ამოცანა ნავიგაცია

⏳ 3 sec. 💾 512 MB

არსებობს **ბმული არაორიენტირებული მარტივი კაქტუსის გრაფი**[1] $N \leq 1000$ ცალი წვეროთი და M ცალი წიბოთი. წვეროებს აქვთ ფერები (აღნიშნულია არაუარყოფითი მთელი რიცხვებით 0-დან 1499-მდე). თავდაპირველად ყველა წვეროს აქვს 0 ფერი. **დეტერმინისტული და მეხსიერების გარეშე რობოტი**[2] იკვლევს გრაფს წვეროდან წვეროზე გადაადგილების გზით. ის ყველა წვეროს ერთხელ მაინც უნდა ეწვიოს და შემდეგ შეწყვიტოს მოძრაობა.

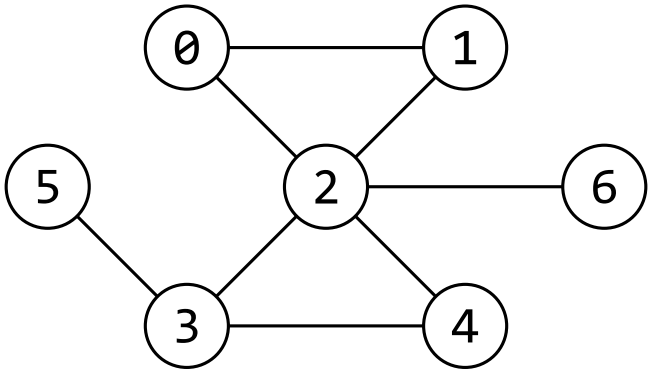
რობოტი იწყებს მუშაობას გრაფის ერთ-ერთი ნებისმიერი წვეროდან. თითოეულ ნაბიჯზე ის ხედავს მისი მიმდინარე წვეროს ფერს და ყველა მეზობელი წვეროს ფერებს **გარკვეული მიმდინარე წვეროსთვის ფიქსირებული თანმიმდევრობით** (ანუ წვეროს ხელახლა მონახულებისას რობოტი მიიღებს მეზობელი წვეროებს იგივე თანმიმდევრობით, მაშინაც კი, თუ მათი ფერები განსხვავდება წინანდელისაგან). რობოტი ასრულებს შემდეგი ორი მოქმედებიდან ერთ-ერთს:

1. ღებულობს გადაწყვეტილებას დაასრულოს მუშაობა.
2. ირჩევს ახალ (ან შესაძლოა იგივე) ფერს მიმდინარე წვეროსათვის და ერთ-ერთ მეზობელ წვეროს, რომელზეც გადავა. მეზობელი წვერო იდენტიფიცირდება ინდექსით 0-დან $D-1$-მდე, სადაც D არის მეზობელი წვეროების რაოდენობა.

მეორე შემთხვევაში, მიმდინარე წვერო იცვლება (ან შესაძლოა იგივე ფერი რჩება) და რობოტი გადადის არჩეულ მეზობელ კვანძზე. ეს მეორდება მანამ, სანამ რობოტი არ დაასრულებს მუშაობას ან არ მიაღწევს იტერაციის ლიმიტს. რობოტი იმარჯვებს, თუ ის ეწვევა ყველა წვეროს და შემდეგ დაასრულებს მუშაობას $L = 3000$ ნაბიჯის იტერაციის ლიმიტის ფარგლებში (წინააღმდეგ შემთხვევაში ის აგებს).

თქვენ რობოტისთვის უნდა შეიმუშაოთ სტრატეგია, რომელსაც შეუძლია პრობლემის გადაჭრა ნებისმიერი ასეთი კაქტუსის გრაფისთვის. გარდა ამისა, თქვენ უნდა შეეცადოთ მინიმუმამდე დაიყვანოთ თქვენს მიერ გამოყენებული განსხვავებული ფერების რაოდენობა. აქ ფერი 0 ყოველთვის ითვლება გამოყენებულად.

[1] *ბმული არაორიენტირებული მარტივი კაქტუსის გრაფი არის ბმული არაორიენტირებული მარტივი გრაფი (ყველა წვერო ხელმისაწვდომია ყველა სხვა წვეროდან; წიბოები ორმხრივია; არ აქვს მარყუჟები ან მულტი-წიბოები), რომელშიც ყველა წიბო ეკუთვნის მაქსიმუმ ერთ მარტივ ციკლს (მარტივი ციკლი არის ციკლი, რომელიც შეიცავს თითოეულ წვეროს მაქსიმუმ ერთხელ). ქვემოთ მოცემული სურათი მაგალითია.*



[2]*რობოტი დეტერმინისტული და მეხსიერების გარეშეა, თუ მისი მოქმედება დამოკიდებულია მხოლოდ მის მიმდინარე მონაცემებზე (ანუ ის არ ინახავს არანაირ მონაცემს ნაბიჯიდან ნაბიჯამდე) და ის ყოველთვის ირჩევს ერთსა და იმავე მოქმედებას ერთი და იმავე მონაცემებისთვის.*

იმპლემენტაციის დეტალები

რობოტის სტრატეგია უნდა იქნეს იმპლემეტირებული შემდეგი ფუნქციის სახით:

```
std::pair<int, int> navigate(int currColor, std::vector<int> adjColors)
```

პარამეტრებად იღებს მიმდინარე წვეროს ფერს და ყველა მეზობელი წვეროს ფერებს (თანმიმდევრობით). მან უნდა დააბრუნოს წყვილი, რომლის პირველი ელემენტია მიმდინარე წვეროს ახალი ფერი და მეორე ელემენტია იმ წვეროს ინდექსი, სადაც რობოტი უნდა გადავიდეს. თუ რობოტი შეწყვეტს მუშაობას, ფუნქციამ უნდა დააბრუნოს წყვილი $(-1, -1)$.

ეს ფუნქცია განმეორებითად გამოიძახება რობოტის მოქმედებების ასარჩევად. რადგან ის დეტერმინისტულია, თუ `navigate` უკვე გამოიძახეს გარკვეული პარამეტრებით, ხელმეორეად იმავე პარამეტრებით გამოძახების ნაცვლად, მისი წინა ანალოგიური გამოძახებიდან დაბრუნებული მნიშვნელობა ხელახლა იქნება გამოყენებული. გარდა ამისა, თითოეული ტესტი შეიძლება შეიცავდეს $T \leq 5$ ქვეტესტს (განსხვავებულ გრაფებს და/ან საწყის პოზიციებს) და შესაძლებელია მათი პარალელურად გაშვება (ანუ თქვენმა პროგრამამ შეიძლება მიიღოს გამოძახებები სხვადასხვა ქვეტესტების შესახებ მონაცვლეობით). და ბოლოს, `navigate`-ის გამოძახებები შეიძლება მოხდეს თქვენი პროგრამის **სხვადასხვა გაშვებისას** (ასევე შეიძლება მოხდეს ერთი და იგივე გაშვების დროსაც). თქვენი პროგრამის შესრულებების ჯამური რაოდენობაა $P = 100$. ამ ყველაფრის გამო, თქვენმა პროგრამამ არ უნდა სცადოს რაიმე ინფორმაციის გადაცემა სხვადასხვა გამოძახებებს შორის.

შეზღუდვები

- $3 \leq N \leq 1000$
- $0 \leq \text{Color} < 1500$
- $L = 3000$
- $T \leq 5$
- $P = 100$

შეფასება

ქვეამოცანის ქულების S წილი, რომელსაც თქვენ მიიღებთ, დამოკიდებულია C-ზე – თქვენს მიერ გამოყენებული განსხვავებული ფერების მაქსიმალურ რაოდენობაზე (ფერი 0-ის ჩათვლით) ამ ქვეამოცანის ნებისმიერ ტესტში ან ნებისმიერ სხვა სავალდებულო ქვეამოცანაში:

- თუ თქვენი ამოხსნა ჩაიჭრება რომელიმე ქვეტესტში, მაშინ $S = 0$.
- თუ $C \leq 4$, მაშინ $S = 1.0$.
- თუ $4 < C \leq 8$, მაშინ $S = 1.0 - 0.6\frac{C-4}{4}$.
- თუ $8 < C \leq 21$, მაშინ $S = 0.4\frac{8}{C}$.
- თუ $C > 21$, მაშინ $S = 0.15$.

ქვეამოცანები

ქვეამოცანა	ქულა	საჭირო ქვეამოცანა	N	დამატებითი შეზღუდვები
0	0	–	≤ 300	მაგალითი.
1	6	–	≤ 300	გრაფი არის ციკლი.[1]
2	7	–	≤ 300	გრაფი არის ვარსკვლავი.[2]
3	9	–	≤ 300	გრაფი არის ბილიკი.[3]
4	16	$2-3$	≤ 300	გრაფი არის ხე.[4]
5	27	–	≤ 300	თითოეულ წვეროს ყავს მაქსიმუმ 3 მეზობელი. წვეროს, რომელშიც რობოტი იწყებს, ჰყავს 1 მეზობელი წვერო.
6	28	$0-5$	≤ 300	–
7	7	$0-6$	–	–

[1]ციკლი გრაფის წიბოებია: $(i, (i+1) \bmod N)$ for $0 \leq i < N$.
[2]ვარსკვლავი გრაფის წიბოებია: $(0, i)$ for $1 \leq i < N$.
[3]ბილიკი გრაფის წიბოებია: $(i, i+1)$ for $0 \leq i < N-1$.
[4]ხე არის გრაფი ციკლების გარეშე.

მაგალითი

განვიხილოთ პირობაში მოცემულ სურათზე გამოსახული გრაფი, რომელსაც აქვს $N = 7$, $M = 8$ და წიბოები $(0,1)$, $(1,2)$, $(2,0)$, $(2,3)$, $(3,4)$, $(4,2)$, $(3,5)$ და $(2,6)$. გარდა ამისა, რადგან მეზობელი წვეროების სიებში ელემენტების თანმიმდევრობა მნიშვნელოვანია, მათ ამ ცხრილში წარმოგიდგენთ:

წვერო	მეზობელი წვეროები
0	$2, 1$
1	$2, 0$
2	$0, 3, 4, 6, 1$
3	$4, 5, 2$
4	$2, 3$
5	3
6	2

დავუშვათ რობოტი იწყებს წვერო 5-ში. შემდეგ მოცემული არის ინტერაქციების ერთ-ერთი შესაძლო (წარუმატებელი) მიმდევრობა:

#	ფერები	წვერო	გამოძახება	დაბრუნებული მნიშვნელობა
1	$0, 0, 0, 0, 0, 0, 0$	5	`navigate(0, {0})`	`{1, 0}`
2	$0, 0, 0, 0, 0, 1, 0$	3	`navigate(0, {0, 1, 0})`	`{4, 2}`
3	$0, 0, 0, 4, 0, 1, 0$	2	`navigate(0, {0, 4, 0, 0, 0})`	`{0, 3}`
4	$0, 0, 0, 4, 0, 1, 0$	6	[1]`navigate(0, {0})`	`{1, 0}`
5	$0, 0, 0, 4, 0, 1, 1$	2	`navigate(0, {0, 4, 0, 1, 0})`	`{8, 0}`
6	$0, 0, 8, 4, 0, 1, 1$	0	`navigate(0, {8, 0})`	`{3, 0}`
7	$3, 0, 8, 4, 0, 1, 1$	2	`navigate(8, {3, 4, 0, 1, 0})`	`{2, 2}`
8	$3, 0, 2, 4, 0, 1, 1$	4	`navigate(0, {2, 4})`	`{1, 1}`
9	$3, 0, 2, 4, 1, 1, 1$	3	`navigate(4, {1, 1, 2})`	`{-1, -1}`

აქ რობოტმა სულ 6 განსხვავებული ფერი გამოიყენა: 0, 1, 2, 3, 4 და 8 (შევნიშნოთ, რომ 0 გამოყენებული იქნებოდა მაშინაც კი, თუ რობოტი არასდროს დააბრუნებდა 0 ფერს, რადგან ყველა კვანძი იწყება 0 ფერით). რობოტმა 9 იტერაცია შეასრულა მუშაობდა დასრულებამდე. თუმცა, ის წარუმატებელი იყო, რადგან 1 წვეროზე ვიზიტის გარეშე შეწყვიტა დაასრულა მუშაობა.

[1]გაითვალისწინეთ, რომ `navigate`-ის გამოძახება იტერაცია 4-ზე რეალურად არ განხორციელდებოდა, რადგან ის იტერაცია 1 გამოძახების ეკვივალენტურია, ამიტომ გრეიდერი უბრალოდ ხელახლა გამოიყენებს თქვენი ფუნქციის დაბრუნებულ მნიშვნელობას ამ გამოძახებიდან. თუმცა, ეს მაინც რობოტის იტერაციად ითვლება.

სანიმუშო გრადერი

სანიმუშო გრადერი არ ასრულებს თქვენი პროგრამის მრავალჯერად გაშვებას, ამიტომ `navigate`-ის ყველა გამოძახება თქვენი პროგრამის ერთსა და იმავე გაშვებაში მოხდება.

შეყვანის ფორმატი შემდეგია: ჯერ იკითხება T (ქვეტესტების რაოდენობა). შემდეგ თითოეული ქვეტესტისთვის:

- სტრიქონი 1: ორი მთელი რიცხვი – N და M;
- სტრიქონი $2 + i$ (for $0 \leq i < M$): ორი მთელი რიცხვი – A_i და B_i, არის წვეროები, რომლებას წიბო i აკავშირებს ($0 \leq A_i, B_i < N$).

სანიმუშო გრადერი შემდეგ დაბეჭდავს თქვენს მიერ გამოყენებული განსხვავებული ფერების რაოდენობას და საჭირო იტერაციების რაოდენობას დასრულებამდე. ალტერნატიულად, თუ თქვენი ამოხსნა ჩაიჭრა, ის დაბეჭდავს შეცდომის შეტყობინებას.

თავისით, სანიმუშო გრადერი ბეჭდავს დეტალურ ინფორმაციას იმის შესახებ, თუ რას ხედავს და აკეთებს რობოტი თითოეული იტერაციის დროს. თქვენ შეგიძლიათ გამორთოთ ეს, DEBUG-ის მნიშვნელობის `true`-დან `false`-ზე შეცვლით.