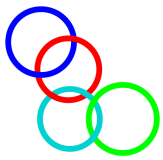
პარაშუტის რგოლები

ადრეული და საკმაოდ რთული ვერსია იმისა, რასაც დღეს ჩვენ პარაშუტს ვეძახით, აღწერილია ლეონარდოს "Codex Atlanticus"-ში (დაახლოებით 1485 წ.). ლეონარდოს პარაშუტი წარმოადგენდა პირამიდის ფორმის ხის კარკასზე გადაჭიმულ მკვრივ ქსოვილს.

"გადაბმული რგოლები"

პარაშუტისტმა ადრიან ნიკოლასმა ლეონარდოს პარაშუტი 500 წლის შემდეგ გამოსცადა, რისთვისაც თანამედროვე მსუბუქი კონსტრუქციის გამოყენებით ლეონარდოს პარაშუტი ადამიანის სხეულზე მიამაგრა. ჩვენ გვსურს გამოვიყენოთ გადაბმული რგოლები, რომლებიც აგრეთვე ქსოვილზე მისამაგრებლადაც გამოიყენება. ყოველი რგოლი მსუბუქი და მტკიცე მასალისაგან არის დამზადებული. რგოლები შეიძლება ადვილად გადავაბათ ერთმანეთს და ყოველი რგოლი ასევე ადვილად შეიძლება გავხსნათ და ისევ ჩავკეტოთ. გადაბმული რგოლების გარკვეულ კონფიგურაციას ჯაჭვი ეწოდება. ჯაჭვი წარმოადგენს რგოლების ისეთ მიმდევრობას, რომელშიც თითოეული რგოლი გადაბმულია მხოლოდ მის მეზობელ (მაქსიმუმ ორ) რგოლთან ისე, როგორც ქვემოთ მოცემულ ნახაზზეა ნაჩვენები. ამ მიმდევრობას უნდა ჰქონდეს დასაწყისი და დასასრული (რგოლები, რომელთაგან თითოეული მაქსიმუმ თითო სხვა რგოლთან არიან დაკავშირებული). რა თქმა უნდა, ერთი რგოლიც ჯაჭვს წარმოადგენს.

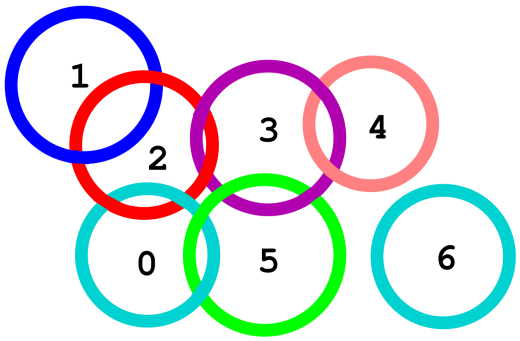


ცხადია, რომ შესაძლებელია სხვა კონფიგურაციებიც, რადგან რგოლი შეიძლება გადაებას სამ ან მეტ რგოლს. ვიტყვით, რომ რგოლი "კრიტიკულია", თუ მისი მოხსნის შემდეგ ყველა დანარჩენი რგოლი ქმნის ჯაჭვების სიმრავლეს (ან სხვა რგოლები საერთოდ აღარ რჩება). სხვა სიტყვებით, ამ რგოლის მოხსნის შემდეგ მხოლოდ ჯაჭვები უნდა დარჩეს.

"მაგალითი"

განვიხილოთ შემდეგ ნახაზზე მოცემული 7 რგოლი, რომლებიც 0-დან 6-მდე რიცხვებითაა გადანომრილი. აქ გვაქვს 2 კრიტიკული რგოლი. ერთ-ერთი მათგანია რგოლი 2: მისი მოხსნის შემდეგ დარჩენილი რგოლები ქმნიან ჯაჭვებს [1], [0, 5, 3, 4] და [6]. მეორე კრიტიკული რგოლია 3: მისი მოხსნის

შემდეგ დარჩენილი რგოლები ქმნიან ჯაჭვებს [1, 2, 0, 5], [4] და [6]. სხვა რომელიმე რგოლის მოხსნით ჩვენ ვერ მივიღებთ განცალკევებული ჯაჭვების მიმდევრობას. მაგალითად, მე-5 რგოლის მოხსნის შემდეგ [6] კი არის ჯაჭვი, მაგრამ გადაბმული რგოლები 0, 1, 2, 3 და 4 ჯაჭვს არ ქმნიან.



ამოცანა

თქვენი ამოცანაა დათვალოთ კრიტიკული რგოლების რაოდენობა მოცემულ კონფიგურაციაში მისი თქვენს პროგრამასთან ურთიერთქმედების შემდეგ.

დასაწყისში გვაქვს განცალკევებული რგოლების გარკვეული რაოდენობა. ამის შემდეგ ხდება რგოლების ერთმანეთთან დაკავშირება. დროის ნებისმიერ მომენტში თქვენ შეიძლება მოგთხოვონ დააბრუნოთ კრიტიკული რგოლების რაოდენობა მიმდინარე კონფიგურაციაში. კერძოდ, თქვენ რეალიზაცია უნდა გაუკეთოთ სამ პროცედურას:

- `Init`(N) — გამოიძახება ზუსტად ერთხელ დასაწყისში იმის შესატყობინებლად, რომ საწყისი კონფიგურაცია შეიცავს N რაოდენობის განცალკევებულ რგოლს, რომლებიც გადანომრილია 1-დან (N-1)-მდე (ჩათვლით).
- `Link`(A, B) — ხდება ორი რგოლის გადაბმა, რომელთა ნომრებია A და B. გარანტირებულია, რომ A და B განსხვავებულია და ისინი ჯერ არ არიან ერთმანეთთან უშუალოდ დაკავშირებული. გარდა ამისა, არ არსებობს რაიმე დამატებითი პირობები A-სა და B-სათვის. კერძოდ, ფიზიკური შეზღუდვებიდან წარმოქმნილი რაიმე პირობები. ცხადია, რომ `Link`(A, B) და `Link`(B, A) ექვივალენტურია.
- `CountCritical`() — გადაბმული რგოლების მიმდინარე კონფიგურაციისათვის აბრუნებს კრიტიკული რგოლების რაოდენობას.

"მაგალითი"

განვიხილოთ ჩვენი ნახაზი N = 7 რგოლისათვის და დავუშვათ, რომ ისინი დასაწყისში განცალკევებულია. ჩვენ ვაჩვენებთ პროცედურების გამოძახების შესაძლებელ მიმდევრობას, სადაც ბოლო გამოძახების შემდეგ ვღებულობთ ქვემოთ მოცემულ ნახაზზე გამოსახულ სიტუაციას.

გამოძახება	დაბრუნებული მნიშვნელობა
`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

ქვეამოცანა 1 [20 ქულა]

- $N \leq 5\,000$.

`CountCritical` ფუნქციის გამოძახება ხდება მხოლოდ ერთხელ, ყველა სხვა გამოძახების შემდეგ. `Link` ფუნქციის გამოძახება ხდება არა უმეტეს 5 000-ჯერ.

ქვეამოცანა 2 [17 ქულა]

- $N \leq 1\,000\,000$.

`CountCritical` ფუნქციის გამოძახება ხდება მხოლოდ ერთხელ, ყველა სხვა გამოძახების შემდეგ. `Link` ფუნქციის გამოძახება ხდება არა უმეტეს 1 000 000-ჯერ.

ქვეამოცანა 3 [18 ქულა]

- $N \leq 20\,000$.

`CountCritical` ფუნქციის გამოძახება ხდება არაუმეტეს 100-ჯერ. `Link` ფუნქციის გამოძახება ხდება არა უმეტეს 10 000-ჯერ.

ქვეამოცანა 4 [14 ქულა]

- $N \leq 100\,000$.

`CountCritical` და `Link` ფუნქციების გამოძახება ხდება ჯამურად არაუმეტეს 100 000-ჯერ.

ქვეამოცანა 5 [31 ქულა]

- $N \leq 1\,000\,000$.

`CountCritical` და `Link` ფუნქციების გამოძახება ხდება ჯამურად არაუმეტეს 1 000 000-ჯერ.

რეალიზაციის დეტალები

თქვენ შესამოწმებლად უნდა გაგზანოთ ზუსტად ერთი ფაილი, სახელით `rings.c`, `rings.cpp` ან `rings.pas`. ეს ფაილი უნდა შეიცავდეს ზემოთ აღწერილი ქვეპროგრამების რეალიზაციას და უნდა იყენებდეს შემდეგ აღწერებს (სიგნატურებს):

პროგრამები C/C++ -ზე

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

პროგრამები Pascal -ზე

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

ეს ქვეპროგრამები უნდა მუშაობდნენ ისე, როგორც ზემოთაა აღწერილი. რა თქმა უნდა, თქვენ შეგიძლიათ რეალიზაცია გაუკეთოთ სხვა ქვეპროგრამებს თქვენი შიდა გამოყენებისათვის. თქვენს მიერ შემოწმებაზე გაგზავნილი ამოხსნები არანაირად არ უნდა იყენებდეს სტანდარტულ შეტანა/გამოტანას ან სხვა ნებისმიერ ფაილს.

შეფასების მაგალითი

შემფასებლის ნიმუში (grader) შესატან მონაცემებს კითხულობს შემდეგ ფორმატში:

- სტრიქონი 1: შეიცავს ორ N და L რიცხვს;
- სტრიქონები 2-დან (L + 1)-მდე შეიცავს:

```
** -1-ს CountCritical პროცედურის გამოსაძახებლად;
** A და B პარამეტრებს Link პროცედურის გამოსაძახებლად.
```

შემფასებლის ნიმუში (grader) გამოიტანს `CountCritical` ფუნქციის გამოძახებათა ყველა შედეგს.