Máy đo dùng sỏi

Leonardo phát minh ra "máy đo" đầu tiên: một cái xe có thể đo các khoảng cách bằng cách thả các viên sỏi mỗi khi bánh xe lăn. Số lần bánh xe lăn có thể được tính bằng cách đếm số sỏi, và cho phép người dùng tính khoảng cách mà máy đo đã đi qua. Với vai trò là nhà chuyên gia máy tính, chúng ta đã thêm vào máy đo phần mềm quản lý để mở rộng tính năng của máy đo. Nhiệm vụ của bạn là điều khiển máy đo dựa vào các quy tắc được nêu cụ thể dưới đây.

Lưới hoạt động

Máy đo di chuyển trong một lưới ô vuông tưởng tượng có kích thước 256 x 256 ô đơn vị. Mỗi ô có thể có nhiều nhất 15 viên sỏi và được xác định bởi 1 cặp tọa độ (hàng, cột), mà mỗi tọa độ nằm trong khoảng 0, ..., 255. Với một ô (i, j), ô cạnh nó (nếu có) là (i-1, j), (i+1, j), (i, j-1) và (i, j+1). Các ô nằm ở hàng đầu hoặc hàng cuối hoặc cột đầu hoặc cột cuối được gọi là "biên". Máy đo luôn bắt đầu ở ô (0,0) (góc tây bắc), nhìn hướng bắc.

Các lệnh cơ bản

Máy đo có thể được lập trình điều khiển sử dụng các lệnh sau.

`left` - quay 90 độ sang trái (ngược chiều kim đồng hồ) và vẫn ở nguyên ô hiện tại (ví dụ nếu máy đo đang nhìn về hướng nam thì nó sẽ chuyển sang nhìn hướng đông sau lệnh này).

`right` - quay 90 độ sang phải (theo chiều kim đồng hồ)và vẫn ở nguyên ô hiện tại (ví dụ nếu máy đo đang nhìn về hướng tây thì nó sẽ chuyển sang nhìn hướng bắc sau lệnh này).

`move` - di chuyển 1 ô về phía trước (theo hướng mà máy đo đang nhìn) sang ô bên cạnh. Nếu không tồn tại ô bên cạnh (đã ở ô biên theo hướng đang nhìn) thì lệnh này không có tác dụng.

`get` - loại bỏ một viên sỏi khỏi ô hiện tại. Nếu ô hiện tại không có sỏi thì lệnh này không có tác dụng.

`put` - thêm một viên sỏi vào ô hiện tại. Nếu ô hiện tại đã có 15 viên sỏi thì lệnh này không có tác dụng. Máy đo không bao giờ hết sỏi.

`halt` - kết thúc việc thực hiện chương trình.

Máy đo thực hiện các lệnh theo thứ tự mà chúng được cho trong chương trình. Chương trình chứa nhiều nhất một câu lệnh trên mỗi dòng. Các dòng trống được bỏ qua. Ký tự # đánh dấu phần chú thích; bất kỳ phần chữ nào đi sau ký tự này cho đến cuối dòng sẽ được bỏ qua. Khi máy đo đạt đến cuối chương trình, việc thực hiện được kết thúc.

Ví dụ 1

Xét chương trình sau cho máy đo. Chương trình sẽ đưa máy đo đến ô (0,2) nhìn hướng đông. (Lưu ý lệnh `move` đầu tiên được bỏ qua vì máy đo đang ở góc tây bắc nhìn hướng bắc.)

```
move  # không tác dụng
right
# bây giờ máy đo đang nhìn hướng đông
move
move
```

Nhãn, Biên và Sỏi

Để thay đổi cách chương trình chạy dựa vào trạng thái hiện tại, bạn có thể sử dụng các nhãn. Mỗi nhãn là một xâu ký tự có phân biệt chữ hoa chữ thường gồm không quá 128 ký tự được chọn từ `a`, …, `z`, `A`, …, `Z`, `0`, …, `9`. Các câu lệnh mới liên quan đến nhãn được liệt kê dưới đây. Trong mô tả dưới đây, *`L`* được sử dụng cho một nhãn hợp lệ.

`L``:` (nghĩa là *`L`* theo sau là 1 dấu hai chấm '`:`') — Khai báo vị trí của nhãn *`L`* trong chương trình. Tất cả các nhãn được khai báo phải là duy nhất. Việc khai báo một nhãn không ảnh hưởng đến máy đo.

- `jump` *`L`* — tiếp tục chạy chương trình bằng cách nhảy vô điều kiện đến dòng có nhãn L.
- `border` *`L`* — tiếp tục chạy chương trình bằng cách nhảy đến dòng có nhãn *`L`* nếu máy đo đang ở biên nhìn ra cạnh của lưới (nghĩa là lệnh move sẽ không có tác dụng); nếu không thỏa mãn điều kiện chương trình sẽ tiếp tục bình thường và lệnh này không có tác dụng
- `pebble` *`L`* — tiếp tục chạy chương trình bằng cách nhảy đến dòng có nhãn *`L`* nếu ô hiện tại có ít nhất 1 viên sỏi; trái lại chương trình sẽ tiếp tục bình thường và lệnh này không có tác dụng.

Ví dụ 2

Chương trình sau đây tìm viên sỏi đầu tiên (nằm lệch về phía tây nhất) ở hàng 0 và dừng ở đó; nếu không có sỏi ở hàng 0, máy dừng ở biên, cuối hàng. Chương trình sử dụng 2 nhãn `leonardo` và `davinci`.

```
right
leonardo:
pebble davinci  # Tìm thấy sỏi
border davinci  # cuối dòng
move
jump leonardo
davinci:
halt
```

Máy đo khởi đầu quay sang phải. Vòng lặp bắt đầu bằng khai báo nhãn `leonardo:` và kết thúc bằng lệnh `jump leonardo`. Trong vòng lặp, máy đo kiểm tra có sỏi ở ô hiện tại không hoặc đã ở ô biên cuối hàng không; nếu không phải máy đó tiến hành lệnh `move` để chuyển từ ô hiện tại (0,j) sang ô bên cạnh (0, j+1) vì ô bên cạnh tồn tại. (Lệnh `halt` không nhất thiết cần ở đây vì chương trình đằng nào cũng kết thúc.)

Phát biểu bài toán

Bạn cần nộp 1 chương trình sử dụng ngôn ngữ của máy đo như mô tả ở trên để máy đo hành động theo yêu cầu. Mỗi subtask (xem dưới đây) mô tả hành vi của máy đo cần thực hiện và các ràng buộc mà chương trình nộp cần thỏa mãn. Các ràng buộc liên quan đến hai vấn đề sau:

Kích cỡ của chương trình — Chương trình phải đủ ngắn. Kích cỡ của chương trình là số câu lệnh trong chương trình. Các câu khai báo nhãn, chú thích và dòng trống *không tính* vào kích cỡ của chương trình.

Độ dài thực thi - Chương trình phải kết thúc đủ nhanh. Độ dài thực thi là *số bước* được thực hiện: mỗi câu lệnh được thực hiện được tính là 1 bước, bất kể câu lệnh có tác dụng hay không; khai báo nhãn, chú thích và dòng trống không tính là một bước.

Trong Ví dụ 1, kích cỡ của chương trình là 4 và độ dài thực thi là 4. Trong Ví dụ 2, kích cỡ của chương trình là 6 và khi thực thi trên lưới với 1 viên sỏi ở ô (0, 10) thì độ dài thực thi là 43 bước: `right`, 10 lần chạy vòng lặp mỗi lần gồm 4 bước (`pebble davinci`; `border davinci`; `move`; `jump leonardo`) và cuối cùng là `pebble davinci` và `halt`.

Subtask 1 [9 points]

Khởi đầu có x viên sỏi ở ô (0,0) và y viên sỏi ở ô (0,1) và tất cả các ô khác đều không có sỏi. Lưu ý tại mỗi ô chỉ có tối đa 15 viên sỏi. Hãy viết một chương trình để máy đo kết thúc ở ô (0,0) nếu x ≤ y và ở ô (0,1) trong trường hợp ngược lại. (Chúng ta không quan tâm đến hướng máy đo nhìn lúc cuối; chúng ta cũng không quan tâm có bao nhiên viên sỏi ở trạng thái cuối cũng như các viên sỏi này nằm ở đâu.)

Giới hạn: kích cỡ chương trình ≤ 100, độ dài thực thi ≤ 1 000.

Subtask 2 [12 points]

Giống như subtask 1 nhưng khi chương trình kết thúc, ô (0,0) phải chứa chính xác x viên sỏi và ô (0,1) phải chứa chính xác y viên sỏi

Giới hạn: kích cỡ chương trình ≤ 200, độ dài thực thi ≤ 2 000.

Subtask 3 [19 points]

Có chính xác hai viên sỏi ở dòng 0: một viên ở ô (0, x) và một viên ở ô (0, y), x và y khác nhau và x + y là một số chẵn. Hãy viết một chương trình mà máy đo kết thúc ở ô (0, (x + y) / 2), ở điểm giữa của 2 ô chứa sỏi. Trạng thái cuối cùng của lưới không quan trọng.

Giới hạn: kích cỡ chương trình ≤ 100, độ dài thực thi ≤ 200 000.

Subtask 4 [up to 32 points]

Có tối đa 15 viên sỏi ở lưới, không có 2 viên sỏi nào trong cùng một ô. Hãy viết chương trình để tập hợp tất cả sỏi về góc tây bắc. Nói chính xác, nếu ban đầu có x viên sỏi ở lưới thì cuối cùng sẽ có chính xác x viên sỏi ở ô (0,0) và không có sỏi ở các ô còn lại.

Điểm của subtask này phụ thuộc vào độ dài thực thi của chương trình bạn nộp. Nói một cách chính xác, nếu L là giá trị lớn nhất của độ dài thực thi cho các test khác nhau, điểm của bạn sẽ là:

- 32 điểm nếu $L \leq 200\,000$;
- $32 - 32 \log_{10}(L / 200\,000)$ điểm nếu $200\,000 < L < 2\,000\,000$;
- 0 điểm nếu $L \geq 2\,000\,000$.

Giới hạn: kích cỡ chương trình ≤ 200.

Subtask 5 [up to 28 points]

Mỗi ô có thể có số lượng viên sỏi tùy ý (tất nhiên trong khoảng từ 0 đến 15). Hãy viết chương trình để tìm ô có số lượng sỏi nhỏ nhất. Nói một cách khác, máy đo kết thúc ở ô (i,j) mà tất cả các ô khác có nhiều hơn hoặc bằng số sỏi ở (i,j). Sau khi chạy chương trình, số lượng sỏi ở các ô vẫn phải như ban đầu trước khi chạy chương trình.

Điểm của subtask này phụ thuộc vào kích cỡ chương trình P của chương trình được nộp. Nói một cách chính xác, điểm của bạn sẽ là:

- 28 điểm nếu $P \leq 444$;
- $28 - 28 \log_{10}(P / 444)$ điểm nếu $444 < P < 4\,440$;
- 0 điểm nếu $P \geq 4\,440$.

Giới hạn: độ dài thực thi ≤ 44 400 000.

Chi tiết cài đặt

Bạn phải nộp đúng một file cho mỗi subtask theo luật cú pháp mô tả dưới đây. Mỗi file nộp có kích cỡ tối đa 5 MB. Với mỗi subtask, chương trình máy đo của bạn sẽ được thử trên 1 vài tests, và bạn sẽ được nhận một vài phản hồi về tài nguyên mà chương trình của bạn sử dụng. Trong trường hợp chương trình của bạn không đúng về cú pháp dẫn đến không thể chạy thử được, bạn sẽ nhận được thông tin về lỗi cú pháp cụ thể.

Các lần nộp của bạn không cần chứa các chương trình cho tất cả subtask. Nếu lần nộp của bạn không chứa chương trình cho subtask X, lần nộp cuối cùng cho subtask X sẽ được tự động thêm vào; nếu chưa có chương trình cho subtask X, lần nộp này sẽ nhận 0 điểm cho subtask X.

Như thường lệ, điểm của một lần nộp là tổng số điểm của mỗi subtask, và điểm cuối cùng là điểm lớn nhất của các lần nộp được release-test và lần nộp cuối cùng.

Bộ mô phỏng

Để kiểm tra chương trình, bạn được cung cấp một bộ mô phỏng cho máy đo mà bạn có thể sử dụng để kiểm tra chương trình của bạn với một lưới đầu vào. Chương trình cho máy đo sẽ được viết theo cú pháp như đã mô tả ở trên.

Mô tả về lưới được cho sử dụng mẫu sau: mỗi dòng trong file phải chứa 3 số R, C và P với ý nghĩa ô ở hàng R và cột C có R viên sỏi. Tất cả các ô không được nêu trong mô tả lưới sẽ không có viên sỏi nào. Ví dụ xem xét file:

```
0 10 3
4 5 12
```

Lưới được mô tả bởi file này có 15 viên sỏi, 3 viên ở ô (0,10) và 12 viên ở ô (4,5).

Bạn có thể sử dụng bộ mô phỏng bằng cách gọi chương trình `simulator.py` trong thư mục của bài, đưa tên chương trình làm tham số. Chương trình mô phỏng chấp nhận các tham số dòng lệnh sau:

- `-h` đưa ra mô tả tổng quan các tham số;
- `-g GRID_FILE` đọc mô tả lưới từ file `GRID_FILE` (mặc định: lưới rỗng);
- `-s GRID_SIDE` gán kích cỡ của lưới thành `GRID_SIDE x GRID_SIDE` (mặc định: 256, như được trong mô tả bài toán); sử dụng lưới với kích cỡ nhỏ hơn có thể dễ cho debug chương trình hơn;
- `-m STEPS` giới hạn số bước thực thi trong mô phỏng tối đa là `STEPS`;
- `-c` chuyển sang chế độ biên dịch; trong chế độ biên dịch, bộ mô phỏng trả về output như cũ, nhưng thay vì tiến hành mô phỏng sử dụng Python, nó tạo ra và biên dịch một chương trình C nhỏ gọn. Việc này ban đầu sẽ mất thời gian hơn, nhưng sẽ cho kết qủa nhanh hơn. Bạn được khuyến cáo sử dụng lựa chọn này khi chương trình của bạn phải chạy nhiều hơn 10000000 bước.

Số lần nộp

Số lần nộp tối đa cho bài này là 128.