Fliegende Orks

> Entgegen anders lautender (vornehmlich vom *Verband der Landschaftspfleger Mittelerdes* verbreiteter) Überlieferungen war es nicht Samweis Gamdschie, der Frodo aus dem Turm von Cirith Ungol befreite…

Frodo ist (mal wieder) in Gefahr und somit ist es an Aragorn, Legolas und Gimli, den in Mordor gefangenen Hobbit zu erretten. In Ermangelung eines mit sich selbst sprechenden Reiseführers müssen sie sich dazu querfeldein durch Feindesland schlagen. Da jedoch ein wesentlicher Bestandteil ihres Plans darauf fußt, *unerkannt* zum Turm zu gelangen, sehen sie sich mit dem Problem konfrontiert, Saurons (hoffentlich nicht wirklich allsehendem) Auge zu entgehen.

Zu diesem Zweck möchten sie zunächst das von Sauron beobachtete Gebiet bestimmen: der ordnungsliebende Legolas hat dazu passend eine in Planquadrate unterteilte Karte Mordors mit den Abmaßen $m \times n$ zur Hand. Es gilt:

- Jedes Planquadrat ist entweder vollständig in Saurons Blickfeld oder vollständig außerhalb.
- Das beobachtete Gebiet ist vollständig in der Karte enthalten.
- Die Freunde kennen einen Punkt (den Schicksalsberg), der definitiv zum beobachteten Gebiet gehört.
- Das von Sauron beobachtete Gebiet ist zusammenhängend und enthält keine Löcher, kann davon abgesehen aber beliebig unregelmäßig sein.

Hierbei nennen wir zwei Felder benachbart, wenn sie eine Kante teilen; *zusammenhängend* bedeutet, dass je zwei beobachtete Felder durch eine Folge jeweils benachbarter, beobachteter Felder verbunden werden können, und *ohne Loch* bedeutet umgekehrt, dass jedes unbeobachtete Feld durch eine Folge jeweils benachbarter, unbeobachteter Felder mit dem Rand der Karte verbunden ist.

Glücklicherweise sind von der Schlacht auf dem Pelennor noch ein Katapult und ein paar Orks übrig geblieben. Legolas hat nun vorgeschlagen, Orks auf vorgegebene Punkte zu schießen, um damit an der Reaktion des Auges festzustellen, ob das entsprechende Planquadrat beobachtet wird (ein früherer Vorschlag, Gimli zu werfen, wurde unter lautstark vorgetragenen zwergischen Flüchen abgelehnt). Da jedoch die Zeit drängt, wollen sie mit möglichst wenig Versuchen auskommen. Hilf ihnen, indem du ein Programm schreibst, das für sie die Zielpunkte auswählt!

Ablauf

Das ist eine interaktive Aufgabe. Zu Beginn kannst du von der Standardeingabe die ganzen Zahlen m und n durch Leerzeichen getrennt lesen. Die nächste Zeile enthält in gleicher Weise die Koordinaten x, y eines Punktes, der von Sauron beobachtet wird. Es gilt $1 \leq x \leq m$ und $1 \leq y \leq n$.

Im Folgenden kannst du Anfragen an das Programm stellen. Diese haben die Form `1` x y, wobei x, y im Format wie oben die Koordinaten des Punktes angeben, auf das mit dem Katapult geschossen wird. Daraufhin bekommst du über die Standardeingabe eine einzelne Zahl: `1`, wenn das entsprechende Feld beobachtet wird, und `0` sonst.

Wenn dein Programm weiß, welches Gebiet Sauron beobachtet, gib eine einzelne `2` und daraufhin n Zeilen aus je m durch Leerzeichen getrennten Zahlen aus. Die i-te Zahl in Zeile j soll dabei eine `1` sein, wenn das Planquadrat (i, j) beobachtet wird, sonst eine `0`.

Wichtiger Hinweis

Um eine reibungslose Kommunikation zu gewährleisten, musst du nach jeder deiner Ausgaben die Standardausgabe »flushen«. Wenn du *cout* verwendest, genügt es, am Ende jeder Ausgabe $\ll$ *flush* zu verwenden, also z.B.

$$\textit{cout} \ll \text{"1 5 1\textbackslash n"} \ll \textit{flush};$$

oder generell *endl* für Zeilenumbrüche zu verwenden. Bei Verwendung von *printf* musst du nach jeder Ausgabe *fflush*(*stdout*) aufrufen.

Beschränkungen und Bewertung

In allen Testfällen gilt $1 \leq m, n \leq 500$.

Teilaufgabe 1 (40 Punkte). $m \leq 200$ und $n \leq 250$

Teilaufgabe 2 (40 Punkte). $m, n \leq 400$

Teilaufgabe 3 (20 Punkte). Keine weiteren Beschränkungen.

Für die Bewertung wird dein Programm mit unserer (sehr effizienten) Musterlösung verglichen. Die Punktzahl für eine Teilaufgabe berechnet sich wie folgt: Für jeden Testfall darfst du höchstens 250 000 Anfragen stellen (diese Grenze ist nur symbolisch, denn sie reicht, um jedes Feld abzufragen), stellst du mehr Anfragen, folgst nicht dem obigen Format oder gibt dein Programm die falsche Antwort aus, erhältst du für diese Teilaufgabe 0 Punkte. Andernfalls bilden wir für jeden Testfall das Verhältnis

$$\alpha := \frac{\text{geworfene Orks in deiner Lösung}}{\text{geworfene Orks in der Musterlösung}}$$

und betrachten das maximale Verhältnis $\alpha_{\max}$ innerhalb dieser Teilaufgabe. Die folgende Tabelle zeigt an, wie viele Punkte du abhängig davon erhältst:

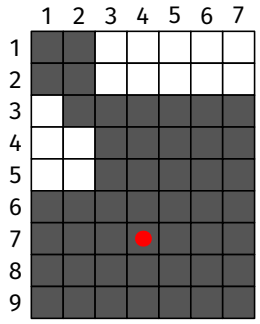
$\alpha_{\max}$	$[0, 2]$	$(2, 5]$	$(5, 20]$	$(20, 50]$	$(50, \infty)$
Anteil der Punkte	100%	75%	50%	25%	0%

Beachte, dass die bepunkteten Testfälle so gewählt sind, dass

- ... eine Lösung, die immer alle Felder abfragt, keine Punkte erhält.
- ... die Anzahl der benötigten Abfragen vergleichsweise groß ist (sodass additive Konstanten für die Bewertung quasi keine Rolle spielen).

Beispiel

Das Gebiet



stellt eine gültige Instanz dar. Beachte, dass es hingegen *nicht* gültig wäre, wenn das Feld 2,3 fehlen würde (dann wäre das Gebiet gemäß unserer Definition nicht zusammenhängend). Ebenso wäre es nicht gültig, wenn z.B. das Feld 3,6 fehlen würde (dann hätte das Gebiet ein Loch).

Eine Interaktion könnte hier wie folgt aussehen:

Dein Programm	Grader	Erklärung
	`7 9`	Abmaße der Karte
	`4 7`	ein bekanntermaßen beobachtetes Feld
`1 1 1`		Abfrage von Feld 1, 1
	`1`	das Feld ist beobachtet
`1 5 1`		Abfrage von Feld 5, 1
	`0`	Feld ist nicht beobachtet
`2`		dein Programm will die Lösung ausgeben
`1 1 0 0 0 0 0`		
`1 1 0 0 0 0 0`		
`0 1 1 1 1 1 1`		
`0 0 1 1 1 1 1`		
`0 0 1 1 1 1 1`		
`1 1 1 1 1 1 1`		
`1 1 1 1 1 1 1`		
`1 1 1 1 1 1 1`		
`1 1 1 1 1 1 1`		
		die Lösung ist korrekt und wird akzeptiert

Beachte allerdings, dass die gestellten Abfragen natürlich nicht genügen, um das Gebiet tatsächlich zu ermitteln. Unsere Musterlösung benötigt beispielsweise 40 Abfragen.

Limits

Zeit: 2 s
Speicher: 1024 MiB

Die obige Zeitangabe enthält auch die Laufzeit unseres Graderprogramms und insbesondere die Zeit für die Kommunikation zwischen den beiden Programmen; es ist aber garantiert, dass in C++ das Stellen der 250 000 erlaubten Anfragen und die Ausgabe der Lösung in unter 2 Sekunden möglich ist.

Feedback

Für diese Aufgabe ist *restricted feedback* verfügbar. Das bedeutet, die angezeigte Punktzahl entspricht der endgültigen Punktzahl deiner Einsendung. Allerdings wird dir für jede Testfallgruppe immer nur der erste Testfall mit minimaler Punktzahl innerhalb der entsprechenden Gruppe angezeigt. (Hierbei ist die Reihenfolge der Fälle innerhalb der jeweiligen Gruppen fest.)