Znajdź pudełko

Nazwa zadania	Znajdź pudełko
Limit czasu	1 sekunda
Limit pamięci	1 GB

Paulina niedawno dowiedziała się, że w piwnicy Uniwersytetu Wrocławskiego znajduje się wartościowy skarb. Skarb jest ukryty w pudełku w pustym pomieszczeniu głęboko pod ziemią. Niestety Paulina nie może tam po prostu wejść - jest tam całkowicie ciemno, a włączenie światła wzbudziłoby podejrzenia. Jednak Paulina wie, jak sobie z tym poradzić - ona i jej rodzina są zafascynowani automatycznymi kosiarkami i odkurzaczami. W związku z tym planuje zdalnie kontrolować poruszanie się automatycznego odkurzacza w piwnicy w celu znalezienia skarbu.

Piwnica jest reprezentowana jako siatka o wymiarach $H \times W$, w której rzędy są ponumerowane od 0 do $H-1$ (od góry do dołu), a kolumny są ponumerowane od 0 do $W-1$ (od lewej do prawej) - oznacza to, że komórka w lewym górnym rogu ma współrzędne $(0,0)$, a w prawym dolnym $(H-1, W-1)$. Pudełko ze skarbem znajduje się w pewnej nieznanej komórce, innej niż komórka $(0,0)$. Każdej nocy odkurzacz zaczyna w lewym górnym rogu i porusza się po piwnicy.

Każdej nocy Paulina może wydać odkurzaczowi ciąg instrukcji, jak się powinien poruszać, w formie napisu zawierającego znaki "<", ">", "^" i "v". Formalnie, jeśli odkurzacz stoi w komórce (r, c), która jest odblokowana ze wszystkich stron (nie sąsiaduje ze ścianą lub pudełkiem ze skarbem), to "<" przesuwa odkurzacz do komórki $(r, c-1)$, ">" przesuwa do $(r, c+1)$, "^" przesuwa do $(r-1, c)$, a "v" przesuwa do $(r+1, c)$.

Ściany piwnicy są grube, więc jeśli odkurzacz spróbuje wyjść poza siatkę, nic się nie stanie. Pudełko także jest porządne i nie może zostać przesunięte. Na koniec każdej nocy odkurzacz zgłasza swoją pozycją i wraca do lewego górnego rogu.

Czas to pieniądz, więc Paulina chce znaleźć pudełko tak szybko, jak to możliwe.

Interakcja

To zadanie jest interaktywne.

- Twój program powinien rozpocząć działanie od przeczytania linii z dwiema liczbami całkowitymi H i W, które oznaczają odpowiednio wysokość i szerokość siatki.

- Następnie Twój program powinien wejść w interakcję ze sprawdzaczką. W każdej rundzie interakcji powinnaś wypisać znak zapytania "`?`", a po nim niepusty napis s składający się ze znaków "`<`", "`>`", "`^`", "`v`". Długość tego napisu może wynosić co najwyżej $20\,000$. Potem Twój program powinien wczytać dwie liczby całkowite r, c ($0 \le r \le H-1,\ 0 \le c \le W-1$) - lokalizację odkurzacza po wykonaniu instrukcji. Pamiętaj, że odkurzacz zawsze wraca do komórki $(0, 0)$ po wykonaniu zapytania.
- Kiedy będziesz znała lokalizację pudełka, wypisz "`!`", a po nim dwie liczby całkowite r_b, c_b - numer rzędu i kolumny pudełka ze skarbem ($0 \le r_b \le H-1$, $0 \le c_b \le W-1$). Następnie, Twój program powinien się zakończyć bez wykonywania kolejnych zapytań. Wypisanie odpowiedzi nie liczy się jako zapytanie przy obliczaniu Twojego wyniku.

Upewnij się, żeby wyczyścić bufor standardowego wyjścia po wydawaniu zapytania - w przeciwnym przypadku Twój program może zostać oceniony jako Time Limit Exceeded. W Pythonie `print()` automatycznie czyści bufor. W C++ `cout << endl;` także czyści bufor oprócz przejścia do nowej linii; jeśli używasz printf, dodaj instrukcję `fflush(stdout)`.

Sprawdzaczka jest nieadaptywna, co oznacza, że pozycja pudełka jest wyznaczana zanim zaczyna się interakcja.

Ograniczenia i ocenianie

- $1 \le H, W \le 50$.
- Pudełko nigdy nie będzie się znajdować w komórce $(0, 0)$. To oznacza, że $H + W \ge 3$.
- Każde zapytanie może składać się z co najwyżej $20\,000$ instrukcji.
- Możesz wydać co najwyżej $2\,500$ zapytań.

Twoje rozwiązanie będzie sprawdzanie na wielu testach. Jeśli dla *pewnego* testu Twój program nie zadziała (np. zgłosi złą pozycję (WA), zakończy się błędem wykonania (RTE), przekroczy limit czasu (TLE), itp.), otrzymasz 0 punktów i odpowiedni werdykt.

Jeśli Twój program znajdzie poprawną lokalizację pudełka we *wszystkich* przypadkach testowych, otrzymasz werdykt AC, a Twoja liczba punktów zostanie obliczona w następujący sposób:

$$\text{wynik} = \min\left(\frac{100\sqrt{2}}{\sqrt{Q}}, 100\right)\ \text{punktów},$$

gdzie Q jest maksymalną liczbą wydanych przez Ciebie zapytań spośród wszystkich testów. Wypisywanie ostatecznej odpowiedzi nie liczy się jako zapytanie. Wynik zostanie zaokrąglony do najbliższej liczby całkowitej.

W szczególności, aby otrzymać 100 punktów, Twój program musi rozwiązać każdy test używając co najwyżej $Q = 2$ zapytań. Tablica poniżej zawiera kilka wartości Q i odpowiadające im wyniki.

Q	2	3	4	5	...	20	...	50	...	2500
Punktacja	100	82	71	63	...	32	...	20	...	3

Oprogramowanie do testowania

Aby ułatwić Ci testowanie rozwiązania, udostępniamy do pobrania proste narzędzie. Możesz je znaleźć w sekcji “attachments” na dole strony Kattis. Używanie tego narzędzia jest opcjonalne, możesz je także modyfikować. Zauważ, że oficjalna sprawdzaczka na Kattis jest inna niż narzędzie do testowania.

Przykładowe użycie (z $H = 4$, $W = 5$ i pudełkiem na pozycji $r = 2$, $c = 3$):

Dla programów w Pythonie, powiedzmy `solution.py` (normalnie uruchamianych jako `pypy3 solution.py`):

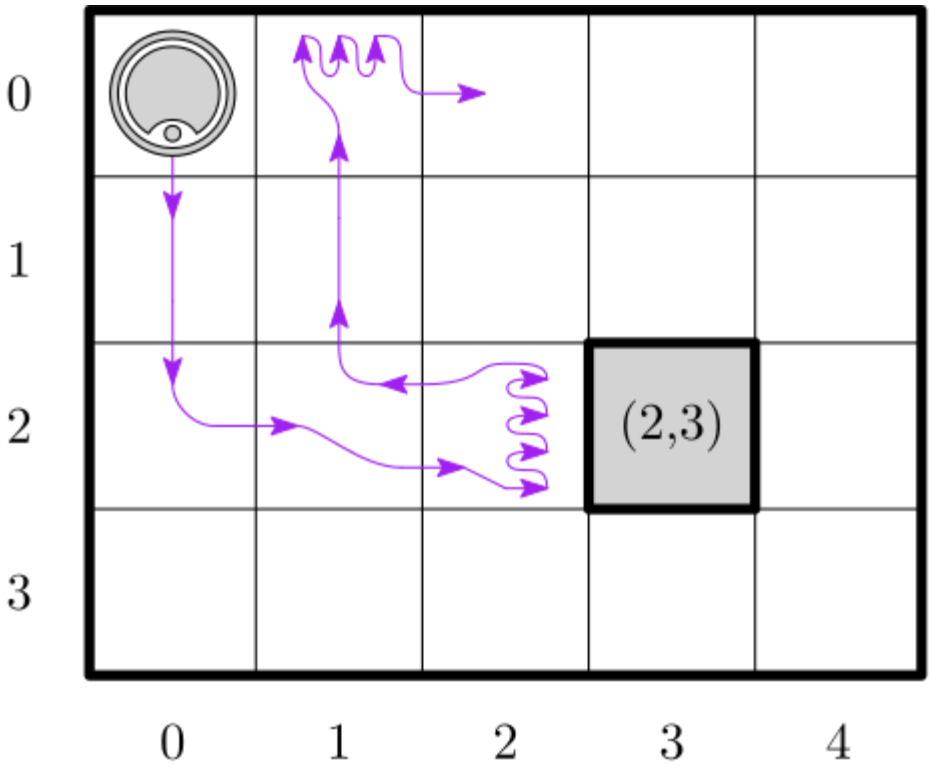
```
python3 testing_tool.py pypy3 solution.py <<<"4 5 2 3"
```

Programy w C++ najpierw należy skompilować (np. poprzez `g++ -std=gnu++17 solution.cpp -o solution.out`), a następnie uruchomić

```
python3 testing_tool.py ./solution.out <<<"4 5 2 3"
```

Przykład

Rozważmy następujący test przykładowy. Siatka ma wysokość $H = 4$ i szerokość $W = 5$, a pudełko znajduje się na pozycji $(r, c) = (2, 3)$. Ilustracja poniżej ukazuje ścieżkę odkurzacza, kiedy wykonuje instrukcje z pierwszego zapytania “`? vv>>>>>><^^^^^>`”, które powoduje, że odkurzacz na koniec znajdzie się w komórce $(r, c) = (0, 2)$. Przed drugim zapytaniem odkurzacz wraca do lewego górnego rogu - czyli na pozycję $(0, 0)$. Następnie program wydaje kolejne zapytanie “`? >>>>>>>>vvvvvvvvvv`”, po którym odkurzacz znajdzie się w prawym dolnym rogu - czyli na pozycji $(r, c) = (3, 4)$. Teraz program decyduje się na zgadnięcie odpowiedzi poprzez wypisanie “`! 2 3`”, które jest poprawną pozycją pudełka.



Wyjście sprawdzaczki	Twoje wyjście
4 5	
	? vv>>>>>><^^^^^>
0 2	
	? >>>>>>>>vvvvvvvvv
3 4	
	! 2 3