrű, ami 3 vagy több gyűrűvel van összekapcsolva. Egy gyűrűt kritikusnak nevezünk, ha elhagyásával a megmaradt gyűrűk csak láncokban lesznek (vagy nem marad egy gyűrű sem).

Példa

Az ábrán látható 7 gyűrű, 0-tól 6-ig sorszámozva. Két kritikus gyűrű van. Az egyik a 2-es, mert elhagyva 3 lánc marad: [1], [0, 5, 3, 4] és [6]. A másik a 3-as, mert elhagyva 3 lánc marad: [1, 2, 0, 5], [4] és [6]. Bármely másik gyűrű eltávolításával nem csak lánc marad. Például az 5-ös elhagyásával a 0, 1, 2, 3 és 4 nem lánc.



Feladat

Írj programot, amely kiszámolja egy adott gyűrűelrendezésre megadja a kritikus gyűrűk számát!

Kezdetben van bizonyos számú különálló gyűrű. Egyes gyűrűket összekapcsolunk egymással. Kérésre bármely pillanatban meg kell tudnod mondani a kritikus gyűrűk számát. Három eljárást kell megvalósítanod:

- `Init`(N) — pontosan egyszer hívható, a kommunikáció kezdetén, a gyűrűk száma N, a gyűrűket 0-tól N-1-ig sorszámozzuk.
- `Link`(A, B) — az A és a B gyűrűt kell összekapcsolni. A és B biztos különböző és még nincs közvetlenül összekapcsolva a művelet előtt. A `Link`(A, B) és a `Link`(B, A) ugyanazt eredményezi.
- `CountCritical`() — meg kell adja az adott pillanatban a kritikus gyűrűk számát.

Példa

Az ábrán N=7 esetén látható hívások lehetséges sorozata és a helyes eredmény.

Hívás	Eredmény
`Init`(7)	
`CountCritical`()	7
`Link`(1, 2)	
`CountCritical`()	7
`Link`(0, 5)	
`CountCritical`()	7
`Link`(2, 0)	
`CountCritical`()	7
`Link`(3, 2)	
`CountCritical`()	4
`Link`(3, 5)	
`CountCritical`()	3
`Link`(4, 3)	
`CountCritical`()	2

1. részfeladat [20 pont]

- $N \leq 5\,000$.
- A `CountCritical`- egyszer hívják, az összes többi hívás után. A `Link`-t legfeljebb 5 000-szer hívják.

2. részfeladat [17 pont]

- $N \leq 1\,000\,000$.
- A `CountCritical`- egyszer hívják, az összes többi hívás után. A `Link`-t legfeljebb 1 000 000-szor hívják.

3. részfeladat [18 pont]

- $N \leq 20\,000$.
- A `CountCritical`-t legfeljebb 100-szor hívják, az összes többi hívás után. A `Link`-t legfeljebb 10 000-szer hívják.

4. részfeladat [14 pont]

- $N \leq 100\,000$.
- A `CountCritical`-t és a `Link`-t együtt legfeljebb 100 000-szer hívják.

5. részfeladat [31 pont]

- $N \leq 1\,000\,000$.
- A `CountCritical`-t és a `Link`-t együtt legfeljebb 1 000 000-szor hívják.

Megvalósítás

Egy file-t kell beküldened: rings.c, rings.cpp vagy rings.pas. Ebben kell megvalósítanod a 3 eljárást!

C/C++ program

```
void Init(int N);
void Link(int A, int B);
int CountCritical();
```

Pascal program

```
procedure Init(N : LongInt);
procedure Link(A, B : LongInt);
function CountCritical() : LongInt;
```

Írhatsz saját eljárásokat is, a megoldásod nem használhatja a standard inputot és outputot és más file-t sem.

Minta értékelő

A minta értékelő (grader) a bemenetet az alábbi formában várja:

- 1. sor: N, L;
- 2, …, L + 1. sorok:
  - -1 jelentése: `CountCritical` hívás;
  - A, B jelentése:`Link(A,B)` hívás.

A minta értékelő minden `CountCritical` hívás eredményét kiírja a standard kimenetre.