Blinde Kuh

Der kleine Wolfgang hat bald Geburtstag. Bei seiner Geburstagsfeier möchte er mit seinen Freunden von der AOI sein Lieblingsspiel "Blinde Kuh" spielen. Das Spiel funktioniert so:

- Wolfgang wird mit verbundenen Augen in einen Raum geführt, indem irgendwo unter einem Kochtopf Süßigkeiten versteckt sind.
- Mit einem Kochlöffel ausgestattet muss Wolfgang versuchen den Topf zu finden. Dazu kann er mit dem Kochlöffel auf den Boden (bzw. schließlich den Kochtopf) schlagen.
- Jedes mal, wenn Wolfgang mit dem Kochlöffel zuschlägt, bekommt er von seinen Freunde einen Tipp: Fall er sich seit seinem letzten Schlag dem Kochtopf genähert hat, rufen sie ihm "Wärmer" zu. Ansonsten "Kälter".

Als routinierte AOI Trainer fragen sich die Freunde, was die minimal notwendige Anzahl an Tipps sind, um den Kochtopf zu finden. Kannst du ihnen helfen?

Implementierungsdetails

Wir betrachten den Raum als $n \times n$ Gitter, dessen Koordinaten von 0 bis $n-1$ durchnummeriert sind. Der Kochtopf befindet sich an ganzzahligen Koordinaten und es kann nur auf ganzzahlige Koordinate geschlagen werden.

Implementiere die Funktion `blinde_kuh(n)`, wobei n die Raumgrößte angibt. Um mit dem Kochlöffel zuzuschlagen und herauszufinden, ob du dich seit deiner letzten Aktion dem Kochtopf genähert hast, kannst du die Funktion `hit(x, y)` aufrufen. `x` und `y` müssen dabei ganzzahlige Koordinaten zwischen 0 und $n-1$ sein. Die Funktion:

- Beendet das Programm, falls (x, y) den Koordinaten des Kochtopfs entspricht.
- Gibt beim ersten Aufruf `true` zurück.
- Gibt bei jedem folgenden Aufruf entweder `true` oder `false` zurück. Es wird `true` zurückgegeben, falls die Distanz von (x, y) zum Kochtopf *strikt kleiner* als beim letzten Aufruf von `hit` ist. Als Distanz wird die *euklidische Distanz*[1] verwendet.

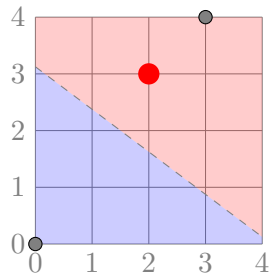
Dein Ziel ist es den Kochtopf zu finden, also `hit` mit den Koordinaten des Kochtopfs aufzurufen. Falls das innerhalb der Ausführung von `blinde_kuh` nicht passiert, zählt deine Lösung als falsch. Deine Lösung zählt auch als falsch, falls du `hit` mit Koordinaten außerhalb des Raumes aufrufst. Ansonsten hängen deine Punkte von der Anzahl an Aufrufe von `hit` ab. Siehe Scoring.

[1] Für zwei Punkt (x_1, y_1), (x_2, y_2) entspricht das $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

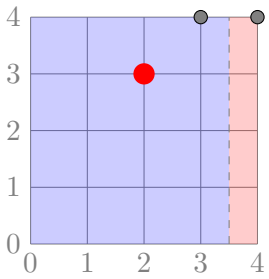
Beispiel

Angenommen der Raum hat Seitenlänge $n = 5$ und der Kochtopf befindet sich an Position $(2, 3)$. Dann könnte dein Programm folgendermaßen ablaufen:

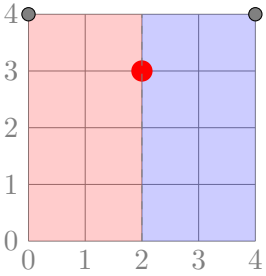
Aufrufe
`blinde_kuh(n = 5)` wird vom Grader aufgerufen.
`hit(0, 0)` gibt true zurück.
`hit(3, 4)` gibt true zurück, weil $\sqrt{(2-3)^2+(3-4)^2} < \sqrt{2^2+3^2}$.
`hit(4, 4)` gibt false zurück, weil $\sqrt{(2-4)^2+(3-4)^2} > \sqrt{(2-3)^2+(3-4)^2}$.
`hit(0, 4)` gibt false zurück, weil $\sqrt{2^2+(3-4)^2} = \sqrt{(2-4)^2+(3-4)^2}$.
`hit(2, 3)` beendet das Programm, weil $(2, 3)$ die Koordinaten des Kochtopfs sind.



(a) Der zweite Versuch liegt näher am Topf (roter Bereich), daher gibt `hit true` zurück.



(b) Der dritte Versuch ist weiter vom Topf entfernt (blauer Bereich), daher gibt `hit false` zurück.



(c) Der vierte Versuch ist gleich weit entfernt wie der vorherige, daher gibt `hit false` zurück.

In diesem Fall hat das Programm 5 Aufrufe von `hit` gemacht.

Subtasks und Scoring

Allgemein gilt: $1 \leq n \leq 10^9$.

Für die ersten 4 Subtasks ist jeweils ein Limit L der erlaubten `hit` Abfragen definiert. Falls du mehr Abfragen verwendest, zählt deine Lösung als falsch. Im letzten Subtask skalieren deine Punkte mit der Anzahl an Abfragen.

Subtask 1 (5 Punkte): $n \leq 1000$ und $L = 1000$. Die y-Koordinate des Kochtopfs ist 42.

Subtask 2 (10 Punkte): $n \leq 100$ und $L = 10^4$

Subtask 3 (15 Punkte): $n \leq 1000$ und $L = 2000$

Subtask 4 (20 Punkte): $L = 69$. Die y-Koordinate des Kochtopfs ist 42.

Subtask 5 (bis zu 50 Punkte):

$$\text{Punkte}(x \text{ Aufrufe von } \texttt{hit}) = 50 \cdot \min\left(1, \frac{69}{x}\right)$$

Wobei für x das Minimum über alle Testcases des Subtasks genommen wird.

Eingabe

Die Eingabe wird über ein von uns vorgegebenes Programm, den Grader, übernommen. In der ZIP-Datei zu dieser Aufgabe findest du für jede Programmiersprache eine grader-Datei, das du zum Testen deiner Lösung mit deinem Programm kompilieren kannst.

Das Eingabeformat ist eine Zeile mit 3 mit Leerzeichen getrennte Zahlen: n, x und y. n ist die Seitenlänge des Raumes, x und y sind die Koordinaten des Kochtopfs. Die Koordinaten des Kochtopfs liegen immer innerhalb des Raumes.

Wenn dein Programm den Kochtopf findet, wird die Anzahl queries auf der Ausgabe ausgegeben. Wenn dein Programm den Kochtopf nicht findet, wird `target not found` ausgegeben und das Programm beendet mit einem Fehlercode.