Mars

Như đã biết, các Phraoh là những người đầu tiên khám phá vũ trụ. Họ đã phóng con tàu vũ trụ đầu tiên lên hành tinh Thutmus I (ngày nay được biết đến với tên là Sao Hỏa). Bề mặt của hành tinh có thể được mô phỏng bởi một lưới $(2n+1) \times (2n+1)$ các ô vuông với mỗi ô chứa đất hoặc nước. Trạng thái của ô ở hàng i và cột j $(0 \leq i, j \leq 2 \cdot n)$ được kí hiệu bằng $s[i][j] =$ '`1`' nếu nó chứa đất, và $s[i][j] =$ '`0`' nếu nó chứa nước.

Hai ô đất được nói là kết nối với nhau nếu giữa chúng có một đường đi chứa các ô đất trong đó hai ô liên tiếp nhau thì có chung một cạnh. Một hòn đảo trên hành tinh được định nghĩa là một tập tối đa các ô đất sao cho hai ô bất kì trên đảo được kết nối với nhau.

Nhiệm vụ của tàu vũ trụ là đếm số hòn đảo trên hành tinh. Tuy nhiên, nhiệm vụ này không dễ dàng bởi vì tàu vũ trụ sử dụng máy tính cổ đại. Máy tính có một bộ nhớ h để chứa dữ liệu dưới dạng một mảng hai chiều kích thước $(2n+1) \times (2n+1)$, trong đó mỗi phần tử của mảng có thể chứa một xâu nhị phân có độ dài 100 trong đó mỗi kí tự là '0' (ASCII 48) hoặc '1' (ASCII 49). Khởi tạo ban đầu, kí tự đầu tiên của mỗi ô nhớ chứa trạng thái của mỗi ô trong lưới, $h[i][j][0] = s[i][j]$ (với mọi $0 \leq i, j \leq 2 \cdot n$). Tất cả các kí tự khác của h được khởi tạo bằng '0' (ASCII 48).

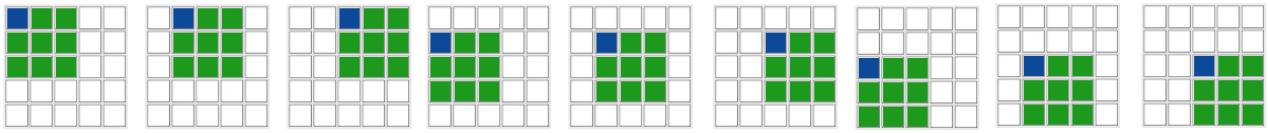
Để xử lý dữ liệu lưu trong bộ nhớ, máy tính chỉ có thể truy cập một vùng nhỏ kích thước 3×3 của bộ nhớ và ghi đè giá trị vào ô trên-trái của vùng đó. Một cách hình thức hơn, máy tính có thể truy cập các giá trị tại các ô $h[i..i+2][j..j+2]$ $(0 \leq i, j \leq 2 \cdot (n-1))$ và ghi đè giá trị tại ô $h[i][j]$. Quá trình này sẽ được gọi là **xử lý ô** (i, j).

Để giải quyết giới hạn của máy tính, các Pharaoh sử dụng một cơ chế như sau:

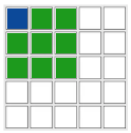
- Máy tính sẽ xử lý bộ nhớ thông qua n bước.
- Ở bước k $(0 \leq k \leq n-1)$, đặt $m = 2 \cdot (n-k-1)$, máy tính sẽ xử lý ô (i, j) với mọi $0 \leq i, j \leq m$, theo thứ tự tăng dần của i, và với mỗi i theo thứ tự tăng dần của j. Nói một cách khác, máy tính sẽ xử lý các ô theo thứ tự sau: $(0,0), (0,1), \cdots, (0,m)$, $(1,0), (1,1), \cdots, (1,m), \cdots, (m,0), (m,1), \cdots, (m,m)$.
- Ở bước cuối ($k = n-1$), máy tính chỉ xử lý ô $(0,0)$. Sau đó, giá trị ghi tại ô $h[0][0]$ sẽ bằng số lượng hòn đảo trên hành tinh dưới dạng nhị phân trong đó bit có trọng số nhỏ nhất trong số được lưu ở kí tự đầu tiên của xâu.

Biểu đồ bên dưới mô tả cách máy tính xử lý một bộ nhớ kích thước 5×5 ($n = 2$). Ô màu xanh nước biển thể hiện các ô đang được ghi đè giá trị vào, và các ô được tô màu thể hiện vùng đang được xử lý.

Trong bước 0, máy tính sẽ xử lý các vùng theo thứ tự sau:



Trong bước 1, máy tính sẽ xử lý duy nhất một vùng:



Nhiệm vụ của bạn là cài đặt một phương pháp để cho phép máy tính đếm số lượng đảo trên hành tinh Thutmus I theo cách máy tính hoạt động.

Chi tiết cài đặt

Bạn cần cài đặt một hàm sau:

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

- a: một mảng 3×3 biểu diễn một vùng đang được xử lý, cụ thể là $a = h[i..i+2][j..j+2]$, trong đó mỗi phần tử của a là một xâu có độ dài bằng đúng 100 và mỗi kí tự sẽ bằng `'0'` (ASCII 48) hoặc `'1'` (ASCII 49).
- i, j: dòng và cột của ô mà máy tính đang xử lý.
- k: bước hiện tại.
- n: số lượng bước, và kích thước của bề mặt hành tinh chứa $(2n+1) \times (2n+1)$ ô.
- Hàm này cần trả lại một xâu nhị phân có độ dài 100. Giá trị trả về sẽ được lưu trong bộ nhớ máy tính tại ô $h[i][j]$.
- Lần gọi cuối của hàm này sẽ xuất hiện khi $k = n - 1$. Trong lần gọi này, hàm cần trả về số lượng đảo trên hành tinh dưới dạng biểu diễn nhị phân trong đó bit có trọng số nhỏ nhất được lưu tại vị trí 0 (kí tự đầu tiên của xâu) và bit có trọng số nhỏ thứ nhì được lưu tại vị trí 1 và tiếp tục như vậy.
- Hàm này bắt buộc phải độc lập với bất cứ biến tĩnh hay biến toàn cục nào, và giá trị nó trả về chỉ nên phụ thuộc vào các tham số được truyền cho nó.

Mỗi test liên quan đến T kịch bản độc lập (tức là các bề mặt khác nhau của hành tinh). Hoạt động của hàm do bạn cài đặt cho mỗi kịch bản bắt buộc phải độc lập với thứ tự các kịch bản, bởi vì các lời gọi đến hàm `process` cho mỗi kịch bản có thể không xảy ra liền nhau. Tuy nhiên, nó được đảm bảo rằng với mỗi kịch bản, các lời gọi `process` xuất hiện theo một trình tự đã được xác định trong đề bài.

Thêm nữa, với mỗi test, phiên bản chương trình của bạn có thể được gọi thực hiện một số lần cùng lúc. Các giới hạn về bộ nhớ và thời gian thực hiện là cho tất cả các phiên bản chương trình gộp lại. Bất cứ việc thử có chủ ý để truyền dữ liệu giữa các phiên bản chương trình sẽ được coi là gian lận và sẽ dẫn đến việc bị loại khỏi kì thi.

Cụ thể là, bất cứ thông tin nào được lưu trong biến tĩnh hay biến toàn cục trong quá trình gọi một hàm `process` sẽ không được bảo tồn trong các lần gọi hàm tiếp theo.

Các rằng buộc

- $1 \leq T \leq 10$
- $1 \leq n \leq 20$
- $s[i][j]$ bằng '`0`'(ASCII 48) hoặc '`1`'(ASCII 49) (với mọi $0 \leq i, j \leq 2 \cdot n$)
- Độ dài của $h[i][j]$ bằng đúng 100 (với mọi $0 \leq i, j \leq 2 \cdot n$)
- Mỗi kí tự của $h[i][j]$ bằng `'0'` (ASCII 48) hoặc `'1'` (ASCII 49) (với mọi $0 \leq i, j \leq 2 \cdot n$)

Với mỗi lời gọi hàm `process` :

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

Subtask

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

Các ví dụ

Ví dụ 1

Xét trường hợp $n = 1$ và s như sau:

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

Trong ví dụ này, bề mặt hành tinh chứa 3×3 ô và 2 hòn đảo. Ở đây chỉ có một bước gọi hàm `process`.

Trong bước 0, trình chấm sẽ gọi hàm `process` đúng 1 lần:

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

Lưu ý rằng chỉ có 3 bit đầu tiên của mỗi ô nhớ của h được hiện ra.

Hàm này cần trả về `"0100..."` (các bit bị bỏ đi tất cả là 0), trong đó $....0010$ dưới dạng nhị phân tương đương 2 dưới dạng thập phân. Lưu ý ở đây có 96 số 0 bị bỏ đi và được thay thế bằng `....`.

Ví dụ 2

Xét trường hợp $n = 2$ và s như sau:

```
'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'
```

Trong ví dụ này bề mặt hành tinh chứa 5×5 ô và 4 hòn đảo. Sẽ có 2 bước gọi hàm `process` .

Trong bước 0, trình chấm sẽ gọi hàm `process` 9 lần:

```
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)
```

Giả sử các lời gọi hàm trên trả lại các giá trị tương ứng là `"011", "000", "000", "111", "111", "011", "110", "010", "111"` trong đó các bit bị bỏ đi có giá trị bằng 0. Do đó sau khi bước 0 kết thúc, h sẽ chứa các giá trị sau:

```
"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"
```

Trong bước 1, trình chấm sẽ gọi hàm `process` một lần:

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

Cuối cùng, hàm này cần trả về `"0010000...."` (các bit bị bỏ đi tất cả đều là 0), trong đó ....0000100 ở dạng nhị phân tương đương 4 ở dạng thập phân. Lưu ý có 93 số 0 bị bỏ đi và được thay thế bằng `....`

Trình chấm mẫu

Trình chấm mẫu đọc dữ liệu đầu vào theo khuôn dạng sau:

- dòng 1: T
- khối i ($0 \le i \le T - 1$): một khối biểu diễn kịch bản thứ i.
    - dòng 1: n
    - dòng $2 + j$ ($0 \le j \le 2 \cdot n$): $s[j][0]$ $s[j][1]$ $\ldots$ $s[j][2 \cdot n]$

Trình chấm mẫu in kết quả ra theo cấu trúc sau:

- dòng $1 + i$ ($0 \le i \le T - 1$): giá trị cuối cùng trả về của hàm `process` cho kịch bản thứ i dưới dạng thập phân.