მხედველობის პროგრამა

თქვენ უნდა მოახდინოთ მხედველობის პროგრამის იმპლემენტაცია რობოტისათვის. ყოველთვის, როდესაც რობოტის კამერა იღებს სურათს, ის შავ-თეთრი გამოსახულების სახით ინახება რობოტის მეხსიერებაში. ყოველი სურათი წარმოადგენს პიქსელების არეს $H \times W$ ზომით, სადაც სტრიქონები გადანომრილია 0-დან $(H-1)$-მდე და სვეტები გადანომრილია 0-დან $(W-1)$-მდე. ყოველ შენახულ გმოსახულებაში არის **ზუსტად ორი** შავი ფერის პიქსელი და ყველა დანარჩენი პიქსელი თეთრი ფერისაა.

რობოტს შეუძლია დაამუშაოს ყოველი გამოსახულება მარტივი ინსტრუქციების საშუალებით. თქვენ გეძლევათ H-ის და W-ს მნიშვნელობები და, ასევე, მთელი დადებითი რიცხვი K. თქვენი მიზანია დაწეროთ პროგრამა რობოტისათვის, რომელიც ნებისმიერ გამოსახულებაში დაადგენს, არის თუ არა **მანძილი** ორ შავ წერტილს შორის ზუსტად K. მანძილი r_1 სტრიქონსა და c_1 სვეტში მდებარე წერტილიდან r_2 სტრიქონსა და c_2 სვეტში მდებარე წერტილამდე ტოლია: $|r_1 - r_2| + |c_1 - c_2|$. ამ ფორმულაში $|x|$ აღნიშნავს x-ის აბსოლუტურ მნიშვნელობას, რაც ტოლია x-ის, თუ $x \geq 0$ და ტოლია $-x$-ის, თუ $x < 0$.

ახლა აღვწეროთ რობოტის მუშაობის პრინციპი.

რობოტის მეხსიერება წარმოადგენს უჯრედების საკმაოდ დიდ მასივს, რომელიც ინდექსირებულია 0-დან. ყოველი უჯრედი ინახავს 0-ს ან 1-ს და შენახვის შემდეგ მნიშვნელობა არ იცვლება. სურათი მეხსიერებაში ინახება, როგორც სტრიქონების მიმდევრობა. უჯრედები გადანომრილია სტრიქონების მიხედვით 0-დან $(H \cdot W - 1)$-მდე. პირველი სტრიქონი შენახულია უჯრედებში 0-დან $(W-1)$-მდე, ხოლო ბოლო სტრიქონი შენახულია უჯრედებში $(H-1)W$-დან $(H \cdot W - 1)$-მდე. კერძოდ, თუ პიქსელი i-ურ სტრიქონსა და j-ურ სვეტში არის შავი, მაშინ $i \cdot W + j$ უჯრედის მნიშვნელობა არის 1, წინააღმდეგ შემთხვევაში - 0.

რობოტის პროგრამა წარმოადგენს **ინსტრუქციების** მიმდევრობას, რომლებიც გადანომრილი თანმიმდევრული მთელი რიცხვებით 0-დან დაწყებული. პროგრამის შესრულებისას ინსტრუქციები სრულდება თანმიმდევრულად, ნომრების ზრდის მიხედვით.

ყოველი ინსტრუქცია კითხულობს ერთი ან რამდენიმე უჯრედის მნიშვნელობას (ამ მნიშვნელობებს ჩვენ ვუწოდებთ **შესატან მნიშვნელობებს**) და აბრუნებს ერთადერთ მნიშვნელობას 0 ან 1 (ამ მნიშვნელობებს ჩვენ ვუწოდებთ **გამოსატან მნიშვნელობებს**). i-ური ინსტრუქციის გამოსატანი მნიშვნელობა შენახულია $H \cdot W + i$ უჯრედში. i-ური ინსტრუქციის შესატანი მნიშვნელობები შეიძლება იყვნენ

მხოლოდ ის უჯრედები, რომლებიც შეიცავენ ან პიქსელებს, ან წინა ინსტრუქციების გამოსატან მნიშვნელობებს, ანუ უჯრედები 0-დან $(H \cdot W + i - 1)$-მდე.

არსებობს 4 ტიპის ინსტრუქცია:

- NOT: აქვს მხოლოდ ერთი შესატანი მნიშვნელობა. თუ შესატანი მნიშვნელობაა 0, მაშინ გამოსატანი მნიშვნელობა 1-ია, წინააღმდეგ შემთხვევაში გამოსატანი მნიშვნელობაა 0.
- AND: აქვს ერთი ან რამდენიმე შესატანი მნიშვნელობა. გამოსატანი მნიშვნელობა 1-ია მაშინ და მხოლოდ მაშინ, თუ **ყველა** შესატანი მნიშვნელობაა 1.
- OR: აქვს ერთი ან რამდენიმე შესატანი მნიშვნელობა. გამოსატანი მნიშვნელობა 1-ია მაშინ და მხოლოდ მაშინ, თუ **ერთი მაინც** შესატანი მნიშვნელობაა 1.
- XOR: აქვს ერთი ან რამდენიმე შესატანი მნიშვნელობა. გამოსატანი მნიშვნელობა 1-ია მაშინ და მხოლოდ მაშინ, თუ **კენტი რაოდენობის** შესატანი მნიშვნელობაა 1.

პროგრამის ბოლო ბრძანება უნდა იყოს 1-ის გამოტანა, თუ მანძილი ორ შავ პიქსელს შორის არის ზუსტად K და 0 - წინააღმდეგ შემთხვევაში.

იმპლემენტაციის დეტალები

თქვენ უნდა შეძლოთ ფუნქციის იმპლემენტაცია:

```
void construct_network(int H, int W, int K)
```

- H, W: კამერის მიერ გადაღებული გამოსახულების ზომები.
- K: დადებითი მთელი რიცხვი.
- ამ ფუნქციამ უნდა შეასრულოს რობოტის პროგრამა. რობოტის კამერით გადაღებული ყოველი სურათისათვის, პროგრამამ უნდა განსაზღვროს, არის თუ არა მანძილი ორ შავ პიქსელს შორის ზუსტად K.

ამ ფუნქციამ უნდა გამოიძახოს ქვემოთ მოყვანილი ერთი ან რამდენიმე ფუნქცია რობოტის პროგრამაში ინსტრუქციის ჩასამატებლად (რობოტის პროგრამა თავიდან ცარიელია).

```
int add_not(int N)
int add_and(int[] Ns)
int add_or(int[] Ns)
int add_xor(int[] Ns)
```

- ჩასვით NOT, AND, OR ან XOR ინსტრუქციები შესაბამისად.
- N (add_not-თვის): უჯრედის ინდექსი, რომლიდანაც NOT ინსტრუქცია კითხულობს შესატან მნიშვნელობას.
- Ns (add_and, add_or, add_xor-თვის): მასივი, რომელიც შეიცავს იმ უჯრედების ინდექსებს, საიდანაც AND, OR ან XOR ინსტრუქციები კითხულობენ შესატან

მნიშვნელობებს.
- ყოველი ფუნქცია აბრუნებს იმ უჯრედის ინდექსს, რომელშიც შენახული იქნა გამოსატანი მნიშვნელობები. ამ ფუნქციების თანმიმდევრული გამოძახება აბრუნებს თანმიმდევრულ მთელ რიცხვებს დაწყებული $H \cdot W$-დან.

რობოტის პროგრამა შეიძლება შეიცავდეს მაქსიმუმ 10 000 ინსტრუქციას. ინსტრუქციებს შეუძლიათ წაიკითხონ მაქსიმუმ 1 000 000 მნიშვნელობა ჯამურად. სხვა სიტყვებით, Ns მასივების ჯამური სიგრძე `add_and`, `add_or` და `add_xor` ინსტრუქციების ყველა გამოძახებაში პლიუს `add_not` ინსტრუქციის გამოძახებათა რაოდენობა, არ უნდა აღემატებოდეს 1 000 000.

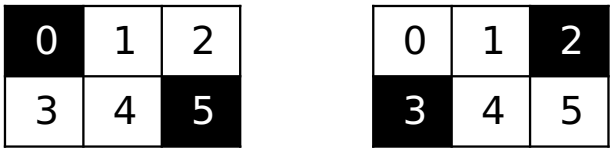
ბოლო ინსტრუქციის დამატების შემდეგ, უნდა დაბრუნდეს ფუნქცია `construct_network` . ამის შემდეგ რობოტის პროგრამა შემოწმდება სურათების გარკვეული რაოდენობისთვის. თქვენი პროგრამა გაივლის ამ ტესტირებას, თუ ამ სურათებიდან მხოლოდ მათთვის დააბრუნებს 1-ს, რომლებშიც ორ შავ პიქსელს შორის მანძილი ზუსტად K-ს ტოლია.

თქვენი ამოხსნის გრადერმა შეიძლება გამოიტანოს შეტყობინებები ინგლისურ ენაზე, რომლებიც ქვემოთაა ჩამოთვლი:

- `Instruction with no inputs`: შესატანი მონაცემები `add_and`, `add_or` და `add_xor` ინსტრუქციებისათვის წარმოადგენს ცარიელ მასივს.
- `Invalid index`: არაკორექტული (შესაძლოა უარყოფითი) ინდექსი `add_and`, `add_or`, `add_xor` ან `add_not` ინსტრუქციისათვის.
- `Too many instructions`: თქვენმა ფუნქციამ გამოიყენა 10 000-ზე მეტი ინსტრუქცია.
- `Too many inputs`: ინსტრუქციები კითხულობენ ჯამურად 1 000 000 მნიშვნელობაზე მეტს.

მაგალითი

ვთქვათ, $H = 2$, $W = 3$, $K = 3$. არსებობს მხოლოდ ორი ვარიანტი იმისა, რომ მანძილი ორ შავ პიქსელს შორის იყოს 3.



- შემთხვევა 1: შავი წერტილებია 0 და 5
- შემთხვევა 2: შავი წერტილებია 2 და 3

შესაძლო ამოხსნა გულისხმობს რობოტის პროგრამის აგებას შემდეგი ბრძანებებით:

1. `add_and([0, 5])`, ამატებს ინსტრუქციას, რომელსაც გამოაქვს 1 მაშინ და მხოლოდ მაშინ, როცა პირველ შემთხვევას აქვს ადგილი. გამოსატანი

მნიშვნელობა ინახება უჯრედში 6.
2. `add_and([2, 3])`, ამატებს ინსტრუქციას, რომელსაც გამოაქვს 1 მაშინ და მხოლოდ მაშინ, როცა მეორე შემთხვევას აქვს ადგილი. გამოსატანი მნიშვნელობა ინახება უჯრედში 7.
3. `add_or([6, 7])`, ამატებს ინსტრუქციას, რომელსაც გამოაქვს 1 მაშინ და მხოლოდ მაშინ, როცა ზემოთ ნაჩვენები ორი შემთხვევიდან ერთ-ერთს აქვს ადგილი.

შეზღუდვები

- $1 \leq H \leq 200$
- $1 \leq W \leq 200$
- $2 \leq H \cdot W$
- $1 \leq K \leq H + W - 2$

ქვეამოცანები

1. (10 ქულა) $\max(H, W) \leq 3$
2. (11ქულა) $\max(H, W) \leq 10$
3. (11 ქულა) $\max(H, W) \leq 30$
4. (15 ქულა) $\max(H, W) \leq 100$
5. (12 ქულა) $\min(H, W) = 1$
6. (8 ქულა) პიქსელი 0 სტრიქონსა და 0 სვეტში არის შავი ყველა სურათზე.
7. (14 ქულა) $K = 1$
8. (19 ქულა) დამატებითი შეზღუდვების გარეშე.

Sample grader

სანიმუშო გრადერი კითხულობს შესატან მონაცემებს შემდეგი ფორმატით:

- სტრიქონი 1: H W K
- სტრიქონი $2 + i$ ($i \geq 0$): $r_1[i]$ $c_1[i]$ $r_2[i]$ $c_2[i]$
- ბოლო სტრიქონი: -1

ყოველი სტრიქონი, გარდა პირველისა და ბოლოსი, წარმოადგენს გამოსახულებას ორი შავი პიქსელით. ჩვენ აღვნიშნავთ i-ურ სურათს სტრიქონზე ნომრით $2 + i$.

პირველი შავი პიქსელი არის $r_1[i]$ სტრიქონსა და $c_1[i]$ სვეტში, ხოლო მეორე - $r_2[i]$ სტრიქონსა და $c_2[i]$ სვეტში.

სანიმუშო გრადერი პირველად იძახებს `construct_network(H, W, K)`. თუ `construct_network` არღვევს რომელიმე შეზღუდვას ამოცანის პირობიდან, მაშინ გრადერს გამოაქვს "იმპლემენტაციის დეტალებში" ჩამოთვლილი შეტყობინებებიდან ერთ-ერთი და ამთავრებს მუშაობას. სხვა შემთხვევაში, გრადერი ასრულებს ორნაირ

გამოტანას.

პირველი: გრადერს გამოაქვს რობოტის პროგრამაში გამოსატანი მონაცემები შემდეგი ფორმატით:

- სტრიქონი $1 + i$ $(0 \le i)$: ბოლო ინსტრუქციის გამოსატანი მნიშვნელობა i-ური სურათისათვის (1 ან 0).

მეორე: მიმდინარე ფოლდერში გრადერი წერს `log.txt` ფაილს შემდეგი ფორმატით:

- სტრიქონი $1 + i$ $(0 \le i)$: $m[i][0]$ $m[i][1]$ $\ldots$ $m[i][c-1]$

მიმდევრობა $(1 + i)$-ე სტრიქონში აღწერს რობოტის მეხსიერების უჯრედებში შენახულ მნიშვნელობებს რობოტის პროგრამის ამუშავების და შესატან მონაცემებად i-ური სურათის განხილვის შემდეგ. კერძოდ, $m[i][j]$ იძლევა j-ური უჯრედის მნიშვნელობას. მიაქციეთ ყურადღება, რომ c-ს მნიშვნელობა (მიმდევრობის სიგრძე) ტოლია $H \cdot W$-ს პლიუს ინსტრუქციების რაოდენობა რობოტის პროგრამაში.