მარსი

ცნობილია, რომ ფარაონები პირველები გავიდნენ ღია კოსმოსში. მათ გაუშვეს პირველი კოსმოსური ხომალდი, რომელიც დაჯდა პლანეტა თუტმუსზე (დღეს ცნობილია მარსის სახელით). პლანეტის ზედაპირი შეიძლება წარმოვიდგინოთ როგორც $(2n+1) \times (2n+1)$ ზომის არე, რომელიც დაყოფილია კვადრატულ უჯრედებად. თითოეული უჯრედი შეიცავს ან მიწას (ხმელეთს), ან წყალს. იმ უჯრედის მდგომარეობა, რომელიც განლაგებულია i-ურ სტრიქონსა და j-ურ სვეტში ($0 \leq i, j \leq 2 \cdot n$) აღნიშნულია როგორც $s[i][j] =$'`1`', თუ ის შეიცავს მიწას, და $s[i][j] =$'`0`', თუკი ის შეიცავს წყალს.

ხმელეთის ორ უჯრედს უწოდებენ ბმულს, თუკი მათ შორის არსებობს მხოლოდ ხმელეთის უჯრედებით შედგენილი გზა. ამასთან, ყოველ ორ მომდევნო უჯრედს ამ გზაზე უნდა ჰქონდეს საერთო გვერდი. კუნძული პლანეტაზე განისაზღვრება, როგორც ხმელეთის უჯრედების მაქსიმალური სიმრავლე, რომლის ნებისმიერი ორი უჯრედი ბმულია.

კოსმოსური ხომალდის ამოცანა იმაში მდგომარეობს, რომ დაითვალოს კუნძულების რაოდენობა პლანეტაზე. თუმცა ხომალდზე არსებული ძველი კომპიუტერისათვის ეს ამოცანა არც ისე მარტივია. კომპიუტერს აქვს h ტევადობის მეხსიერება, რომელშიც ინახება $(2n+1) \times (2n+1)$ ზომის ორგანზომილებიანი მასივის მონაცემები. მასივის თითეულ ელემენტს შეუძლია შეინახოს 100 სიგრძის ორობითი სტრინგი, რომლის თითოეული წევრი არის '`0`' (ASCII 48) ან '`1`' (ASCII 49). თავდაპირველად, ყოველი უჯრის პირველ ბიტში არის შენახული ამ უჯრის მდგომარეობა, ანუ, $h[i][j][0] = s[i][j]$ ($0 \leq i, j \leq 2 \cdot n$). h-ს ყველა სხვა ბიტი თავიდან არის '`0`' (ASCII 48).

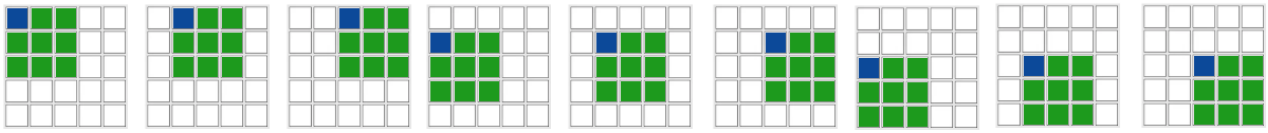
მეხსიერებაში არსებული მონაცემების დასამუშავებლად თითოეულ მომენტში კომპიუტერს წვდომა აქვს მხოლოდ 3×3 პორციაზე და ინფორმაციის შეცვლა შეუძლია მხოლოდ ამ პორციის ზედა მარცხენა კუთხეში. ფორმალურად, კომპიუტერს აქვს წვდომა $h[i..i+2][j..j+2]$ ($0 \leq i, j \leq 2 \cdot (n-1)$) მნიშვნელობებზე და შეუძლია შეცვალოს $h[i][j]$ მნიშვნელობა. ამ პროცესს მოვიხსენიებთ, როგორც (i, j) **უჯრის დამუშავებას**.

კომპიუტერის შეზღუდული მონაცემების გამო ფარაონებმა შემდეგი მექანიზმი შეიმუშავეს:

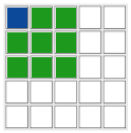
- კომპიუტერი მეხსიერებას დაამუშავებს n ფაზის განმავლობაში.
- ფაზა k-სთვის ($0 \leq k \leq n-1$), ავიღოთ $m = 2 \cdot (n-k-1)$, კომპიუტერი დაამუშავებს (i, j) უჯრებს თითოეული $0 \leq i, j \leq m$-სთვის, i-ს ზრდადობის მიხედვით, და თითეული i-სთვის, j-ს ზრდადობის მიხედვით. სხვა სიტყვებით, კომპიუტერი დაამუშავებს უჯრებს შემდეგი თანმიმდევრობით: $(0,0), (0,1), \cdots, (0,m)$, $(1,0), (1,1), \cdots, (1,m), \cdots, (m,0), (m,1), \cdots, (m,m)$.
- ბოლო ფაზაში ($k = n-1$), კომპიუტერი დაამუშავებს მხოლოდ $(0,0)$ უჯრას. რომლის შემდეგაც, $h[0][0]$-ში ჩაწერილი რიცხვი უნდა იყოს პლანეტაზე კუნძლების რაოდენობის ორობითი ჩანაწერი, სადაც სტრინგის პირველი სიმბოლო აღნიშნავს ყველაზე პატარა ბიტს.

მოცემული დიაგრამა აჩვენებს როგორ ამუშავებს კომპიუტერი 5×5 ($n = 2$) ზომის მეხსიერებას. ლურჯი უჯრა გამოხატავს უჯრას, რომელშიც ინფორმაცია იცვლება და მწვანე უჯრა გამოხატავს პორციას რომლის დამუშავებაც ხდება.

ფაზა 0-ს დროს, კომპიუტერი ამუშავებს პორციებს შემდეგი თანმიმდევრობით:



ფაზა 1-ს დროს, კომპიუტერი ამუშავებს მხოლოდ ერთ პორციას:



თქვენი ამოცანაა გაუკეთოთ იმპლემენტაცია მეთოდს, რომელიც საშუალებას მისცემს კომპიუტერს, რომ დაითვალოს კუნძულების რაოდენობა პლანეტა თუტმუსზე მოცემული მექანიზმის მიხედვით.

იმპლემენტაციის დეტალები

თქვენ უნდა გააკეთოთ შემდეგი პროცედურის იმპლემენტაცია

```
string process(string[][] a, int i, int j, int k, int n)
```

- a: a 3×3 მასივი, რომელიც აღწერს ქვემასივს, რომლის დამუშავებაც ახლა მიმდინარეობს, კონკრეტულად, $a = h[i..i+2][j..j+2]$, სადაც a-ს თითოეული ელემენტი არის ზუსტად 100 სიგრძის მქონე სტრინგი და მასში თითოეული სიმბოლო არის `'0'` (ASCII 48) ან `'1'` (ASCII 49).
- i, j: სტრიქონის და სვეტის ნომრები იმ უჯრის, რომლის დამუშავებაც მიმდინარეობს.
- k: მიმდინარე ფაზის ნომერი.
- n: სულ ფაზების რაოდენობა და პლანეტის ზომა.პლანეტა შეიცავს $(2n+1) \times (2n+1)$ უჯრას.
- პროცედურამ უნდა დააბრუნოს 100სიგრძის ორობითი სტრინგი. დაბრუნებული მნიშვნელობა ჩაიწერება კომპიუტერის მეხსიერებაში $h[i][j]$ ადგილზე.
- პროცედურის ბოლო გამოძახება მოხდება როცა $k = n - 1$. ამ გამოძახების დროს პროცედურამ უნდა დააბრუნოს პლანეტაზე კუნძულების რაოდენობის ორობითი ჩანაწერი, სადაც ყველაზე პატარა ბიტი იქნება ჩაწერილი ინდექს 0-ზე (სტრინგის პირველი სიმბოლო), მეორე ყველაზე პატარა ბიტი იქნება ინდექს 1-ზე და ასე შემდეგ.
- ეს პროცედურა დამოუკიდებელი უნდა იყოს ნებისმიერი სტატიკური ან გლობალური ცვლადებისგან და მისი დაბრუნებული მნიშვნელობა დამოკიდებული უნდა იყოს მხოლოდ მისთვის მიწოდებულ პარამეტრებზე.

თითოეული ტესტი შეიცავს T ცალ განსხვავებული სიტუაციას (ე.ი., სხვადასხვა პლანეტების ზედაპირებს). თქვენი იმპლემენტაციის ქცევა დამოუკიდებელი უნდა იყოს სიტუაციების თანმიმდევრობისგან, რადგან `process` პროცედურის გამოძახებები შეიძლება არ მოხდეს

თანმიმდევრობით. გარანტირებულია, რომ თითოეული სიტუაციისთვის `process` იქნება გამოძახებული იმ თანმიმდევრობის მიხედვით, რომელიც პირობაშია აღწერილი.

დამატებით, თითოეული ტესტისთვის თქვენი პროგრამის რამდენიმე ვერსია პარალელურად იქნება გაშვებული. მეხსიერების და დროის ლიმიტები არის ამ ყველა ვერსიებითვის ერთად. ნებისმიერი მცდელობა მონაცემების გატანის საზღვრებს გარეთ ამ პროცესებს შორის ჩაითვლება თაღლითობად და გამოიწვევს დისკვალიფიკაციას.

კონკრეტულად, ნებისმიერი ინფორმაცია შენახული სტატიკურ ან გლობალურ ცვლადებად `process` პროცედურიდან, არ არის გარანტირებული, რომ დაგხვდებათ შემდეგი გამოძახებისას.

შეზღუდვები

- $1 \leq T \leq 10$
- $1 \leq n \leq 20$
- $0 \leq s[i][j] \leq 1$ (ყველა $0 \leq i, j \leq 2 \cdot n$)
- $h[i][j]$-ის სიგრძე არის ზუსტად 100 (ყველა $0 \leq i, j \leq 2 \cdot n$)
- $s[i][j]$ is either '`0`'(ASCII 48) or '`1`'(ASCII 49) (ყველა $0 \leq i, j \leq 2 \cdot n$)

`process` პროცედურის ყოველი გამოძახებისთვის:

- $0 \leq k \leq n - 1$
- $0 \leq i, j \leq 2 \cdot (n - k - 1)$

ქვეამოცანები

1. (6 ქულა) $n \leq 2$
2. (8 ქულა) $n \leq 4$
3. (7 ქულა) $n \leq 6$
4. (8 ქულა) $n \leq 8$
5. (7 ქულა) $n \leq 10$
6. (8 ქულა) $n \leq 12$
7. (10 ქულა) $n \leq 14$
8. (24 ქულა) $n \leq 16$
9. (11 ქულა) $n \leq 18$
10. (11 ქულა) $n \leq 20$

მაგალითები

მაგალითი 1

განვიხილოთ შემთხვევა, როცა $n = 1$ და s-ს აქვს სახე:

```
'1' '0' '0'
'1' '1' '0'
'0' '0' '1'
```

ამ მაგალითში პლანეტის ზედაპირი შედგება 3×3 უჯრედის და 2 კუნძულისაგან. მოხდება `process` პროცედურის გამოძახების მხოლოდ 1 ფაზა.

0 ფაზის დროს გრადერი გამოიძახებს `process` პროცედურას ზუსტად ერთხელ:

```
process([["100","000","000"],["100","100","000"],["000","000","100"]],0,0,0,1)
```

მიაქციეთ ყურადღება, რომ ნაჩვენებია h-ის ყოველი უჯრედის მხოლოდ პირველი 3 ბიტი.

პროცედურა დააბრუნებს `"0100..."` (ყველა გამოტოვებული ბიტი 0-ია). სადაც ....0010 ორობითში ტოლია 2-ის ათობითში. 96 ნულიანიგამოტოვებულია დაშეცვლილია `...`-ით.

მაგალითი 2

განვიხილოთ შემთხვევა, როცა $n = 2$ და s-ს აქვს სახე:

```
'1' '1' '0' '1' '1'
'1' '1' '0' '0' '0'
'1' '0' '1' '1' '1'
'0' '1' '0' '0' '0'
'0' '1' '1' '1' '1'
```

ამ მაგალითში პლანეტის ზედაპირი შეიცავს 5×5 უჯრედს და 4 კუნძულს. გვექნება `process` პროცედურის გამოძახების 2 ფაზა.

0 ფაზის დროს გრადერი გამოიძახებს `process` პროცედურას 9-ჯერ:

```
process([["100","100","000"],["100","100","000"],["100","000","100"]],0,0,0,2)
process([["100","000","100"],["100","000","000"],["000","100","100"]],0,1,0,2)
process([["000","100","100"],["000","000","000"],["100","100","100"]],0,2,0,2)
process([["100","100","000"],["100","000","100"],["000","100","000"]],1,0,0,2)
process([["100","000","000"],["000","100","100"],["100","000","000"]],1,1,0,2)
process([["000","000","000"],["100","100","100"],["000","000","000"]],1,2,0,2)
process([["100","000","100"],["000","100","000"],["000","100","100"]],2,0,0,2)
process([["000","100","100"],["100","000","000"],["100","100","100"]],2,1,0,2)
process([["100","100","100"],["000","000","000"],["100","100","100"]],2,2,0,2)
```

დავუშვათ, ზემოთ ნაჩვენებმა გამოძახებებმა დააბრუნეს მნიშვნელობები: `"011", "000", "000", "111", "111", "011", "110", "010", "111"` შესაბამისად, სადაც გამოტოვებული ბიტები წარმოადგენენ ნულებს. ამრიგად, 0 ფაზის დასრულების შემდეგ, h-ში შენახული იქნება შემდეგი მნიშვნელობები:

```
"011", "000", "000", "100", "100"
"111", "111", "011", "000", "000"
"110", "010", "111", "100", "100"
"000", "100", "000", "000", "000"
"000", "100", "100", "100", "100"
```

1 ფაზის დროს გრადერი გამოიძახებს `process` პროცედურას ერთხელ:

```
process([["011","000","000"],["111","111","011]",["110","010","111"]],0,0,1,2)
```

საბოლოოდ პროცედურა დააბრუნებს `"0010000...."` (გამოტოვებული ბიტები წარმოადგენენ ნულებს), სადაც ....0000100 ორობითში ტოლია 4-ის ათობითში. შევნიშნოთ, რომ 93 ნული შეცვლილია `...`-ით.

სანიმუშო გრადერი

სანიმუშო გრადერი კითხულობს შემოსატან მონაცემებს შემდეგი ფორმატით:

- სტრიქონი 1: T
- ბლოკი i ($0 \leq i \leq T-1$): ბლოკი წარმოადგენს i-ურ სცენარს.
  - სტრიქონი 1: n
  - სტრიქონი $2+j$ ($0 \leq j \leq 2 \cdot n$): $s[j][0]$ $s[j][1]$ ... $s[j][2 \cdot n]$

სანიმუშო გრადერი ბეჭდავს გამოსატან მონაცემებს შემდეგი ფორმატით:

- სტრიქონი $1+i$ ($0 \leq i \leq T-1$): უკანასკნელი დასაბრუნებელი მნიშვნელობა `process` პროცედურიდან i-ური სცენარისთვის ათობითში.