Aufgabe 1: Laubmaschen

Teilnahme-ID: 69082

Bearbeiter dieser Aufgabe:
Vincent Xigu Liu

05. März 2024

Inhaltsverzeichnis

Im Laubblaseproblem geht es darum, wie der Hausmeister im quadratischen Feld das Laub möglichst auf ein Feld versammelt. Hierbei betrachten wir allgemeiner direkt denn Fall, dass der Hausmeister sich das entsprechende Feld aussuchen kann, an dem er das Laub bläst. Da wir zudem feststellen, dass eine Verallgemeinerung zum rechteckigen Feld als Erweiterung gut mit den bisherigen Lösungsansätzen vereinbar ist, behandeln wir im Folgenden den allgemeineren Ansatz mit rechteckigen Feldern. Da quadratische Felder auch rechteckig sind, lassen sich also alle im Folgenden verwendeten Ansätze auch als Lösungen für den quadratischen Schulhof verstehen.

Wenn wir das Feld als eine zweidimensionale Liste modellieren, können wir den Blasevorgang als Umverteilung ansehen, wobei man zwei Felder der Liste auswählt und dann den Blasevorgang durchführt. Es sei dazu der gegebene Schulhof S und $S_{i,j}$ sei dabei das i-te Feld in der j-ten Spalte des Schulhofes und der Wert $S_{i,j}$ die Anzahl der Blätter auf diesem Feld. I und j seien dabei natürliche Zahlen größer als 2, da ein Schulhof mit einem Planquadrat bereits fertiggepustet ist und

Schulhöfe mit einer Länge oder Breite von 2 kein Feld haben, welches kein Randquadrat ist und somit der Abfuhrwagen keinen Platz zum Parken hat. Der Blasevorgang über ein Feld, der nun einen Algorithmus zur Veränderung von S darstellt und dabei ein Feld und eine Richtung (oben, unten, links, rechts) als Eingabe nimmt, wird durch die Koordinaten $F =$ (i,j) und die Richtungsvektoren $\vec{r}$ = ((0,-1), (0,1), (1,0), (-1,0)) dargestellt. Das Ziel ist es nun, die Laubanzahl in S für ein gegebenes Feld S_{ij} zu maximieren oder zumindest möglichst viel Laub in die Nähe des Feldes zu bringen, um dem Hausmeister seine Arbeit beim Zusammenblasen des Laubes zu vereinfachen.

1. Überlegungen zum Simulationsprogramm

Zunächst wollen wir uns Gedanken darüber machen, wie es möglich ist, ein Simualtionsprogramm entsprechend der Aufgabenstellung und zu Implementieren. Dazu untersuchen wir zunächst, welche Umsetzung für die vorgegebenen Regeln sinnvoll sind und beziehen uns dann auf Fälle, welche den Rand und die Ecken betreffen, um uns dort realistische Ansätze zu überlegen. Da der Hausmeister auf einem Feld stehen muss, um auf ein anderes zu pusten, nennen wir einen Zug Legitim, wenn das Feld, worauf er steht sowie das Feld, worauf er pustet beide Teile des Schulhofes sind. Daher nimmt das Programm zur Simulation des Blasevorgangs auch zunächst die Eingaben des Feldes $S_{i,j}$, auf den der Hausmeister steht, und die Pusterichtung $\vec{r}$, in die geblasen werden soll, um daraufhin den Blasevorgang zu simulieren.

1.1 Ansätze für die vorgegebenen Regeln

Wenn die beiden Eingaben für das Programm getätigt wurden, ist nun das Feld, von dem Laub geblasen wird, $S_{i,j} + \vec{r}$, das Feld, auf das das Laub landet, $S_{i,j} + 2 * \vec{r}$ und das entsprechende Feld dahinter $S_{i,j} + 3 * \vec{r}$. Da durch das Vertauschen der x- und y-Koordinate alle von uns betrachteten Vektoren um 90° auf der Ebene gedreht werden, stellt $S_{i,j} + 2 * \vec{r} + \vec{s}$ ein Seitenfeld dar, wenn $\vec{s}$ der gedrehte Vektor ist. Die Drehung können wir dadurch erzielen, dass wir die x-und y-Koordinaten des Vektors $\vec{r}$ miteinander vertauschen. Da die Seitenfelder auf den gegenüberliegenden Seiten des geblasenden Feldes liegen, ist das andere Seitenfeld entsprechend $S_{i,j} + 2 * \vec{r} - \vec{s}$ (siehe Abb.1)[1].

Um die Laubblätter einzeln zu simulieren, nutzen wir hierfür einen Pseudo-Zufallszahlengenerator aus der Python-Bibliothek `random`. Für jedes Blatt aus $S_{i,j} + 2 * \vec{r}$ generieren wir also zunächst eine natürliche Zufallszahl zwischen 0 und einschließlich 9 und subtrahieren genau dann 1 von $S_{i,j} + 2 * \vec{r}$ und addieren 1 auf $S_{i,j} + 3 * \vec{r}$, wenn diese Zahl 0 ist. Diesen Prozess wiederholen wir für jedes Blatt genau ein mal. Analog generieren wir dann für jedes Blatt aus $S_{i,j} + \vec{r}$ zwei natürliche Zufallszahlen zwischen 0 und einschließlich 9,

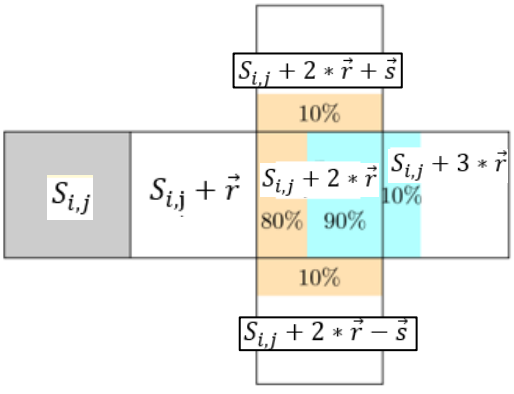

Abb. 1: Felder nach Beschreibung durch dem Blasefeld $S_{i,j}$ und dem Richtungsvektor $\vec{r}$.

[1] Abbildung nach https://bwinf.de/fileadmin/bundeswettbewerb/42/aufgaben422.pdf

und addieren 1 zu in $S_{i,j} + 2 * \vec{r} + \vec{s}$, wenn die erste Zahl 0 beträgt beziehungsweise addieren 1 zu $S_{i,j} + 2 * \vec{r} + \vec{s}$ und ziehen entsprechend vom Feld $S_{i,j} + \vec{r}$ die Blätter ab. Der Rest der Blätter wird dann auf das Feld $S_{i,j} + 2 * \vec{r}$ gepustet. Dadurch kann das Laubfeld entsprechend der Regeln verändert werden und die Simulation für jedes Blatt auf den betreffenden Feldern einzeln durchgeführt werden.

1.2 Randregeln

Dass alle Felder aus Abb.1 nun auch auf dem Schulhof liegen, muss nicht immer der Fall sein. In solchen Fällen ist es daher sinnvoll, sich realistische Regeln selbst zu überlegen. Im Folgenden gehen wir davon aus, dass der Schulhof innerhalb des Schulgebäudes liegt, also überall am Rande des Hofes Wände des Schulgebäudes stehen. Dadurch ist es nicht möglich, dass durch Pusten Laub vom Rand des Schulhofes verloren geht. Im Folgenden versuchen wir modellierend Regeln für ein solches Phänomen darzustellen:

Fall 1: $S_{i,j} + 2 * \vec{r}$ liegt außerhalb des Schulhofes.

In einem solchen Fall versucht der Hausmeister prinzipiell, gegen eine Wand zu pusten. Da das Laub, welches $S_{i,j} + \vec{r}$ verlässt, nun keinen Platz mehr auf $S_{i,j} + 2 * \vec{r}$ hat, wegen dem Laubdruck jedoch woanders hinmuss, nehmen wir modellierend an, dass jedes Laubblatt mit Wahrscheinlichkeit 50% beschließt, jeweils auf $S_{i,j} + 2 * \vec{r} + \vec{s}$ beziehungsweise auf $S_{i,j} + 2 * \vec{r} - \vec{s}$ zu landen.

Fall 2: $S_{i,j} + 3 * \vec{r}$ liegt außerhalb des Schulhofes.

Da hier die Blätter auf $S_{i,j} + 2 * \vec{r}$ analog zu Fall 1 nicht weiter nach vorne gepustet werden können, jedoch die „Blaskraft“ des Laubbläsers schwächer ist, verbleiben diese einfach auf das Feld $S_{i,j} + 2 * \vec{r}$.

Fall 3: $S_{i,j} + 2 * \vec{r} - \vec{s}$ oder $S_{i,j} + 2 * \vec{r} + \vec{s}$ liegen außerhalb des Schulhofes.

In einem solchen Fall wir das Laubblatt zwar von $S_{i,j} + \vec{r}$ geblasen, kann jedoch nicht auf die Seitenfelder landen. Wir nehmen daher modellierend an, dass die Wand die Laubblätter einfach zurückhält und alle Blätter, welche auf ein Seitenfeld hätten landen sollen, stattdessen auf $S_{i,j} + 2 * \vec{r}$ landen.

Wir beachten dabei insbesondere, dass die Fälle nicht exklusiv eintreten: Wenn Fall 1 eintritt, so tritt Fall 2 automatisch ebenfalls ein. Wenn Fall 2 eintritt, kann Fall 3 möglicherweise auch eintreten. In solchen Fällen ist es wichtig, die betrachteten Felder von denen Laub gepustet werden soll, falls überhaupt im Schulhof liegend, einzeln zu simulieren und dabei die Reihenfolge der Fallnummern einzuhalten.

1.3 Laufzeit des Simulationprogramms

Wenn wir von einer konstanten Laufzeit zur Generierung einer Zufallszahl ausgehen, ist die Laufzeit eines derartigen Simulationsprogramms nicht höher als $O(n + k)$, wobei n die Anzahl der Laubblätter in den Feldern auf Abb.1 außer $S_{i,j}$ ist und k eine Konstante ist, um zu ermitteln, welche der beachteten Felder nicht im Schulhof liegen, da das Programm jedes Laubblatt betrachet, eine Zufallszahl für diese generiert und das Blatt entsprechend der Zufallszahl im Schulhof umverlegt. Wenn wir annähernd von einer festen benötigten Anzahl an Zügen z(g), welche von der Feldgröße g abhängt, ausgehen (was aber nicht der Fall hier ist), können wir annähernd sagen, dass damit die Gesamtlaufzeit zur Simulation einer kompletten Blasesequenz $O(z(g) * (n + k))$ nicht übersteigt, wobei wir die Feldgröße g = $i * j$ als die Anzahl der Planquadrate im Schulhof definieren und n die durchschnittliche Anzahl der Blätter auf den Feldern darstellt.

2. Heuristische Untersuchungen zum Lösungsansatz

Im folgenden Abschnitt wollen wir verschiedene Lösungsansätze untersuchen und diskutieren, welche zum Blasen auf ein Feld sinnvoll sind. Da der Simulationsprozess allerdings mit Zufallsvariablen bestückt ist, muss man in der Regel dynamische Strategien wählen, um zufällige Abweichungen vom Erwartungszustand des Schulhofes mit berücksichtigen zu können. Bevor wir allerdings auf allgemeine Blasestrategien zurückkommen, untersuchen wir zunächst das Blasen der Blätter aus zwei Feldern Entfernung sowie das Blasen der Blätter vom Rand, welche leichter zu lösende Spezialfälle des Laubblaseproblems darstellen.

2.1 Sonderfall: Blasen von zwei Feldern Entfernung

Wenn der Hausmeister ein Planquadrat bepustet, welches zwei Felder von ihm entfernt ist, ist es dem Hausmeister möglich, sämtliche Blätter vom bepusteten Planquadrat in das Zielquadrat zu befördern. Ein solcher Sonderfall ist dann möglich, wenn das betrachtete Planquadrat als $S_{i,j} + 2 * \vec{r}$ ausgedrückt werden kann, wobei $S_{i,j}$ Teil des Schulhofes ist. Das Quadrat, auf dem dann Laub gepustet wird, ist entsprechend $S_{i,j} + 3 * \vec{r}$. Der Hausmeister steht dann zunächst auf $S_{i,j}$ und pustet wiederholt auf das Feld $S_{i,j} + \vec{r}$. Dadurch wird jedes Blatt aus $S_{i,j} + 2 * \vec{r}$ mit einer Wahrscheinlichkeit von 10% auf das Feld $S_{i,j} + 3 * \vec{r}$ gepustet. Die erwartete Anzahl[2] der Blätter, die also in $S_{i,j} + 3 * \vec{r}$ gepustet wird, beträgt $(S_{i,j} + 2 * \vec{r}) * 0{,}1$. Da das Feld $S_{i,j} + 2 * \vec{r}$ nach konsekutivem Pusten erst 0 wird, wenn alle Blätter auf das Feld $S_{i,j} + 3 * \vec{r}$ gelandet sind, ist die erwartungswert der Blätter immer positiv. Da zudem keine Blätter vom Feld $S_{i,j} + 2 * \vec{r}$ gepustet werden, steigt die Anzahl der Blätter auf diesem Feld während des Blasevorgangs immer. Dies führt dazu, dass nach endlich vielen Wiederholungen des Pustens im Erwartungsfall alle Blätter irgendwann auf $S_{i,j} + 2 * \vec{r}$ landen.

Wenn wir die Anzahl der Pustwiederholungen untersuchen, die erwartet nötig sind, um alle Blätter in das Feld zu bringen, nehmen wir modellhaft an, dass ab einer erwarteten Anzahl von 0.5 Blättern keine Blätter mehr auf dem Feld $(S_{i,j} + 2 * \vec{r})$ liegen. Nun untersuchen wir, wie viele Wiederholungen nötig sind, um alle bis auf theoretische 0.5 Blätter auf $S_{i,j} + 2 * \vec{r}$ zu pusten. Da nach jeder Pustewiederholung erwartungsgemäß nur noch 90% der Blätter in $S_{i,j} + \vec{r}$ überbleibt, gilt die Gleichung:

$$x * 0.9^{y} \; = \; 0{,}5; \; x, y \geq 0$$

Wobei x die Anzahl der Blätter und y die Anzahl der Pustewiederholungen ist. Nach umstellen erhalten wir:

$$y = log_{0{,}9}\left(\frac{0{,}5}{x}\right) \qquad (1.1.4)$$

Welches die Anzahl der erwarteten Wiederholungen beschreibt, die nötig sind, um alle Blätter bis auf eins in $S_{i,j} + \vec{r}$ wegzupusten. Diese theoretische Modellierung entspricht den mithilfe eines Python Programms berechneten Simulationsergebnissen, die anhand des folgenden Pseudocodes durchgeführt wurden (siehe Abb. 4):

Algorithmus 1. Laubblatt-Simulation

```
Blätter ← x
Pusteanzahl ← 0
While Blätter > 0 do:
    Pusteanzahl ← Pusteanzahl + 1
    For i = 0, i < Blätter, i ← i + 1 do:
        rand ← Zufallszahl zwischen einschl. 0 und 9
```

[2] https://de.wikipedia.org/wiki/Erwartungswert

If rand = 0 **do:**
Blätter ← Blätter − 1
Return Pusteanzahl

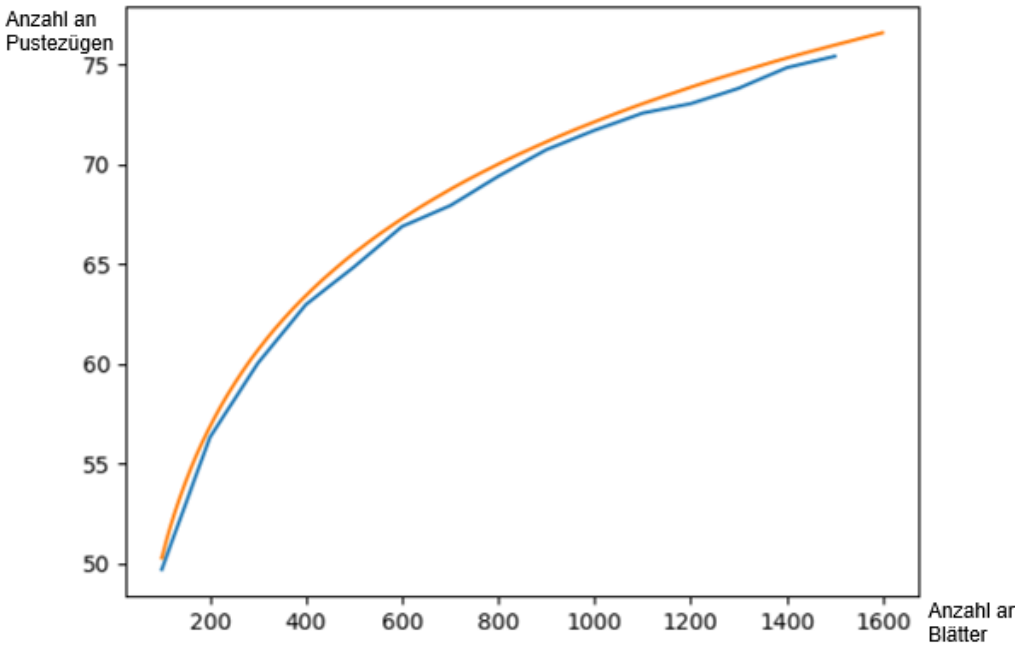

Abb. 4: Simulationsergebnisse mithilfe des Algorithmus 1. Es wurde für jede Anzahl der Blätter 5000 mal die nötige Pusteanzahl simuliert und davon der Mittelwert genommen. Blau unterlegt ist der simulierte Graph, orange unterlegt ist der Graph der Gleichung von (1.1.4) zu sehen.

Anhand dieser Untersuchungen erkennen wir, dass die Anzahl der Pustezügen einen Logarithmischen Zusammenhang mit der Anzahl der zu pustenden Blätter auf ein Feld aufweist. Da die Laufzeit zum Bepusten der Blätter als konstant angenommen werden kann, lässt sich daraus beim Design der Algorithmen auf die Laufzeit schließen.

2.2 Sonderfall: Bepustung der Ränder

Wenn wir die Ränder bepusten möchten, fällt auf, dass diese nicht sehr schnell vom Rand des Schulhofes gepustet werden können, weil analog zum Abschnitt 2.1 immer nur ein Teil der Blätter durch die Seitenverluste vom Rand geblasen werden kann und dieser Vorgang entsprechend lange dauert. Insbesondere in der Implementation eines eher schnellen Algorithmus mit wenigen Pustevorgängen müsste der Hausmeister also sehr viel und sehr lange pusten. Da der Hausmeister aber auch Besen und Tüte hat, wollen wir für einen schnellen Ansatz möglichst schnell möglichst viele Blätter in eine Ecke des Hofes pusten, von wo der Hausmeister anschließend diese auffegen kann. Dazu startet der Hausmeister an einer Ecke und entfernt dort alle Blätter, welche in der Ecke liegen, durch wiederholtes Pusten in die Ecke. Anschließend

3. Lösungsansätze

Im Folgenden präsentieren wir nun verschiedene Lösungsansätze für das Laubblasen und Diskutieren diese anhand der Güte der Lösungen sowie anhand ihrer Effizienz.

3.1 Greedy Ansatz

Im folgenden Greedy Ansatz möchten wir einen Greedy Algorithmus schreiben, welchen den Schulhof analysiert, mögliche Pustezüge des Hausmeisters berechnet und bewertet, um anschließend den besten Zug herauszugeben. Dabei wird ein Greedy Schritt so oft ausgeführt, bis die Bewertung des Schulhofes nicht mehr verbessert werden kann. Ein Ansatz mit einem Greedy Algorithmus eignet sich hierbei insbesondere für die Aufgabe, da das Sammeln des Laubes durch Pustevorgänge eine stetige Annäherung von allen Laubblättern zum Zielquadrat darstellt und daher insbesondere für große Felder eine gute Bewertungsfunktion mit nicht sehr tiefer Suche auch effizient und schnell zum optimalen Ergebnis führen kann. Der Greedy-Algorithmus in drei Teile gegliedert, welche im Folgenden erläutert werden:

1. Ermitteln aller legitimen Züge (wird nur einmal ausgeführt)

Wiederhole bis Bewertung nicht mehr optimierbar:

2. Errechnen der Schulhofbewertung anhand des Pusteverhaltens des Hausmeisters
3. Ermitteln des besten Zuges

1. Ermitteln aller legitimen Züge

Um zunächst alle möglichen Pustezüge in einem Schulhof ermitteln zu können, suchen wir alle Kombinationen heraus und speichern diese in eine Liste. Da die möglichen Züge sich nicht durch das Umverteilen der Blätter ändern, müssen wir diesen Schritt nur einmal durchführen. Die Anzahl der legitimen Züge ist dabei bei sehr großen Schulhöfen annähernd $4 * i * j$, ist aber nie kleiner als $2 * i * j$, da man in sehr großen Schulhöfen fast von jedem Feld in vier Richtungen pusten kann, es jedoch keinen Schulhof gibt, bei dem man an einem Feld nur in eine Richtung oder weniger pusten kann (da i, j > 2). Diese Anmerkung wird später bei der Laufzeitanalyse noch wichtig.

2. Errechnen der Schulhofbewertung nach den Zügen

Da in den meisten Fällen aufgrund der Varianzen durch den zufälligen Pustevorgang genaue Blätterverteilungen nicht vorhersehbar sind (oder nach vielen vorausberechneten Pustezügen vom erwartungswert stark abweichen), berechnen wir die Güte des Zuges direkt anhand einer heuristischen Bewertung und führen keine Tiefensuche der Züge durch, um die optimalste Strategie des Hausmeisters zu überprüfen, was die Laufzeit unseres Programms deutlich verkürzt und damit auch für große Schulhöfe die Rechenzeit realistisch macht. Dafür seien $Z = (s, t)$ die Koordinaten des Zielfeldes. Die *Entfernung zu Zielfeld* eines Feldes S = (i, j) definieren wir als die Anzahl der Felder, über die man von S mindestens gehen muss, um Z zu gelangen (vergleiche Abb.4). Formal ist diese Entfernung also $|i - s| + |j - t|$ und die Bewertung ist dann die gewichtete Summe der Entfernungen aller Blätter vom Zielquadrat:

$$\sum_{\forall i} \sum_{\forall j} (|i - s| + |j - t|) * S_{i,j}, Z = (s, t)$$

Die Bewertung, die ein Schulhof erhält, richtet sich zusätzlich daran, ob ein Blatt auf einem Randquadrat oder gar einem Eckquadrat liegt. Dies liegt daran, dass man nur durch Seitenverluste Blätter vom Rand pusten kann. Dadurch verschlechtert sich die Bewertung, wenn man versucht, die Blätter von der Mitte des Randes in Richtung Ecke zu pusten, um durch Seitenverluste einige Blätter vom Rand zu treiben. Um dem entgegenzuwirken, addieren wir zu jedem Randfeld die Kantenlänge, um das Entfernen der Blätter zu motivieren. Die nun modifizierten Gewichte sind in Abb.3

einzusehen. Es ist wichtig zu erwähnen, dass die Bewertungen mit vorausgesagten Zügen erwartungswerte für die Blattverteilung sind. Daher ist es hier auch möglich, dass die Blätter eine Fließkommazahl sind und nicht mehr nur auf natürliche Zahlen beschränkt sind. Man muss hier nur beachten, dass die Zahlen in Dezimalschreibweise ganze nicht unendlich lang sind, weil dann die Berechnung unendlich lang dauern würde.

3. Ermitteln des besten Zuges

Nachdem alle möglichen Züge bewertet wurden, wählt man nun den Zug mit bester Bewertung (also mit geringestem Wert) aus und vergleicht diese mit der Bewertung des aktuellen Hofes. Ist die Bewertung nicht besser als die bisherige Bewertung, terminiert der Greedy-Algorithmus. Ist die Bewertung besser als die bisherige Bewertung, lässt der Algorithmus die Simulation den Pusteschritt ausführen und beginnt einen neuen Greedy-Schritt mit Teilschritt 2 und 3.

```
13 12 11 10 11 12 13
12 4  3  2  3  4  12
11 3  2  1  2  3  11
10 2  1  0  1  2  10
11 3  2  1  2  3  11
12 4  3  2  3  4  12
13 12 11 10 11 12 13
```

Abb.3: Gewichtungen in der Greedy Bewertung bei einem 7*7-Schulhof

```
4 3 2 3 4
3 2 1 2 3
2 1 0 1 2
3 2 1 2 3
4 3 2 3 4
```

Abb.4: *Entfernungen* von Planquadraten zum Mittigen Feld

Die erwartete Laufzeit unseres Greedy-Ansatzes ist in der Theorie Polynomial. Wenn wir davon ausgehen, dass z(g) die Anzahl an Pustevorgängen, die der Hausmeister durchführen muss, um ein Schulhof fertig zu pusten, übersteigt unsere erwartete Laufzeit nicht $O(\mathrm{z(g)} * g^2 * a + a * g)$, da die Ermittlung aller möglichen Züge eine einmalige Laufzeit von $O(a * g) = O(a * i * j), 2 < a < 4$ braucht und der Greedy-Schritt für jeden der Berechnungen erwartet nicht langsamer als $O(a * g * g)$ ist, da diese pro Schritt jeden der $g * a$ Pustezüge mit g Feldern simulieren muss. Anhand von Simulationseingaben erkennen wir experimentell zudem, dass z(g) porpotional zu $log\ g$ ist, sodass sich insgesamt eine asymptotische erwartete Höchstlaufzeit von $O(g^2 * log\ g)$ und zusammen mit dem Simulationsprogramm eine asymptotische Höchstlaufzeit von $O(n * g^2 * log\ g)$ ergibt.

3.1.1 Greedy Ansatz mit Optimierung

Alternativ zum vorgestellten Ansatz können wir auch noch eine Optimierung betreiben, sodass der Greedy Ansatz nicht mehr versucht, alle Blätter auf ein Planquadrat zu versammeln, sondern diese nur zu einem gewünschten Zielquadrat im Schulhof zu konzentrieren. Dazu lassen wir Suche nach möglichen Zügen vor jedem Greedy-Schritt laufen und verbieten Züge, für die $S_{i,\mathrm{j}} + \vec{r}$ ein leeres Feld ist. Dadurch kommt es nicht zum intentionellen Pusten wie im Abschnitt 2.1, sodass, abgesehen vom Pusten der Blätter vom Rand, „ineffiziente“ Züge vermieden werden.

3.2 Greedy Ansatz mit Blasen von zwei Feldern Entfernung

Mit dem Greedy Ansatz haben wir eine effiziente Lösung gefunden, mit dem wir mit wenig Rechenaufwand auf gute Ergebnisse kommen (bei hinreichend großen Feldern kann man sogar alle Blätter auf ein Feld blasen, wie die Beispiele weiter unten zeigen). Allerdings kann sich bei sehr

großen Feldern die Laufzeit bis ins unermessliche steigern. Aus diesem Grund stellen wir einen dynamischen Algorithmus vor, der auch bei großen Schulhöfen gut angewandt werden kann. Hierbei muss allerdings zunächst mithilfe eines Greedy Algorithmus alle Blätter auf Felder gebracht werden, die mindestens einen Abstand von 2 zum Schulhofrand haben (Achtung: mit Abstand ist nicht die *Entfernung* zum Zielfeld gemeint. Vielmehr geht es um die Bedingung, die im Abschnitt 2.1 thematisiert wird). Die Lage des Zielfeldes ist dabei beliebig. Anschließend lässt sich anhand des Ansatzes in Abschnitt 2.1 die Blätter mit folgendem Algorithmus zusammenblasen:

1. Blase alle Blätter, die auf einem Planquadrat höheren j-Wert als das Zielfeld liegen, auf den j-Wert des Zielfelds zu und alle Blätter, die auf ein Planquadrat mit einem geringeren j-Wert als das Zielfeld liegen, ebenfalls auf den j-Wert des Zielfeldes zu.
2. Wiederhole denselben Prozess mit dem i-Wert der Blätter.

Der so erstellte Ansatz hat bei sehr großen Höfen den Vorteil, dass nach dem fertigen Bepusten des Randes durch den Greedy-Ansatz der Rest des Schulhofes vom Hausmeister sehr intuitiv bepustet werden kann. Insbesondere ist bei hinreichend großen Schulhöfen dann die erwartete asymptotische Programmlaufzeit nicht höher als $O(g * log\ n)$, wobei n die Anzahl der Blätter ist. Zusammen mit dem Simulationsprogramm ergibt sich dann eine erwartete asymptotische Laufzeit von nicht mehr als $O(z(g) * \mathrm{n} * \mathrm{g} * log\ n)$, welches in Theorie schneller ist als der Greedy-Ansatz, wobei $z(g)$ proportional zu g ist, da wir jedes Planquadrat im Schulhof nur einmal vom Laub befreien müssen. Die konkrete erwartete Laufzeit des Programms ist also nicht höher als $O(g^2 * n * log\ n)$. Im Vergleich zum Greedy Ansatz erkennt man daher, dass der Ansatz mit Blasen von zwei Feldern Entfernung daher besonders bei sehr großen Feldern effizienter ist, während der reine Greedy-Ansatz für eher kleinere Felder effizienter ist. In der Praxis muss man jedoch beachten, dass der Pustevorgang durch den Hausmeister auf den realen Schulhof deutlich länger dauert als in der Simulation, sodass tatsächlich noch bis zu sehr großen Schulhöfen die Laufzeit zur Berechnung eines Pusteschritts durch den Greedy-Algorithmus akzeptabel ist. Beachten wir jedoch sehr, sehr große Schulhöfe, die ein sehr pflichtbewusster Hausmeister dann für sehr lange bepusten möchte, so erweist sich der Algorithmus mit Blasen von zwei Feldern Entfernung als effizienter.

3.3 Erweiterung: Nichtdynamischer heuristischer Ansatz

Wenn der Hausmeister es jedoch wünscht, sehr schnell alle Felder fertig zu bepusten, können die oben genannten Ansätze trotzdem etwas unrealistisch sein, da diese hunderte von Pusteschrittenfür auch kleine Schulhöfe fordern. Aus diesem Grund stellen wir einen nichtdeterministischen Ansatz vor, welchen jedes der g Felder genau einmal bepustet und damit möglichst viele Blätter auf einem Feld konzentrieren zu versucht. Dabei beschränken wir den Ansatz darauf, dass der Hausmeister die Blätter immer nur in der Mitte des Schulhofes zu versammeln versucht.

Da unser Ansatz nichtdynamisch ist, können wir nun mit Erwartungswerten arbeiten, um die Effizienz unseres Ansatzes zu prüfen. Den erarbeiteten Algorithmus können wir dann einfach auf reale Simulationen anwenden, welche der Pustreihenfolge der Felder nach ausgeführt werden.

Da es nicht möglich ist, Blätter vom Rand durch einmalige Bepustung komplett wegzubekommen, wir jedoch auch nicht jedes Feld mehr als einmal bepusten wollen, sammeln wir alle Blätter vom Rand in eine Ecke des Schulhofes, wo der Hausmeister diese dann mit einem Sack oder einer Kiste zusammenschaufeln und zum großen Haufen in der Mitte des Schulhofes transportieren kann. Die einzelnen Teilschritte werden nun im Folgenden genauer erläutert.

1. Bepustung der Ränder

Um alle Randfelder zu bepusten und das gesamte Laub in eine Ecke zu sammeln, kommen wir leider ohne mehrmalige Bepustung der Ecken nicht komplett aus. Es seien die Eckquadrate des

Quadratischen Feldes A, B, C und D. Ausgehend von C pusten wir nun zunächst durch wiederholtes Pusten alles Laub von diesem Feld und pusten dann den Kanten entlang die Blätter der Randfelder in Richtung B bzw. C. Nun erreichen wir durch wiederholtes Pusten abermals, dass auf den Feldern B und D keine Blätter mehr liegen. Nun pusten wir nur noch alle Blätter in Richtung Eckfeld A und sind zum Schluss fertig. Eine Illustration ist hierbei der Abb. 5 zu entnehmen:
Unsere Heuristik ist bei der Bepustung des restlichen Feldes nun darauf ausgelegt, die Felder von

Abb.5: Pustereihenfolge der Randfelder. Die Pfeile geben den richtungsvektoren an, in die der Hausmeister jeweils pusten soll. Es wird dabei von unten rechts nach oben links gepustet.

innen nach außen zu bepusten, um möglichst viele Blätter in die Mitte blasen zu können. Dabei bepusten wir immer schichtweise die „Kanten“ und die „Ecken“, die am weitesten außen liegen, abwechselnd punktsymmetrisch zum mittigem Zielfeld, um so langsam alle Blätter auf einne Haufen zu konzentrieren.

2. Bepustung der „Ecken“

Eine *Ecke* eines Schulhofes sei definiert als die vier Planquadrate, welche die größte *Entfernung* zum Zielfeld haben und zeitgleich noch nicht gepustet wurden. Der Hausmeister stellt sich nun abwechselnd in eine der Ecken und pustet diese in eine feste Richtung. Durch die vorgeschriebene Punktsymmetrie, sind dadurch alle anderen Pusteschritte auch vorgeschrieben. In unserem Programm legen wir dabei fest, dass die Ecke, die dem Eckfeld C am nächsten ist, immer in Richtung $\vec{r} = C - B$ gepustet werden soll.

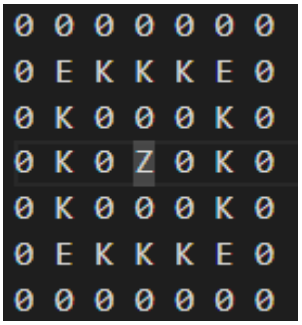

Abb.6: Rand und Eckfelder in einem Schulhof. Die Rand- und Eckfelder des Schulhofes werden hierbei als schon einmal bepustet angenommen. E = Ecke, K = Kante, Z = Zielfeld

3. Bepustung der „Kanten“

Eine *Kante* definieren wir als alle Felder, die ausschließlich durch horizontale oder vertikale Schritte von den Ecken erreicht werden können. Betrachten wir eine Reihe von Kantenfeldern, ist die Pusterichtung immer in Richtung des Zielfeldes, und die Pustereihenfolge ergibt sich von außen nach innen. Mann bemerkt insbesondere, dass bei einer Pustereihenfolge von außen nach innen die erwartete Verteilung der Blätter mehr in der Mitte konzentriert werden, was für bessere Ergebnisse spricht (siehe Abb. 7). Daher bepusten wir alle Kanten nun von außen nach innen abweselnd, wie es in der Abbildung vorgegeben ist. Die Pusteschritte durch den Hausmeister wiederholen wir analog zu Punkt 2 punktsymmetrisch zum Zielfeld, sodass nach einem Pustezyklus alle Kanten bepustet wurden.

Schritt 2 und 3 wiederholen wir nun so lange, bis alle Felder bis auf das Zielfeld einmal bepustet wurden. Dadurch beträgt die erwartete Laufzeit des Programms in jedem Fall nicht mehr als $O(g + 4 * log\ n)$, was deutlich schneller als die vorherigen zwei Ansätze ist. Die asymptotische Laufzeit beträgt dann mit dem Simualtionsprogramm zusammen höchstens $O(n * g + n * log\ n)$, da die benötigte Anzahl an Schritten asymptotisch gleich der Hofgröße g ist.

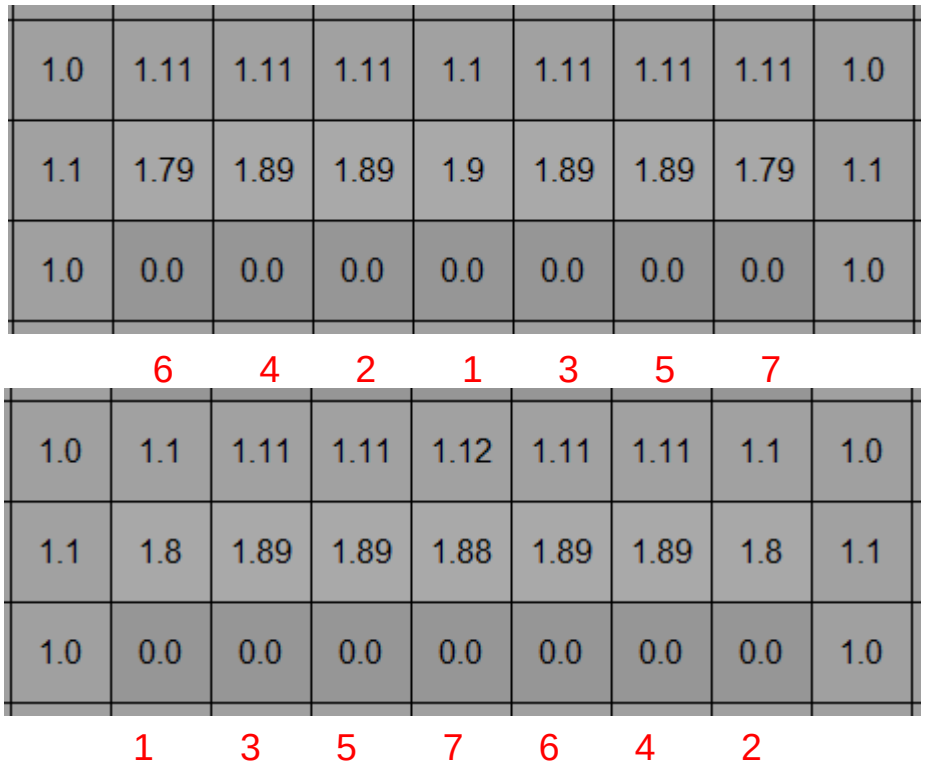

Abb. 6: Erwartete Wahrscheinlichkeiten beim Pusten von Kantenfeldern. In diesem Fall wird die unterste Reihe an Feldern betrachtet und es wird immer nach oben gepustet. Die roten Zahlen geben die Reihenfolge des Pustens an.

4. Implementierung

Für die Laubblattsimulationen, die genutzt wurden, um Abb.4 darzustellen, benötigten wir etwa 683.22 Sekunden Laufzeit, und das Programm lief auf einem Intel Core i5 2430M. Den Quellcode für diese Simulation führen wir hier nicht mehr weiter an.

Bezüglich der restlichen Algorithmen sind neben den oben genannten Implementationsdetails zu den einzelnen Algorithmen sind im Quellcode auch nochmal die Funktionen und Details der einzelnen Programmabschnitte dokumentiert. Für die Laubblattsimulation, welche als Funktion direkt in den einzelnen Programmen mit enthalten ist, nutzen wir zum Generieren der Pseudo-Zufallszahlen die random-Bibliothek in Python. Zudem nutzen wir die Numpy-Biblihothek in Python, um insbesondere bei Laubblattsimulationen und Pustevorgängen schnell Vektoroperationen durchführen zu können.

5. Beispiele

Im Folgenden untersuchen wir verschiedene Beispiele, welche unter Veränderung von Parametern

wie Länge des Hofes, Breite des Hofes und Laubanzahl sowie Position des Zielquadrates unterschiedliche Ergebnisse aufweisen. Wenn das Zielquadrat sich genau mittig auf dem Schulhof befindet und der Schulhof eine ungerade quadratische Kantenlänge hat, kann die Heuristik auch für solche Fälle entsprechende Blasevorgänge errechnen. Wichtig ist hier nochmal anzumerken, dass die Laufzeiten in Gegenwart von Zufallsvariablen ermittelt wurden und somit eventuell abweichen. Da diese aber in der Größenordnung diese aber stimmen, lassen sich daraus auch Rückschlüsse über die oben genannten Theorien der Laufzeit schließen.

Die betrachteten Ansätze sind die folgenden:

G1 unser Greedy Algorithmus mit eingebauter Bewertung

G2 unsere optimierte Greedy-Variante

ZF+G unsere zweite Kombination, in der alle gültigen Pfade optimiert werden

Heur Unsere Heuristik

Laufende Nummer	**i**	**j**	**r**	**s**	**g = i*j**	**n**	**G1**	**G2**	**ZF+G**	**Heur.**
1 (mittiges Feld)	5	5	2	2	25	100	29.64818 Score: 664	27.36031 Score: 630	22.5269	0.56880
2	6	6	2	2	35	100	65.52072 Score: 413	44.36264 Score: 549	45.5676	/
3 (mittiges Feld)	7	7	3	3	49	100	114.58766 Score: 0	74.74762 Score: 302	81.5640	0.48832
4 (mittiges Feld)	9	9	4	4	81	100	561.93707 Score: 0	164.81192 Score:641	228.5535	0.37187
5	6	11	2	3	66	100	220.72360 Score: 242	115.18400 Score: 473	116.4798	/
6	10	5	2	2	45	100	169.60460 Score:959	93.26955 Score:1015	129.7127	/
7	5	5	2	2	25	10.000	55.97033 Score: 51832	69.60069 Score: 52998	57.93715	5.21345
8 (mittiges Feld)	13	13	6	6	169	10	Sehr lange	Sehr lange	719.2269	1.15065
9(mittiges Feld)	30	30	14	14	9000	100	Sehr lange	Sehr lange	Sehr lange	3.98672

Bemerkungen:

1. Die Programme liefen dabei auf einem Kern eines Intel i5 2430M.
2. Bei hinreichend großen Feldern (ab $i \leq 7, j \leq 7$) liefern dabei G1 sowie ZF+G immer perfekte Ergebnisse.
3. s und t stellen die Indizien des Laubfeldes dar, sodass diese nicht den

wirklichen Koordinaten entsprechen. $(s,t) = (2,2)$ beschreibt beispielsweise ein Feld in der dritten Spalte, dritte Reihe

Anhand der Tabelle erkennen wir, dass die optimierte Greedy-version zwar vor allem bei großen eingaben schneller ist als der originale Greedy-Ansatz G1,

Im Folgenden geben wir nun die Blasesequenzen des Hausmeisters für die Quadratischen Felder aus, welche anhand von unserem (originalen) Greedy-Algorithmus ermittelt wurden. Anhand dieser ausgewählten Beispiele erkennt man jedoch bereits die Funktionalität des Programms und

Nummer 1:

count: 664
eval: 0

**[[[0, 1], [0, -1]], [[0, 3], [0, 1]], [[3, 0], [1, 0]], [[3, 4], [1, 0]], [[3, 4], [1, 0]], [[3, 0], [1, 0]], [[0, 1], [0, -1]], [[0, 3], [0, 1]], [[3, 4], [1,
0]], [[3, 0], [1, 0]], [[0, 1], [0, -1]], [[0, 3], [0, 1]], [[0, 2], [1, 0]], [[2, 0], [0, 1]], [[1, 2], [1, 0]], [[3, 0], [0, 1]], [[1, 0], [0, 1]], [[0, 2], [1,
0]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[4, 0], [0, 1]], [[0, 4], [1, 0]], [[3, 4],
[-1, 0]], [[0, 4], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [0, 1]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[4, 4], [0, -1]], [[4,
1], [0, 1]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[3, 4], [-1, 0]], [[0, 4], [1, 0]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[3, 0], [1, 0]], [[3, 0], [0, 1]],
[[3, 4], [-1, 0]], [[0, 4], [1, 0]], [[0, 1], [0, -1]], [[3, 0], [0, 1]], [[0, 3], [1, 0]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[3, 0], [0, 1]], [[3, 4], [-1,
0]], [[0, 4], [1, 0]], [[3, 0], [0, 1]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4],
[-1, 0]], [[1, 4], [1, 0]], [[0, 3], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[3, 4], [1, 0]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[3, 0], [0, 1]], [[0,
0], [1, 0]], [[2, 0], [0, 1]], [[0, 3], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[3, 0], [0, 1]], [[0, 3], [0, 1]], [[3, 0], [1, 0]], [[0, 3], [1, 0]], [[3,
0], [0, 1]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[0, 3], [1, 0]], [[3, 0], [0, 1]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[0, 3], [1, 0]], [[3, 0], [0, 1]],
[[3, 0], [-1, 0]], [[0, 0], [1, 0]], [[2, 0], [0, 1]], [[1, 2], [1, 0]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[0, 3], [1, 0]], [[0, 1], [0, -
1]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[3, 0], [0, 1]], [[1, 0], [0, 1]], [[0, 2], [1, 0]], [[4, 4], [-1, 0]], [[1, 4],
[1, 0]], [[0, 3], [1, 0]], [[3, 0], [0, 1]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[2,
0], [0, 1]], [[3, 0], [0, 1]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[2, 0], [0, 1]], [[3, 0], [0, 1]], [[4, 0], [-1, 0]],
[[1, 0], [1, 0]], [[3, 0], [0, 1]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[2, 0], [0, 1]], [[3, 0], [0, 1]], [[0, 2], [1, 0]], [[3, 0], [0, 1]], [[0, 3], [1, 0]],
[[0, 3], [1, 0]], [[2, 1], [0, 1]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[3, 0], [0, 1]], [[0, 3], [1, 0]], [[0, 0], [0, 1]], [[0, 4], [0, -1]], [[0, 1], [0, 1]],
[[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 2], [1, 0]], [[0, 3], [1, 0]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 3], [1,
0]], [[0, 2], [1, 0]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 3], [1, 0]], [[0, 4], [0, -1]], [[0, 2], [1, 0]], [[0, 1], [0, 1]], [[0, 3], [1, 0]], [[2, 1],
[0, 1]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[3, 0], [1, 0]], [[3, 4], [1, 0]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 3], [1, 0]], [[4, 4],
[-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[0, 3], [1, 0]], [[4,
4], [0, -1]], [[4, 1], [0, 1]], [[3, 0], [0, 1]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[3, 0], [0, 1]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[0, 3], [1, 0]],
[[4, 4], [0, -1]], [[4, 1], [0, 1]], [[0, 3], [1, 0]], [[3, 0], [0, 1]], [[2, 0], [0, 1]], [[0, 2], [1, 0]], [[3, 0], [0, 1]], [[0, 3], [1, 0]], [[3, 4], [1, 0]],
[[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 3], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[3, 0], [0, 1]], [[4, 0], [-1,
0]], [[1, 0], [1, 0]], [[3, 0], [0, 1]], [[0, 3], [1, 0]], [[2, 0], [0, 1]], [[0, 2], [1, 0]], [[3, 0], [0, 1]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[0, 3], [1,
0]], [[3, 0], [0, 1]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 3], [1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[3, 0],
[0, 1]], [[0, 3], [1, 0]], [[4, 0], [0, 1]], [[3, 0], [0, 1]], [[0, 1], [0, -1]], [[0, 3], [0, 1]], [[0, 3], [1, 0]], [[0, 4], [1, 0]], [[4, 0], [-1, 0]], [[2,
0], [0, 1]], [[1, 0], [1, 0]], [[3, 0], [0, 1]], [[0, 3], [1, 0]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 2], [1, 0]], [[2, 1], [0, 1]], [[0, 3], [1, 0]], [[0,
3], [1, 0]], [[0, 3], [1, 0]], [[3, 4], [-1, 0]], [[0, 3], [1, 0]], [[0, 4], [1, 0]], [[0, 3], [1, 0]], [[3, 0], [0, 1]], [[4, 4], [0, -1]], [[4, 1], [0, 1]],
[[4, 4], [0, -1]], [[4, 1], [0, 1]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[3, 0], [0,
1]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[0, 3], [1, 0]], [[4, 4], [-1, 0]], [[1, 4],
[1, 0]], [[0, 3], [1, 0]], [[3, 0], [0, 1]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 3], [1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[2, 0], [0, 1]], [[3,
0], [0, 1]], [[0, 2], [1, 0]], [[0, 3], [1, 0]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[0, 3], [1, 0]], [[3, 0], [0, 1]], [[0, 4], [0, -1]], [[0, 1], [0, 1]],
[[0, 3], [1, 0]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[0, 3], [1, 0]], [[3, 0], [0, 1]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[0, 3], [1, 0]], [[4, 4], [0, -
1]], [[4, 1], [0, 1]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[2, 0], [0, 1]], [[3, 0], [0, 1]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 1], [0, -1]], [[0, 1],
[0, -1]], [[0, 0], [1, 0]], [[0, 3], [0, 1]], [[0, 3], [0, 1]], [[3, 0], [1, 0]], [[3, 0], [0, 1]], [[0, 3], [1, 0]], [[0, 4], [0, -1]], [[0, 2], [1, 0]], [[0,
1], [0, 1]], [[0, 3], [1, 0]], [[3, 0], [0, 1]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[2, 0], [0, 1]], [[3, 0], [0, 1]],
[[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[2, 0], [0, 1]], [[1, 2], [1, 0]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0,
1]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[3, 0], [0, 1]], [[0, 3], [1, 0]], [[4, 4], [-1, 0]], [[0, 3], [1, 0]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[4, 4],
[0, -1]], [[4, 1], [0, 1]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[3, 0], [0, 1]], [[0, 3], [1, 0]], [[3, 0], [0, 1]], [[4, 0], [-1, 0]], [[2, 0], [0, 1]], [[1,
0], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[3, 0], [0, 1]], [[0, 3], [1, 0]], [[4, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[4, 4], [0, -1]], [[4,
1], [0, 1]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[3, 0], [0, 1]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[1, 4], [1, 0]],
[[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 2], [1, 0]], [[0, 3], [1, 0]], [[3, 0], [0, 1]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[3, 0], [0, 1]], [[3, 0], [0,
1]], [[3, 0], [0, 1]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[4, 0], [-1, 0]], [[1, 0],
[1, 0]], [[3, 0], [0, 1]], [[0, 3], [1, 0]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 3], [1, 0]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 2], [1, 0]], [[4,
0], [-1, 0]], [[1, 0], [1, 0]], [[3, 0], [0, 1]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[4, 4], [0, -1]], [[3, 0], [0, 1]], [[0, 3], [1, 0]],
[[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[3, 0], [0, 1]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[1, 4], [1, 0]], [[3, 0], [0,
1]], [[4, 1], [0, 1]], [[2, 0], [0, 1]], [[0, 3], [1, 0]], [[2, 1], [0, 1]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 0], [0, 1]], [[3, 0], [0,
1]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[3, 0], [0, 1]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-
1, 0]], [[1, 4], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[4, 4], [0, -1]], [[3,
0], [0, 1]], [[3, 0], [0, 1]], [[4, 1], [0, 1]], [[3, 0], [0, 1]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[3, 0], [0, 1]], [[0, 4], [0, -1]], [[0, 1], [0, 1]],**

**[[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 3], [1, 0]], [[0, 4], [0, -1]], [[0, 1], [0,
1]], [[0, 2], [1, 0]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 2], [1, 0]], [[0, 4], [0, -1]], [[0, 2], [1, 0]], [[1, 2],
[1, 0]], [[3, 0], [0, 1]], [[0, 1], [0, 1]], [[0, 3], [1, 0]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[4, 4], [0, -1]], [[4, 1],
[0, 1]], [[3, 0], [0, 1]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[0, 3], [1, 0]], [[3, 0], [0, 1]], [[0, 3], [1, 0]], [[2,
0], [0, 1]], [[3, 0], [0, 1]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[2, 0], [0, 1]], [[1, 0], [1, 0]], [[3, 0], [0, 1]], [[0, 4], [0, -1]],
[[0, 2], [1, 0]], [[0, 1], [0, 1]], [[0, 3], [1, 0]], [[3, 0], [0, 1]], [[4, 4], [-1, 0]], [[0, 3], [1, 0]], [[4, 4], [0, -1]], [[3, 0], [0, 1]], [[0, 3], [1,
0]], [[3, 0], [0, 1]], [[0, 0], [1, 0]], [[0, 3], [1, 0]], [[2, 1], [0, 1]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1,
0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 4], [0, -1]], [[0, 2], [1, 0]], [[0, 1], [0, 1]], [[0, 3], [1, 0]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0,
1]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[2, 0], [0, 1]], [[1, 2], [1, 0]], [[3, 0], [0, 1]], [[4, 1],
[0, 1]], [[1, 0], [1, 0]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3,
0], [0, 1]], [[3, 0], [0, 1]], [[1, 4], [1, 0]], [[3, 0], [0, 1]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[3, 0], [0, 1]], [[4,
0], [0, 1]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]],
[[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1,
0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [0, -1]], [[4, 1], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[4, 4], [0, -1]], [[3, 0],
[0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[3, 0], [0, 1]], [[0, 0], [0, 1]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 3],
[1, 0]], [[1, 2], [1, 0]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[1,
0], [1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[3, 0], [0, 1]], [[0, 3], [1, 0]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[0, 1], [0, 1]],
[[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 3], [1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1,
0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4],
[1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4,
4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]],
[[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 4], [-1, 0]], [[1, 4], [1,
0]], [[3, 0], [0, 1]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1,
0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1,
0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0,
1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]]]**

```
[[   0    0    0    0    0]
 [   0    1  228    1    0]
 [   0   62 1853  206    0]
 [   0    6  137    6    0]
 [   0    0    0    0    0]]
```

Nummer 2:

count: 665
eval: 413

**[[[0, 1], [0, -1]], [[0, 4], [0, 1]], [[4, 0], [1, 0]], [[4, 5], [1, 0]], [[4, 0], [1, 0]], [[4, 5], [1, 0]], [[0, 4], [0, 1]], [[0, 1], [0, -1]], [[4, 0], [1, 0]],
[[0, 1], [0, -1]], [[4, 5], [1, 0]], [[0, 4], [0, 1]], [[3, 5], [0, -1]], [[3, 4], [0, -1]], [[4, 2], [-1, 0]], [[3, 0], [0, 1]], [[5, 3], [-1, 0]], [[5, 2], [-1, 0]],
[[4, 2], [-1, 0]], [[3, 4], [0, -1]], [[4, 0], [1, 0]], [[2, 5], [0, -1]], [[2, 4], [0, -1]], [[0, 3], [1, 0]], [[2, 0], [0, 1]], [[0, 0], [1, 0]], [[1, 0], [1, 0]],
[[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[3, 0], [0, 1]], [[4, 2], [-1, 0]],
[[0, 2], [1, 0]], [[5, 4], [-1, 0]], [[3, 5], [0, -1]], [[3, 4], [0, -1]], [[0, 1], [0, -1]], [[4, 5], [1, 0]], [[0, 5], [1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]],
[[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[2, 5], [0, -1]], [[2, 4], [0, -1]], [[2, 0], [0, 1]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[3, 5], [0, -1]], [[5, 0], [0, 1]],
[[5, 1], [0, 1]], [[5, 4], [0, -1]], [[5, 2], [-1, 0]], [[4, 2], [-1, 0]], [[5, 1], [0, 1]], [[5, 4], [0, -1]], [[5, 3], [-1, 0]], [[3, 4], [0, -1]], [[5, 1], [-1,
0]], [[3, 0], [0, 1]], [[1, 5], [0, -1]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[5, 2], [-1, 0]], [[5, 1], [0, 1]], [[5, 4], [0, -1]], [[0, 3], [1, 0]], [[4, 0], [1,
0]], [[5, 5], [0, -1]], [[5, 4], [0, -1]], [[5, 1], [0, 1]], [[5, 4], [0, -1]], [[5, 3], [-1, 0]], [[4, 3], [-1, 0]], [[2, 4], [0, -1]], [[5, 1], [0, 1]], [[5, 4], [0,
-1]], [[5, 2], [-1, 0]], [[4, 2], [-1, 0]], [[0, 2], [1, 0]], [[5, 1], [0, 1]], [[5, 4], [0, -1]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[1, 0], [0, 1]], [[1, 5], [1,
0]], [[4, 5], [-1, 0]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[0, 0], [0, 1]], [[0, 1], [0, 1]], [[0, 4], [0, -
1]], [[0, 1], [0, 1]], [[0, 3], [1, 0]], [[0, 4], [0, -1]], [[2, 5], [0, -1]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[5, 1], [0,
1]], [[5, 4], [0, -1]], [[5, 3], [-1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[3, 5], [0, -1]], [[3, 4], [0, -1]], [[5, 2], [-1, 0]], [[3, 0], [0, 1]], [[0, 4], [0,
1]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[4, 5], [1, 0]], [[5, 1], [0, 1]], [[5, 4], [0, -1]], [[5, 0], [-1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1,
0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[2, 0], [0, 1]], [[2, 4], [0, -1]], [[4, 2], [-1, 0]], [[0, 2], [1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[3, 0], [0,
1]], [[4, 3], [-1, 0]], [[5, 2], [-1, 0]], [[4, 0], [1, 0]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[0, 3], [1, 0]], [[0, 5], [0, -1]], [[0, 4], [0, -1]], [[0, 1], [0,
1]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[2, 5], [0, -1]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[1, 0], [1,
0]], [[4, 0], [-1, 0]], [[4, 5], [1, 0]], [[5, 1], [0, 1]], [[5, 4], [0, -1]], [[5, 3], [-1, 0]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[0, 3], [1, 0]], [[5, 5], [-1,
0]], [[4, 5], [-1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[2, 5], [0, -1]], [[3, 5], [0, -1]], [[4, 3], [-1, 0]], [[1, 5], [1,
0]], [[4, 5], [-1, 0]], [[2, 5], [0, -1]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[2, 5], [0, -1]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[5, 2], [-1, 0]], [[0, 1], [0,
1]], [[0, 4], [0, -1]], [[5, 1], [0, 1]], [[5, 4], [0, -1]], [[1, 5], [1, 0]], [[3, 5], [0, -1]], [[4, 5], [-1, 0]], [[2, 5], [0, -1]], [[5, 2], [-1, 0]], [[0, 1], [0,
-1]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[5, 1], [0, 1]], [[5, 3], [-1, 0]], [[3, 4], [0, -1]], [[3, 0], [0, 1]], [[5, 2], [-1, 0]], [[5, 4], [0, -1]], [[5, 2], [-
1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[2, 5], [0, -1]], [[0, 3], [1, 0]], [[3, 4], [0, -1]], [[2, 5], [0, -1]], [[5, 2], [-1, 0]], [[0, 1], [0, 1]], [[0, 4],
[0, -1]], [[2, 5], [0, -1]], [[5, 2], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[0, 1], [0, -1]], [[4, 0], [1, 0]], [[4, 5], [1, 0]], [[2, 5], [0, -1]], [[0, 1],
[0, 1]], [[0, 4], [0, -1]], [[5, 1], [0, 1]], [[5, 4], [0, -1]], [[5, 2], [-1, 0]], [[2, 5], [0, -1]], [[1, 5], [1, 0]], [[3, 5], [0, -1]], [[4, 5], [-1, 0]], [[2, 5],
[0, -1]], [[5, 2], [-1, 0]], [[2, 5], [0, -1]], [[1, 0], [1, 0]], [[3, 0], [0, 1]], [[4, 3], [-1, 0]], [[2, 5], [0, -1]], [[0, 3], [1, 0]], [[2, 5], [0, -1]], [[5, 2],
[-1, 0]], [[4, 0], [-1, 0]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[5, 2], [-1, 0]], [[5, 1], [0, 1]], [[5, 4], [0, -1]], [[5, 2], [-1, 0]], [[5, 3], [-1, 0]], [[2,
5], [0, -1]], [[0, 0], [1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[0, 1], [0, 1]], [[0,
4], [0, -1]], [[0, 1], [0, -1]], [[0, 4], [0, 1]], [[2, 5], [0, -1]], [[4, 0], [1, 0]], [[5, 0], [0, 1]], [[5, 2], [-1, 0]], [[5, 1], [0, 1]], [[5, 4], [0, -1]], [[5,**

2], [-1, 0]], [[5, 1], [0, 1]], [[5, 4], [0, -1]], [[5, 1], [0, 1]], [[5, 4], [0, -1]], [[5, 2], [-1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[2, 5], [0, -1]], [[5, 3], [-1, 0]], [[3, 4], [0, -1]], [[5, 2], [-1, 0]], [[3, 0], [0, 1]], [[5, 2], [-1, 0]], [[2, 5], [0, -1]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[5, 2], [-1, 0]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[2, 5], [0, -1]], [[5, 1], [0, 1]], [[5, 4], [0, -1]], [[5, 2], [-1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[2, 5], [0, -1]], [[3, 5], [0, -1]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[5, 2], [-1, 0]], [[0, 5], [1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[2, 5], [0, -1]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[2, 5], [0, -1]], [[4, 3], [-1, 0]], [[0, 3], [1, 0]], [[2, 5], [0, -1]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[3, 5], [0, -1]], [[2, 5], [0, -1]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[2, 5], [0, -1]], [[5, 1], [0, 1]], [[5, 4], [0, -1]], [[5, 3], [-1, 0]], [[5, 2], [-1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[2, 5], [0, -1]], [[4, 3], [-1, 0]], [[2, 5], [0, -1]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[5, 1], [0, 1]], [[5, 4], [0, -1]], [[5, 2], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[2, 5], [0, -1]], [[0, 0], [0, 1]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[2, 5], [0, -1]], [[5, 2], [-1, 0]], [[3, 0], [0, 1]], [[5, 2], [-1, 0]], [[2, 5], [0, -1]], [[5, 2], [-1, 0]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[2, 5], [0, -1]], [[5, 1], [0, 1]], [[5, 4], [0, -1]], [[2, 5], [0, -1]], [[4, 3], [-1, 0]], [[2, 5], [0, -1]], [[5, 1], [0, 1]], [[5, 4], [0, -1]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[1, 5], [1, 0]], [[3, 5], [0, -1]], [[4, 5], [-1, 0]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[5, 2], [-1, 0]], [[2, 5], [0, -1]], [[5, 2], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[1, 5], [1, 0]], [[3, 5], [0, -1]], [[4, 3], [-1, 0]], [[0, 3], [1, 0]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[5, 1], [0, 1]], [[5, 4], [0, -1]], [[5, 2], [-1, 0]], [[3, 0], [0, 1]], [[5, 2], [-1, 0]], [[5, 1], [0, 1]], [[5, 4], [0, -1]], [[5, 1], [0, 1]], [[5, 4], [0, -1]], [[5, 2], [-1, 0]], [[5, 3], [-1, 0]], [[3, 4], [0, -1]], [[3, 0], [0, 1]], [[5, 2], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[2, 5], [0, -1]], [[4, 5], [-1, 0]], [[2, 5], [0, -1]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[0, 1], [0, 1]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[5, 2], [-1, 0]], [[0, 4], [0, -1]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[4, 0], [-1, 0]], [[5, 1], [0, 1]], [[5, 3], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[2, 5], [0, -1]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[1, 5], [1, 0]], [[3, 5], [0, -1]], [[3, 4], [0, -1]], [[5, 2], [-1, 0]], [[2, 5], [0, -1]], [[4, 5], [-1, 0]], [[4, 5], [4, 0, -1]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[3, 0], [0, 1]], [[5, 2], [-1, 0]], [[5, 3], [-1, 0]], [[0, 0], [1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[2, 5], [0, -1]], [[4, 3], [-1, 0]], [[2, 5], [0, -1]], [[3, 0], [0, 1]], [[5, 2], [-1, 0]], [[4, 0], [-1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[1, 5], [1, 0]], [[3, 5], [0, -1]], [[5, 2], [-1, 0]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[0, 1], [0, 1]], [[0, 4], [0, -1]], [[1, 0], [1, 0]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[4, 5], [-1, 0]], [[5, 0], [-1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[3, 0], [0, 1]], [[4, 0], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[5, 2], [-1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[5, 1], [0, 1]], [[5, 4], [0, -1]], [[5, 1], [0, 1]], [[5, 4], [0, -1]], [[5, 1], [0, 1]], [[5, 3], [-1, 0]], [[3, 5], [0, -1]], [[3, 4], [0, -1]], [[3, 0], [0, 1]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[0, 3], [1, 0]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[5, 2], [-1, 0]], [[5, 0], [0, 1]], [[0, 1], [0, 1]], [[0, 3], [1, 0]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[4, 0], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 4], [0, -1]], [[5, 1], [0, 1]], [[5, 4], [0, -1]], [[5, 2], [-1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[1, 5], [1, 0]], [[4, 5], [-1, 0]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[5, 1], [0, 1]], [[5, 4], [0, -1]], [[5, 1], [0, 1]], [[5, 4], [0, -1]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5,], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[2, 5], [0, -1]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]], [[5, 2], [-1, 0]]]

```
[[   0    0    0    0    0    0]
 [   0    0  103    0    0    0]
 [   0  310 3187    0    0    0]
 [   0    0    0    0    0    0]
 [   0    0    0    0    0    0]
 [   0    0    0    0    0    0]]
```

Nummer 3:

count: 824
eval: 0
[[[0, 1], [0, -1]], [[0, 5], [0, 1]], [[5, 0], [1, 0]], [[5, 6], [1, 0]], [[0, 1], [0, -1]], [[5, 0], [1, 0]], [[0, 5], [0, 1]], [[5, 6], [1, 0]], [[0, 1], [0, -1]], [[5, 0], [1, 0]], [[0, 5], [0, 1]], [[5, 6], [1, 0]], [[5, 0], [1, 0]], [[0, 2], [1, 0]], [[1, 2], [1, 0]], [[3, 1], [0, 1]], [[2, 0], [0, 1]], [[4, 0], [0, 1]], [[5, 2], [-1, 0]], [[3, 0], [0, 1]], [[3, 1], [0, 1]], [[6, 2], [-1, 0]], [[0, 4], [1, 0]], [[1, 4], [1, 0]], [[3, 5], [0, -1]], [[5, 4], [-1, 0]], [[3, 6], [0, -1]], [[2, 6], [0, -1]], [[0, 1], [0, -1]], [[1, 2], [1, 0]], [[4, 6], [0, -1]], [[6, 4], [-1, 0]], [[4, 5], [0, -1]], [[5, 3], [-1, 0]], [[1, 3], [1, 0]], [[3, 5], [0, -1]], [[4, 1], [0, 1]], [[6, 3], [-1, 0]], [[5, 3], [-1, 0]], [[3, 1], [0, 1]], [[5, 6], [1, 0]], [[0, 3], [1, 0]], [[1, 4], [1, 0]], [[0, 0], [1, 0]], [[1, 0], [1, 0]], [[2, 0], [1, 0]], [[5, 0], [-1, 0]], [[2, 0], [1, 0]], [[5, 0], [-1, 0]], [[2, 0], [1, 0]], [[5, 0], [-1, 0]], [[3, 0], [0, 1]], [[4, 0], [0, 1]], [[4, 1], [0, 1]], [[2, 0], [1, 0]], [[5, 0], [-1, 0]], [[6, 6], [0, -1]], [[6, 5], [0, -1]], [[6, 2], [0, 1]], [[6, 5], [0, -1]], [[6, 2], [0, 1]], [[6,

5], [0, -1]], [[6, 3], [-1, 0]], [[6, 2], [0, 1]], [[6, 4], [-1, 0]], [[4, 5], [0, -1]], [[6, 5], [0, -1]], [[5, 3], [-1, 0]], [[1, 3], [1, 0]], [[0, 5], [0, 1]], [[0, 6], [1, 0]], [[1, 6], [1, 0]], [[2, 6], [1, 0]], [[5, 6], [-1, 0]], [[2, 6], [1, 0]], [[5, 6], [-1, 0]], [[2, 6], [1, 0]], [[5, 6], [-1, 0]], [[3, 6], [0, -1]], [[3, 5], [0, -1]], [[3, 1], [0, 1]], [[4, 6], [0, -1]], [[2, 6], [1, 0]], [[5, 6], [-1, 0]], [[2, 6], [1, 0]], [[5, 6], [-1, 0]], [[5, 4], [-1, 0]], [[6, 0], [0, 1]], [[6, 1], [0, 1]], [[6, 2], [0, 1]], [[6, 5], [0, -1]], [[6, 2], [0, 1]], [[6, 5], [0, -1]], [[6, 3], [-1, 0]], [[6, 2], [0, 1]], [[6, 4], [-1, 0]], [[6, 5], [0, -1]], [[6, 2], [0, 1]], [[6, 5], [0, -1]], [[6, 5], [-1, 0]], [[4, 6], [0, -1]], [[5, 4], [-1, 0]], [[3, 5], [0, -1]], [[5, 3], [-1, 0]], [[6, 2], [0, 1]], [[6, 5], [0, -1]], [[6, 1], [-1, 0]], [[4, 0], [0, 1]], [[4, 1], [0, 1]], [[4, 6], [-1, 0]], [[1, 6], [1, 0]], [[0, 5], [1, 0]], [[2, 6], [0, -1]], [[1, 4], [1, 0]], [[3, 6], [0, -1]], [[0, 1], [1, 0]], [[2, 0], [0, 1]], [[1, 2], [1, 0]], [[3, 0], [0, 1]], [[2, 0], [1, 0]], [[5, 0], [-1, 0]], [[6, 3], [-1, 0]], [[5, 0], [1, 0]], [[6, 2], [0, 1]], [[6, 5], [0, -1]], [[3, 1], [0, 1]], [[3, 5], [0, -1]], [[1, 3], [1, 0]], [[5, 3], [-1, 0]], [[2, 6], [1, 0]], [[5, 6], [-1, 0]], [[2, 0], [1, 0]], [[5, 0], [-1, 0]], [[3, 0], [0, 1]], [[6, 4], [-1, 0]], [[4, 5], [0, -1]], [[6, 2], [0, 1]], [[6, 5], [0, -1]], [[0, 1], [0, 1]], [[0, 2], [0, 1]], [[0, 5], [0, -1]], [[0, 2], [0, 1]], [[0, 5], [0, -1]], [[0, 2], [0, 1]], [[0, 4], [1, 0]], [[0, 5], [0, -1]], [[0, 3], [1, 0]], [[1, 4], [1, 0]], [[0, 2], [0, 1]], [[0, 5], [0, -1]], [[5, 6], [1, 0]], [[2, 6], [1, 0]], [[5, 6], [-1, 0]], [[0, 2], [0, 1]], [[0, 5], [0, -1]], [[0, 3], [1, 0]], [[6, 6], [-1, 0]], [[5, 6], [-1, 0]], [[2, 6], [1, 0]], [[5, 6], [-1, 0]], [[3, 6], [0, -1]], [[4, 6], [-1, 0]], [[1, 6], [1, 0]], [[2, 6], [1, 0]], [[4, 6], [0, -1]], [[5, 6], [-1, 0]], [[4, 5], [0, -1]], [[2, 6], [1, 0]], [[5, 6], [-1, 0]], [[3, 6], [0, -1]], [[6, 0], [-1, 0]], [[5, 0], [-1, 0]], [[2, 0], [1, 0]], [[5, 0], [-1, 0]], [[3, 0], [0, 1]], [[4, 0], [0, 1]], [[4, 1], [0, 1]], [[4, 0], [-1, 0]], [[1, 0], [1, 0]], [[6, 3], [-1, 0]], [[2, 0], [1, 0]], [[5, 0], [-1, 0]], [[3, 0], [0, 1]], [[2, 0], [1, 0]], [[5, 0], [-1, 0]], [[3, 0], [0, 1]], [[0, 6], [0, -1]], [[0, 5], [0, -1]], [[0, 2], [0, 1]], [[0, 5], [0, -1]], [[0, 4], [1, 0]], [[0, 2], [0, 1]], [[0, 5], [0, -1]], [[0, 3], [1, 0]], [[0, 2], [0, 1]], [[0, 5], [0, -1]], [[0, 0], [0, 1]], [[0, 1], [0, 1]], [[0, 2], [0, 1]], [[0, 5], [0, -1]], [[0, 3], [1, 0]], [[0, 4], [1, 0]], [[1, 4], [1, 0]], [[0, 2], [0, 1]], [[0, 5], [0, -1]], [[0, 2], [0, 1]], [[0, 5], [0, -1]], [[0, 3], [1, 0]], [[0, 2], [0, 1]], [[6, 2], [0, 1]], [[6, 5], [0, -1]], [[3, 6], [0, -1]], [[0, 3], [1, 0]], [[0, 2], [1, 0]], [[0, 4], [1, 0]], [[4, 6], [-1, 0]], [[1, 6], [1, 0]], [[3, 6], [0, -1]], [[4, 5], [0, -1]], [[6, 2], [0, 1]], [[6, 5], [0, -1]], [[0, 1], [0, -1]], [[6, 2], [0, 1]], [[6, 3], [-1, 0]], [[4, 6], [-1, 0]], [[5, 4], [-1, 0]], [[3, 6], [0, -1]], [[6, 3], [-1, 0]], [[0, 3], [1, 0]], [[3, 0], [0, 1]], [[2, 0], [1, 0]], [[5, 0], [-1, 0]], [[3, 0], [0, 1]], [[0, 2], [0, 1]], [[0, 4], [1, 0]], [[2, 5], [0, -1]], [[1, 2], [1, 0]], [[3, 0], [0, 1]], [[0, 3], [1, 0]], [[0, 5], [0, -1]], [[0, 3], [1, 0]], [[6, 3], [-1, 0]], [[0, 5], [0, 1]], [[5, 0], [1, 0]], [[5, 6], [1, 0]], [[3, 6], [0, -1]], [[4, 6], [-1, 0]], [[1, 6], [1, 0]], [[3, 6], [0, -1]], [[6, 2], [0, 1]], [[6, 5], [0, -1]], [[6, 3], [-1, 0]], [[3, 0], [0, 1]], [[4, 0], [0, 1]], [[3, 0], [0, 1]], [[5, 2], [-1, 0]], [[3, 0], [0, 1]], [[2, 0], [0, 1]], [[3, 6], [0, -1]], [[6, 3], [-1, 0]], [[2, 0], [1, 0]], [[5, 0], [-1, 0]], [[0, 3], [1, 0]], [[3, 0], [0, 1]], [[6, 2], [0, 1]], [[6, 5], [0, -1]], [[6, 3], [-1, 0]], [[2, 1], [0, 1]], [[0, 3], [1, 0]], [[0, 5], [0, 1]], [[2, 0], [1, 0]], [[5, 0], [-1, 0]], [[3, 0], [0, 1]], [[5, 6], [1, 0]], [[5, 6], [1, 0]], [[4, 6], [-1, 0]], [[1, 6], [1, 0]], [[3, 6], [0, -1]], [[3, 6], [0, -1]], [[0, 2], [0, 1]], [[0, 5], [0, -1]], [[0, 3], [1, 0]], [[3, 0], [0, 1]], [[3, 6], [0, -1]], [[6, 3], [-1, 0]], [[2, 0], [1, 0]], [[5, 0], [-1, 0]], [[3, 0], [0, 1]], [[6, 2], [0, 1]], [[6, 5], [0, -1]], [[6, 3], [-1, 0]], [[3, 6], [0, -1]], [[2, 6], [0, -1]], [[1, 4], [1, 0]], [[4, 5], [0, -1]], [[6, 3], [-1, 0]], [[3, 6], [0, -1]], [[6, 4], [0, -1]], [[4, 0], [0, 1]], [[6, 3], [-1, 0]], [[4, 1], [0, 1]], [[6, 3], [-1, 0]], [[0, 3], [1, 0]], [[2, 0], [1, 0]], [[5, 0], [-1, 0]], [[3, 0], [0, 1]], [[6, 2], [0, 1]], [[6, 5], [0, -1]], [[6, 3], [-1, 0]], [[4, 6], [-1, 0]], [[1, 6], [1, 0]], [[3, 6], [0, -1]], [[3, 0], [0, 1]], [[0, 3], [1, 0]], [[6, 3], [-1, 0]], [[2, 0], [1, 0]], [[5, 0], [-1, 0]], [[3, 0], [0, 1]], [[4, 0], [0, 1]], [[6, 2], [-1, 0]], [[5, 2], [-1, 0]], [[3, 0], [0, 1]], [[3, 6], [0, -1]], [[3, 0], [0, 1]], [[6, 2], [0, 1]], [[0, 5], [0, -1]], [[0, 3], [1, 0]], [[6, 3], [-1, 0]], [[0, 3], [1, 0]], [[3, 6], [0, -1]], [[0, 2], [1, 0]], [[0, 4], [1, 0]], [[1, 4], [1, 0]], [[3, 6], [0, -1]], [[6, 2], [0, 1]], [[6, 5], [0, -1]], [[6, 3], [-1, 0]], [[4, 6], [-1, 0]], [[5, 0], [-1, 0]], [[2, 0], [1, 0]], [[4, 0], [0, 1]], [[4, 1], [0, 1]], [[6, 3], [-1, 0]], [[2, 6], [1, 0]], [[5, 6], [-1, 0]], [[3, 6], [0, -1]], [[5, 0], [-1, 0]], [[3, 0], [0, 1]], [[6, 2], [0, 1]], [[6, 5], [0, -1]], [[6, 6], [0, -1]], [[6, 5], [0, -1]], [[6, 2], [0, 1]], [[6, 5], [0, -1]], [[6, 3], [-1, 0]], [[6, 2], [0, 1]], [[6, 5], [0, -1]], [[6, 4], [-1, 0]], [[6, 2], [0, 1]], [[6, 4], [-1, 0]], [[4, 5], [0, -1]], [[6, 3], [-1, 0]], [[0, 3], [1, 0]], [[3, 6], [0, -1]], [[0, 2], [0, 1]], [[0, 4], [1, 0]], [[1, 4], [1, 0]], [[3, 6], [0, -1]], [[0, 6], [1, 0]], [[1, 6], [1, 0]], [[2, 6], [1, 0]], [[4, 6], [0, -1]], [[5, 6], [-1, 0]], [[3, 6], [0, -1]], [[4, 6], [-1, 0]], [[1, 6], [1, 0]], [[2, 6], [1, 0]], [[5, 6], [-1, 0]], [[3, 6], [0, -1]], [[5, 4], [-1, 0]], [[3, 6], [0, -1]], [[3, 0], [0, 1]], [[1, 2], [1, 0]], [[3, 0], [0, 1]], [[6, 0], [0, 1]], [[6, 1], [0, 1]], [[6, 3], [-1, 0]], [[6, 5], [0, -1]], [[6, 2], [0, 1]], [[6, 5], [0, -1]], [[6, 3], [-1, 0]], [[6, 4], [-1, 0]], [[0, 5], [0, -1]], [[0, 3], [1, 0]], [[6, 3], [-1, 0]], [[6, 3], [-1, 0]], [[4, 6], [-1, 0]], [[1, 6], [1, 0]], [[4, 6], [-1, 0]], [[1, 6], [1, 0]], [[3, 6], [0, -1]], [[0, 3], [1, 0]], [[2, 6], [0, -1]], [[3, 0], [0, 1]], [[3, 6], [0, -1]], [[6, 4], [0, -1]], [[6, 1], [0, 1]], [[6, 3], [-1, 0]], [[2, 0], [1, 0]], [[5, 0], [-1, 0]], [[2, 0], [1, 0]], [[5, 0], [-1, 0]], [[3, 0], [0, 1]], [[2, 6], [1, 0]], [[4, 6], [0, -1]], [[3, 0], [0, 1]], [[5, 6], [-1, 0]], [[6, 2], [0, 1]], [[6, 5], [0, -1]], [[6, 3], [-1, 0]], [[5, 4], [-1, 0]], [[3, 6], [0, -1]], [[3, 6], [0, -1]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[6, 3], [-1, 0]], [[3, 0], [0, 1]], [[3, 6], [0, -1]], [[3, 6], [0, -1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[6, 3], [-1, 0]], [[4, 6], [-1, 0]], [[2, 0], [1, 0]], [[5, 0], [-1, 0]], [[3, 0], [0, 1]], [[6, 4], [0, -1]], [[6, 2], [-1, 0]], [[6, 3], [-1, 0]], [[6, 3], [-1, 0]], [[6, 3], [-1, 0]], [[6, 3], [-1, 0]], [[0, 3], [1, 0]], [[1, 6], [1, 0]], [[3, 6], [0, -1]], [[6, 1], [0, 1]], [[6, 3], [-1, 0]], [[3, 0], [0, 1]], [[3, 6], [0, -1]], [[3, 6], [0, -1]], [[3, 6], [0, -1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[2, 6], [1, 0]], [[5, 6], [-1, 0]], [[2, 6], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[2, 0], [1, 0]], [[5, 0], [-1, 0]], [[2, 0], [1, 0]], [[5, 0], [-1, 0]], [[2, 0], [1, 0]], [[5, 0], [-1, 0]], [[2, 0], [1, 0]], [[4, 0], [0, 1]], [[4, 1], [0, 1]], [[3, 6], [0, -1]], [[4, 6],

[0, -1]], [[0, 3], [1, 0]], [[2, 6], [0, -1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[1, 2], [1, 0]], [[3, 6], [0, -1]], [[6, 3], [-1, 0]], [[6, 3], [-1, 0]], [[6, 3], [-1, 0]], [[0, 3], [1, 0]], [[2, 5], [0, -1]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 5], [0, -1]], [[3, 0], [0, 1]], [[3, 6], [0, -1]], [[5, 4], [-1, 0]], [[3, 6], [0, -1]], [[5, 0], [-1, 0]], [[5, 6], [-1, 0]], [[6, 3], [-1, 0]], [[3, 0], [0, 1]], [[3, 6], [0, -1]], [[4, 6], [-1, 0]], [[1, 6], [1, 0]], [[3, 6], [0, -1]], [[0, 2], [0, 1]], [[0, 4], [1, 0]], [[1, 4], [1, 0]], [[3, 6], [0, -1]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[2, 0], [1, 0]], [[5, 0], [-1, 0]], [[3, 0], [0, 1]], [[3, 6], [0, -1]], [[6, 2], [0, 1]], [[6, 4], [-1, 0]], [[6, 3], [-1, 0]], [[6, 3], [-1, 0]], [[6, 3], [-1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[3, 6], [0, -1]], [[3, 6], [0, -1]], [[6, 3], [-1, 0]], [[3, 6], [0, -1]], [[3, 6], [0, -1]], [[3, 6], [0, -1]], [[5, 2], [-1, 0]], [[6, 3], [-1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 5], [0, -1]], [[6, 3], [-1, 0]], [[6, 3], [-1, 0]], [[6, 3], [-1, 0]], [[6, 3], [-1, 0]], [[6, 5], [0, -1]], [[4, 6], [-1, 0]], [[1, 6], [1, 0]], [[4, 6], [-1, 0]], [[2, 6], [0, -1]], [[2, 5], [0, -1]], [[0, 2], [0, 1]], [[0, 5], [0, -1]], [[0, 2], [0, 1]], [[0, 5], [0, -1]], [[0, 2], [0, 1]], [[0, 5], [0, -1]], [[0, 2], [0, 1]], [[0, 5], [0, -1]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[2, 0], [1, 0]], [[5, 0], [-1, 0]], [[2, 0], [1, 0]], [[5, 0], [-1, 0]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[6, 2], [0, 1]], [[6, 5], [0, -1]], [[6, 2], [0, 1]], [[6, 4], [-1, 0]], [[5, 4], [-1, 0]], [[3, 6], [0, -1]], [[3, 6], [0, -1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[6, 3], [-1, 0]], [[6, 3], [-1, 0]], [[6, 3], [-1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[3, 0], [0, 1]], [[3, 6], [0, -1]], [[3, 6], [0, -1]], [[3, 6], [0, -1]], [[6, 3], [-1, 0]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 6], [0, -1]], [[3, 6], [0, -1]], [[3, 6], [0, -1]], [[3, 6], [0, -1]], [[3, 6], [0, -1]], [[3, 6], [0, -1]], [[3, 6], [0, -1]], [[3, 6], [0, -1]], [[6, 3], [-1, 0]], [[6, 3], [-1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[0, 3], [1, 0]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 0], [0, 1]], [[3, 6], [0, -1]], [[3, 6], [0, -1]], [[3, 6], [0, -1]], [[3, 6], [0, -1]], [[3, 6], [0, -1]], [[3, 6], [0, -1]], [[3, 6], [0, -1]], [[6, 3], [-1, 0]], [[6, 3], [-1, 0]]]

```
[[   0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0]
 [   0    0    0 4900    0    0    0]
 [   0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0]]
```

Nummer 4:

count: 1152

eval: 0

[[[0, 1], [0, -1]], [[0, 7], [0, 1]], [[7, 0], [1, 0]], [[7, 8], [1, 0]], [[0, 1], [0, -1]], [[7, 0], [1, 0]], [[0, 7], [0, 1]], [[7, 8], [1, 0]], [[7, 8], [1, 0]], [[0, 1], [0, -1]], [[7, 0], [1, 0]], [[0, 7], [0, 1]], [[0, 1], [0, -1]], [[0, 2], [1, 0]], [[2, 1], [0, 1]], [[1, 3], [1, 0]], [[2, 3], [1, 0]], [[4, 2], [0, 1]], [[3, 1], [0, 1]], [[0, 3], [1, 0]], [[1, 3], [1, 0]], [[2, 3], [1, 0]], [[4, 1], [0, 1]], [[4, 2], [0, 1]], [[6, 3], [-1, 0]], [[1, 4], [1, 0]], [[2, 4], [1, 0]], [[1, 5], [1, 0]], [[3, 6], [0, -1]], [[1, 6], [1, 0]], [[3, 7], [0, -1]], [[3, 6], [0, -1]], [[2, 4], [1, 0]], [[6, 4], [-1, 0]], [[4, 7], [0, -1]], [[4, 6], [0, -1]], [[4, 2], [0, 1]], [[3, 8], [0, -1]], [[5, 7], [0, -1]], [[5, 6], [0, -1]], [[7, 4], [-1, 0]], [[7, 5], [-1, 0]], [[5, 0], [0, 1]], [[5, 1], [0, 1]], [[6, 3], [-1, 0]], [[6, 1], [0, 1]], [[7, 3], [-1, 0]], [[6, 3], [-1, 0]], [[6, 4], [-1, 0]], [[6, 8], [0, -1]], [[7, 6], [-1, 0]], [[5, 7], [0, -1]], [[5, 6], [0, -1]], [[5, 8], [0, -1]], [[8, 6], [-1, 0]], [[7, 6], [-1, 0]], [[5, 7], [0, -1]], [[5, 6], [0, -1]], [[8, 3], [-1, 0]], [[2, 0], [0, 1]], [[8, 5], [-1, 0]], [[7, 5], [-1, 0]], [[3, 0], [0, 1]], [[1, 2], [1, 0]], [[3, 1], [0, 1]], [[3, 2], [0, 1]], [[2, 4], [1, 0]], [[6, 4], [-1, 0]], [[4, 2], [0, 1]], [[4, 6], [0, -1]], [[6, 5], [-1, 0]], [[0, 6], [1, 0]], [[2, 8], [0, -1]], [[1, 6], [1, 0]], [[3, 7], [0, -1]], [[2, 5], [1, 0]], [[0, 5], [1, 0]], [[1, 5], [1, 0]], [[6, 0], [0, 1]], [[8, 2], [-1, 0]], [[6, 1], [0, 1]], [[7, 3], [-1, 0]], [[5, 2], [0, 1]], [[7, 4], [-1, 0]], [[7, 0], [1, 0]], [[4, 7], [0, -1]], [[3, 6], [0, -1]], [[1, 4], [1, 0]], [[0, 7], [0, 1]], [[7, 8], [1, 0]], [[4, 0], [0, 1]], [[4, 1], [0, 1]], [[2, 4], [1, 0]], [[6, 4], [-1, 0]], [[0, 1], [0, -1]], [[0, 4], [1, 0]], [[1, 4], [1, 0]], [[4, 8], [0, -1]], [[4, 7], [0, -1]], [[8, 4], [-1, 0]], [[8, 8], [0, -1]], [[8, 7], [0, -1]], [[8, 6], [0, -1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 5], [-1, 0]], [[7, 5], [-1, 0]], [[5, 6], [0, -1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 4], [-1, 0]], [[7, 4], [-1, 0]], [[8, 5], [0, -1]], [[8, 2], [0, 1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 5], [-1, 0]], [[0, 8], [1, 0]], [[1, 8], [1, 0]], [[2, 8], [1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[3, 8], [1, 0]], [[5, 8], [0, -1]], [[6, 8], [-1, 0]], [[5, 7], [0, -1]], [[4, 8], [0, -1]], [[4, 7], [0, -1]], [[4, 6], [0, -1]], [[4, 2], [0, 1]], [[6, 5], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[7, 5], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[0, 0], [1, 0]], [[1, 0], [1, 0]], [[2, 0], [1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[5, 0], [0, 1]], [[5, 1], [0, 1]], [[6, 3], [-1, 0]], [[4, 0], [0, 1]], [[4, 1], [0, 1]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[8, 0], [0, 1]], [[8, 1], [0, 1]], [[8, 2], [0, 1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 4], [-1, 0]], [[7, 4], [-1, 0]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 5], [-1, 0]], [[7, 5], [-1, 0]], [[6, 5], [-1, 0]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[4, 7], [0, -1]], [[3, 6], [0, -1]], [[2, 3], [1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[5, 0], [0, 1]], [[5, 1], [0, 1]], [[6, 3], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[0, 7], [1, 0]], [[2, 8], [0, -1]], [[1, 6], [1, 0]], [[3, 7], [0, -1]], [[0, 1], [1, 0]], [[2, 0], [0, 1]], [[1, 2], [1, 0]], [[3, 1], [0, 1]], [[2, 3], [1, 0]], [[4, 1], [0, 1]], [[7, 8], [0, -1]], [[8, 6], [-1, 0]], [[7, 6], [-1, 0]], [[5, 7], [0, -1]], [[5, 6], [0, -1]], [[5, 8], [0, -1]], [[7, 4], [-1, 0]], [[8, 1], [-1, 0]], [[6, 0], [0, 1]], [[6, 1], [0, 1]], [[7, 3], [-1, 0]], [[6, 3], [-1, 0]], [[4, 2], [0, 1]], [[4, 6], [0, -1]], [[6, 4], [-1, 0]], [[2, 4], [1, 0]], [[3, 6], [0, -1]], [[8, 3], [-1, 0]], [[4, 1], [0, 1]], [[5, 7], [0, -1]], [[7, 3], [-1, 0]], [[7, 0], [1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[0, 2], [0, 1]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 5], [1, 0]], [[1, 5], [1, 0]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 4], [1, 0]], [[1, 4], [1, 0]], [[2, 5], [1, 0]], [[3, 2], [0, 1]], [[7, 0], [-1, 0]], [[6, 0], [-1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[4, 0], [0, 1]], [[4, 1], [0, 1]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[5, 0], [0, 1]], [[5,

1], [0, 1]], [[5, 2], [0, 1]], [[6, 5], [-1, 0]], [[7, 4], [-1, 0]], [[4, 1], [0, 1]], [[3, 0], [0, 1]], [[4, 7], [0, -1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[3, 1], [0, 1]], [[3, 8], [0, -1]], [[8, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 8], [-1, 0]], [[6, 8], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[3, 8], [1, 0]], [[5, 8], [0, -1]], [[6, 8], [-1, 0]], [[4, 8], [0, -1]], [[4, 7], [0, -1]], [[5, 7], [0, -1]], [[3, 7], [0, -1]], [[3, 6], [0, -1]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[1, 3], [1, 0]], [[3, 2], [0, 1]], [[1, 4], [1, 0]], [[0, 7], [0, -1]], [[0, 6], [0, -1]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 5], [1, 0]], [[1, 5], [1, 0]], [[8, 5], [-1, 0]], [[7, 5], [-1, 0]], [[6, 5], [-1, 0]], [[3, 6], [0, -1]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[4, 1], [0, 1]], [[1, 4], [1, 0]], [[7, 4], [-1, 0]], [[0, 1], [0, 1]], [[0, 2], [0, 1]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 4], [1, 0]], [[1, 4], [1, 0]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 5], [1, 0]], [[1, 5], [1, 0]], [[3, 6], [0, -1]], [[2, 3], [1, 0]], [[1, 4], [1, 0]], [[4, 1], [0, 1]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 8], [0, -1]], [[0, 7], [0, -1]], [[0, 6], [0, -1]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 5], [1, 0]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[2, 7], [0, -1]], [[1, 5], [1, 0]], [[1, 4], [1, 0]], [[8, 0], [-1, 0]], [[7, 0], [-1, 0]], [[6, 0], [-1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[5, 0], [0, 1]], [[5, 1], [0, 1]], [[5, 0], [-1, 0]], [[2, 0], [1, 0]], [[4, 0], [0, 1]], [[4, 1], [0, 1]], [[6, 3], [-1, 0]], [[4, 1], [0, 1]], [[5, 0], [-1, 0]], [[2, 0], [1, 0]], [[8, 8], [-1, 0]], [[7, 8], [-1, 0]], [[6, 8], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[5, 8], [0, -1]], [[4, 7], [0, -1]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[4, 8], [0, -1]], [[4, 7], [0, -1]], [[5, 7], [0, -1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[7, 4], [-1, 0]], [[4, 7], [0, -1]], [[0, 1], [0, -1]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 4], [1, 0]], [[1, 4], [1, 0]], [[4, 1], [0, 1]], [[2, 5], [1, 0]], [[4, 7], [0, -1]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[7, 4], [-1, 0]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[5, 6], [0, -1]], [[7, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 7], [0, -1]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[4, 1], [0, 1]], [[6, 8], [0, -1]], [[6, 7], [0, -1]], [[7, 5], [-1, 0]], [[7, 4], [-1, 0]], [[0, 2], [1, 0]], [[2, 1], [0, 1]], [[1, 3], [1, 0]], [[2, 3], [1, 0]], [[5, 2], [0, 1]], [[5, 6], [0, -1]], [[7, 4], [-1, 0]], [[4, 1], [0, 1]], [[1, 4], [1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[7, 4], [-1, 0]], [[4, 7], [0, -1]], [[4, 1], [0, 1]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[0, 5], [1, 0]], [[1, 4], [1, 0]], [[3, 7], [0, -1]], [[1, 5], [1, 0]], [[8, 5], [-1, 0]], [[7, 4], [-1, 0]], [[4, 7], [0, -1]], [[0, 7], [0, 1]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[7, 0], [1, 0]], [[4, 1], [0, 1]], [[1, 4], [1, 0]], [[4, 7], [0, -1]], [[7, 3], [-1, 0]], [[5, 2], [0, 1]], [[7, 4], [-1, 0]], [[7, 5], [-1, 0]], [[0, 6], [1, 0]], [[1, 6], [1, 0]], [[3, 7], [0, -1]], [[2, 5], [1, 0]], [[4, 7], [0, -1]], [[6, 5], [-1, 0]], [[4, 7], [0, -1]], [[0, 3], [0, 1]], [[4, 1], [0, 1]], [[5, 8], [0, -1]], [[5, 7], [0, -1]], [[4, 7], [0, -1]], [[7, 4], [-1, 0]], [[1, 4], [1, 0]], [[3, 0], [0, 1]], [[3, 1], [0, 1]], [[4, 1], [0, 1]], [[0, 6], [0, -1]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 4], [-1, 0]], [[7, 4], [-1, 0]], [[8, 2], [-1, 0]], [[7, 2], [-1, 0]], [[5, 1], [0, 1]], [[4, 7], [0, -1]], [[5, 0], [0, 1]], [[7, 4], [-1, 0]], [[4, 1], [0, 1]], [[1, 4], [1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[0, 3], [1, 0]], [[1, 4], [1, 0]], [[4, 7], [0, -1]], [[5, 1], [0, 1]], [[6, 3], [-1, 0]], [[3, 2], [0, 1]], [[1, 4], [1, 0]], [[3, 6], [0, -1]], [[4, 1], [0, 1]], [[1, 4], [1, 0]], [[7, 4], [-1, 0]], [[5, 0], [-1, 0]], [[2, 0], [1, 0]], [[4, 0], [0, 1]], [[4, 1], [0, 1]], [[4, 7], [0, -1]], [[4, 1], [0, 1]], [[7, 4], [-1, 0]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[7, 0], [1, 0]], [[7, 8], [1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[4, 7], [0, -1]], [[1, 4], [1, 0]], [[7, 4], [-1, 0]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[1, 3], [1, 0]], [[4, 1], [0, 1]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[1, 4], [1, 0]], [[3, 2], [0, 1]], [[1, 4], [1, 0]], [[4, 8], [0, -1]], [[4, 7], [0, -1]], [[0, 3], [0, 1]], [[0, 5], [1, 0]], [[1, 5], [1, 0]], [[1, 4], [1, 0]], [[0, 6], [0, -1]], [[4, 7], [0, -1]], [[7, 4], [-1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[6, 0], [0, 1]], [[7, 3], [-1, 0]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[4, 1], [0, 1]], [[7, 4], [-1, 0]], [[4, 7], [0, -1]], [[8, 3], [0, 1]], [[8, 5], [-1, 0]], [[7, 5], [-1, 0]], [[6, 5], [-1, 0]], [[3, 6], [0, -1]], [[4, 7], [0, -1]], [[1, 4], [1, 0]], [[4, 7], [0, -1]], [[7, 4], [-1, 0]], [[4, 1], [0, 1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[1, 0], [1, 0]], [[2, 0], [1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[4, 0], [0, 1]], [[4, 1], [0, 1]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[6, 3], [-1, 0]], [[4, 1], [0, 1]], [[5, 0], [0, 1]], [[7, 4], [-1, 0]], [[0, 3], [0, 1]], [[0, 5], [1, 0]], [[1, 5], [1, 0]], [[2, 5], [1, 0]], [[5, 6], [0, -1]], [[7, 4], [-1, 0]], [[4, 7], [0, -1]], [[5, 1], [0, 1]], [[1, 4], [1, 0]], [[2, 8], [1, 0]], [[6, 8], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[4, 8], [0, -1]], [[4, 7], [0, -1]], [[7, 4], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[4, 1], [0, 1]], [[6, 3], [-1, 0]], [[3, 2], [0, 1]], [[4, 1], [0, 1]], [[1, 4], [1, 0]], [[1, 4], [1, 0]], [[3, 7], [0, -1]], [[3, 6], [0, -1]], [[1, 4], [1, 0]], [[7, 4], [-1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[4, 1], [0, 1]], [[4, 7], [0, -1]], [[6, 5], [-1, 0]], [[4, 7], [0, -1]], [[4, 7], [0, -1]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[3, 8], [1, 0]], [[5, 8], [0, -1]], [[7, 6], [-1, 0]], [[5, 7], [0, -1]], [[4, 7], [0, -1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[1, 0], [1, 0]], [[2, 0], [1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[4, 0], [0, 1]], [[4, 1], [0, 1]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[6, 3], [-1, 0]], [[4, 1], [0, 1]], [[5, 0], [0, 1]], [[7, 4], [-1, 0]], [[0, 3], [0, 1]], [[0, 5], [1, 0]], [[1, 5], [1, 0]], [[2, 5], [1, 0]], [[5, 6], [0, -1]], [[7, 4], [-1, 0]], [[4, 7], [0, -1]], [[5, 1], [0, 1]], [[1, 4], [1, 0]], [[2, 8], [1, 0]], [[6, 8], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[4, 8], [0, -1]], [[4, 7], [0, -1]], [[7, 4], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[4, 1], [0, 1]], [[6, 3], [-1, 0]], [[3, 2], [0, 1]], [[4, 1], [0, 1]], [[1, 4], [1, 0]], [[1, 4], [1, 0]], [[3, 7], [0, -1]], [[3, 6], [0, -1]], [[1, 4], [1, 0]], [[7, 4], [-1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[4, 1], [0, 1]], [[4, 7], [0, -1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[3, 0], [0, 1]], [[4, 1], [0, 1]], [[5, 8], [0, -1]], [[5, 7], [0, -1]], [[0, 6], [0, -1]], [[4, 7], [0, -1]], [[8, 7], [0, -1]], [[8, 6], [0, -1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 5], [-1, 0]], [[7, 5], [-1, 0]], [[5, 6], [0, -1]], [[7, 4], [-1, 0]], [[8, 4], [-1, 0]], [[7, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 7], [0, -1]], [[4, 7], [0, -1]], [[7, 3], [-1, 0]], [[4, 1], [0, 1]], [[7, 4], [-1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[3, 0], [1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[6, 0], [-1, 0]], [[4, 1], [0, 1]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[7, 4], [-1, 0]], [[3, 1], [0, 1]], [[2, 3], [1, 0]], [[6, 3], [-1, 0]], [[4, 1], [0, 1]], [[4, 1], [0, 1]], [[1, 4], [1, 0]], [[7, 4], [-1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[4, 1], [0, 1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[2, 0], [0, 1]], [[2, 1], [0, 1]], [[1, 3], [1, 0]], [[2, 3], [1, 0]], [[4, 1], [0, 1]], [[4, 7], [0, -1]], [[7, 4], [-1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[4, 0], [0, 1]], [[4, 1], [0, 1]], [[4, 1], [0, 1]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[1, 4], [1, 0]], [[1, 4], [1, 0]], [[7, 4], [-1, 0]], [[5, 2], [0, 1]], [[7, 4], [-1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[3, 8], [1, 0]], [[5, 8], [0, -1]], [[5, 7], [0, -1]], [[5, 6], [0, -1]], [[7, 4], [-1, 0]], [[6, 8], [-1, 0]], [[4, 1], [0, 1]], [[4, 7], [0, -1]], [[5, 0], [0, 1]], [[5, 1], [0, 1]], [[7, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 7], [0, -1]], [[0, 2], [0, 1]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 4], [1, 0]], [[1, 4], [1, 0]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[3, 0], [1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[4, 8], [0, -1]], [[4, 7], [0, -1]], [[4, 1], [0, 1]], [[6, 0], [-1, 0]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 3], [-1, 0]], [[1, 4], [1, 0]], [[4, 7], [0, -1]], [[4, 1], [0, 1]], [[4, 7], [0, -1]], [[1, 4], [1, 0]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 4], [1, 0]], [[1, 4], [1, 0]], [[1, 4], [1, 0]], [[1, 4], [1, 0]], [[1, 8], [1, 0]], [[3, 8], [0, -1]], [[3, 7], [0, -1]], [[3, 6], [0, -1]], [[2, 3], [1, 0]], [[5, 2], [0, 1]], [[7, 4], [-1, 0]], [[4, 7], [0, -1]], [[4, 1], [0, 1]], [[7, 4], [-1, 0]], [[4, 1], [0, 1]], [[1, 4], [1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[8, 2], [0, 1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 5], [-1, 0]], [[7, 4], [-1, 0]], [[4, 1], [0, 1]], [[7, 4], [-1, 0]], [[7, 5], [-1, 0]], [[8, 6], [-1, 0]], [[1, 4], [1, 0]], [[4, 1], [0, 1]], [[1, 4], [1, 0]], [[4, 7], [0, -1]], [[7, 6], [-1, 0]], [[5, 7], [0, -1]], [[8, 2], [-1, 0]], [[6,

1], [0, 1]], [[8, 3], [-1, 0]], [[7, 4], [-1, 0]], [[6, 1], [0, 1]], [[7, 3], [-1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[3, 0], [1, 0]], [[5, 0], [0, 1]], [[4, 1], [0, 1]], [[8, 3], [0, 1]], [[0, 5], [1, 0]], [[1, 4], [1, 0]], [[1, 5], [1, 0]], [[4, 1], [0, 1]], [[4, 7], [0, -1]], [[2, 8], [1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[4, 8], [0, -1]], [[4, 7], [0, -1]], [[4, 7], [0, -1]], [[4, 1], [0, 1]], [[4, 7], [0, -1]], [[7, 4], [-1, 0]], [[8, 6], [0, -1]], [[8, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[6, 0], [-1, 0]], [[4, 0], [0, 1]], [[4, 1], [0, 1]], [[6, 5], [-1, 0]], [[2, 5], [1, 0]], [[4, 1], [0, 1]], [[0, 0], [1, 0]], [[1, 0], [1, 0]], [[2, 0], [1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[1, 4], [1, 0]], [[4, 0], [0, 1]], [[4, 1], [0, 1]], [[4, 1], [0, 1]], [[4, 7], [0, -1]], [[5, 2], [0, 1]], [[7, 4], [-1, 0]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[1, 4], [1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[4, 7], [0, -1]], [[5, 0], [-1, 0]], [[4, 1], [0, 1]], [[4, 1], [0, 1]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[0, 8], [1, 0]], [[1, 8], [1, 0]], [[2, 8], [1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[4, 8], [0, -1]], [[4, 7], [0, -1]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[3, 8], [1, 0]], [[6, 8], [-1, 0]], [[1, 4], [1, 0]], [[2, 0], [1, 0]], [[4, 1], [0, 1]], [[4, 7], [0, -1]], [[7, 4], [-1, 0]], [[8, 5], [-1, 0]], [[7, 5], [-1, 0]], [[4, 7], [0, -1]], [[4, 7], [0, -1]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[1, 3], [1, 0]], [[3, 1], [0, 1]], [[1, 3], [1, 0]], [[3, 2], [0, 1]], [[3, 6], [0, -1]], [[1, 4], [1, 0]], [[2, 8], [0, -1]], [[1, 6], [1, 0]], [[2, 6], [1, 0]], [[4, 7], [0, -1]], [[4, 7], [0, -1]], [[4, 7], [0, -1]], [[4, 7], [0, -1]], [[5, 0], [0, 1]], [[5, 1], [0, 1]], [[7, 4], [-1, 0]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 4], [1, 0]], [[1, 4], [1, 0]], [[3, 0], [0, 1]], [[2, 2], [1, 0]], [[4, 1], [0, 1]], [[6, 3], [-1, 0]], [[6, 8], [0, -1]], [[6, 7], [0, -1]], [[7, 5], [-1, 0]], [[3, 8], [1, 0]], [[5, 8], [0, -1]], [[4, 7], [0, -1]], [[5, 7], [0, -1]], [[5, 6], [0, -1]], [[7, 4], [-1, 0]], [[4, 1], [0, 1]], [[4, 7], [0, -1]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[4, 1], [0, 1]], [[7, 4], [-1, 0]], [[8, 4], [-1, 0]], [[7, 4], [-1, 0]], [[1, 4], [1, 0]], [[1, 4], [1, 0]], [[1, 4], [1, 0]], [[4, 0], [0, 1]], [[4, 1], [0, 1]], [[4, 7], [0, -1]], [[4, 7], [0, -1]], [[4, 8], [0, -1]], [[5, 7], [0, -1]], [[6, 5], [-1, 0]], [[4, 7], [0, -1]], [[4, 7], [0, -1]], [[4, 7], [0, -1]], [[4, 7], [0, -1]], [[7, 4], [-1, 0]], [[8, 8], [0, -1]], [[8, 6], [-1, 0]], [[6, 7], [0, -1]], [[6, 6], [0, -1]], [[8, 4], [-1, 0]], [[7, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 1], [0, 1]], [[4, 1], [0, 1]], [[4, 1], [0, 1]], [[4, 1], [0, 1]], [[4, 1], [0, 1]], [[4, 7], [0, -1]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[1, 4], [1, 0]], [[1, 4], [1, 0]], [[4, 1], [0, 1]], [[6, 8], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 3], [0, 1]], [[0, 6], [0, -1]], [[0, 4], [1, 0]], [[1, 4], [1, 0]], [[1, 4], [1, 0]], [[1, 4], [1, 0]], [[1, 4], [1, 0]], [[1, 4], [1, 0]], [[1, 4], [1, 0]], [[1, 4], [1, 0]], [[3, 2], [0, 1]], [[3, 0], [1, 0]], [[6, 0], [-1, 0]], [[4, 0], [0, 1]], [[4, 1], [0, 1]], [[3, 8], [1, 0]], [[5, 8], [0, -1]], [[4, 7], [0, -1]], [[2, 5], [1, 0]], [[4, 1], [0, 1]], [[6, 3], [-1, 0]], [[2, 3], [1, 0]], [[4, 1], [0, 1]], [[4, 7], [0, -1]], [[4, 7], [0, -1]], [[6, 5], [-1, 0]], [[4, 7], [0, -1]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 3], [0, 1]], [[8, 5], [-1, 0]], [[6, 6], [0, -1]], [[7, 4], [-1, 0]], [[8, 7], [0, -1]], [[8, 6], [0, -1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 3], [0, 1]], [[8, 6], [0, -1]], [[8, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 1], [0, 1]], [[4, 1], [0, 1]], [[4, 1], [0, 1]], [[4, 1], [0, 1]], [[4, 7], [0, -1]], [[4, 7], [0, -1]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[1, 4], [1, 0]], [[1, 4], [1, 0]], [[1, 4], [1, 0]], [[1, 4], [1, 0]], [[1, 4], [1, 0]], [[4, 1], [0, 1]], [[4, 7], [0, -1]], [[4, 7], [0, -1]], [[4, 7], [0, -1]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[1, 4], [1, 0]], [[4, 1], [0, 1]], [[4, 1], [0, 1]], [[4, 1], [0, 1]], [[4, 1], [0, 1]], [[4, 1], [0, 1]], [[4, 7], [0, -1]], [[4, 7], [0, -1]], [[4, 7], [0, -1]], [[4, 7], [0, -1]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[1, 4], [1, 0]], [[1, 4], [1, 0]], [[1, 4], [1, 0]], [[1, 4], [1, 0]], [[4, 1], [0, 1]], [[4, 1], [0, 1]], [[4, 1], [0, 1]], [[4, 1], [0, 1]], [[4, 1], [0, 1]], [[4, 1], [0, 1]], [[4, 1], [0, 1]], [[4, 1], [0, 1]], [[4, 1], [0, 1]], [[4, 1], [0, 1]], [[4, 1], [0, 1]], [[4, 7], [0, -1]], [[4, 7], [0, -1]], [[4, 7], [0, -1]], [[4, 7], [0, -1]], [[4, 7], [0, -1]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]], [[7, 4], [-1, 0]]]

6. Quellcode

Da aus Platzgründen die Programme der verschiedenen Ansätze nicht komplett auf die Arbeit passen, wird nur der Greedy-Algorithmus G1 hier abgedruckt. Die Quellcodes aller anderen Ansätze sind jedoch auch in der Abgabe enthalten und auskommentiert, sodass diese dort auch nochmal eingesehen werden können. Hinweis zum Boxengenerator: Das Programm kann nur mit einer .txt Datei namens „paeckchengen.txt“ im selben Ordner ausgeführt werden. Eine leere Eingabedatei ist in der Einsendung enthalten.

```
###dephsearch alg (rein greedy)
import numpy as np
import random
import sys
import time


g = int(input("Bitte eine Zahl für i eingeben: "))
f = int(input("Bitte eine Zahl für j eingeben: "))
s = int(input("bitte eine Zahl für s eingeben: "))
t = int(input("bitte eine Zahl für t eingeben: "))
```

```
blaetter = int(input("bitte eine Zahl für die Anzahl der Blätter eingeben: "))

start_time = time.time()
groessen = (f,g)

zielquadrat = [s,t]

schulhof = np.full((groessen), blaetter)
pustpfad = []

#die Hausmeister-Pustefunktion.
#Argumente: Position des Hausmeisters und Richtung des Pustens
#Simuliert das Pusten der Blätter auf dem Schulhof.
def hausmeister_pustet(position, richtung):
    def istImHof(feld):
        if 0 <= feld[0] < groessen[0] and 0<= feld[1] < groessen[1]:
            return 1
        else:
            return 0

    #Definieren der in der Doku angegebenen einzelnen Felder:
    pustfeld = np.add(position, richtung)
    ziel = np.add(pustfeld, richtung)
    ziel_links = np.add(ziel, [richtung[1], richtung[0]])
    ziel_rechts = np.add(ziel, [richtung[1]*-1, richtung[0]*-1])
    neuesziel = np.add(ziel, richtung)

    if istImHof(pustfeld)==0:#wenn das gegebene Feld sich nicht im Hof befindet...
        print(f"FEHLER!!!: das pustfeld {pustfeld} ist nicht im Hof")#ist ein Fehler
aufgetreten
        exit()

    if not istImHof(ziel):#wenn das Ziel sich nicht im Schulhof befindet, wird die
Hälfte einfahc zur Seite geblasen.
        haelfte_der_blaetter = schulhof[tuple(pustfeld)]
        if istImHof(np.add(pustfeld, [richtung[1], richtung[0]])):#ist das Eine
Seitfeld im hof?
            for i in range(schulhof[tuple(pustfeld)]):
                if not random.randint(0,1):
                    schulhof[tuple(np.add(pustfeld, [richtung[1], richtung[0]]))]+=1
                    schulhof[tuple(pustfeld)]-=1
            if istImHof(np.add(pustfeld, [richtung[1]*-1, richtung[0]*-1])):#Addiere
die restlichen Blätter zum anderen Seitenfeld, sofern dieser auch im Hof ist
                schulhof[tuple(np.add(pustfeld, [richtung[1]*-1, richtung[0]*-1]))]+=
schulhof[tuple(pustfeld)]
                schulhof[tuple(pustfeld)] = 0
        elif istImHof(np.add(pustfeld, [richtung[1]*-1, richtung[0]*-1])):#ist das
andere Seitfeld im hof?
```

```
            for i in range(schulhof[tuple(pustfeld)]):
                if random.randint(0,1):
                    schulhof[tuple(np.add(pustfeld, [richtung[1]*-1, richtung[0]*-
1]))]+=1
                    schulhof[tuple(pustfeld)]-=1
        return schulhof

    if istImHof (neuesziel):#ist das Feld neues ziel im Hof (also das hinterste Feld)?
Wenn nicht, puste wie normal die restliche Blätter. Wenn doch:
        for i in range(schulhof[tuple(ziel)]):#für jedes Blatt aus den Feld "ziel"
            if not random.randint(0,9):#wahrscheinlichkeit 10%
                schulhof[tuple(neuesziel)] += 1
                schulhof[tuple(ziel)]-=1

    #Wir simulieren die Blätter auf den Seitenrändern.
    for i in range(schulhof[tuple(pustfeld)]):
        if not random.randint(0,9):#erstelle eine zufallszahl
            if istImHof(ziel_links):#wenn das linke Ziel im Hof ist, dann füge ein
Blatt hinzu
                schulhof[tuple(ziel_links)] += 1
                schulhof[tuple(pustfeld)] -= 1
            else:
                pass
        if not random.randint(0,9):#erstelle eine andere zufallszahl
            if istImHof(ziel_rechts):#wenn das rechte Ziel im Hof ist, dann füge ein
Blatt hinzu
                schulhof[tuple(ziel_rechts)] += 1
                schulhof[tuple(pustfeld)] -= 1
            else:
                pass

    #wir entfernen die Blätter aus dem Pustfeld und fügen sie dem Ziel hinzu
    schulhof[tuple(ziel)] += schulhof[tuple(pustfeld)]
    schulhof[tuple(pustfeld)] = 0

    return#fertig

#Funktion: enumerate_moves
#keine argumente
#Ausgabe: eine Liste von möglichen Zügen für den Schulhof
#Die Funktion prüft, ob ein möglicher Zug legitim ist und fügt diese der Liste von
möglichen Zügen hinzu.
def enumerate_moves():#enumeriert alle legitime Züge für den Schulhof
    moves = []
    #zugstruktur: [[position],[richtung]]
    for i in range(groessen[0]):
        for j in range(groessen[1]):#prüfe für alle Felder auf den der Hasumeister
stehen könnte, ob die Rcihtung...
```

```
            if i+1<groessen[0]:
                moves.append([[i,j],[1,0]])#oben
            if j+1<groessen[1]:
                moves.append([[i,j],[0,1]])#rechts
            if i-1>=0:
                moves.append([[i,j],[-1,0]])#links
            if j-1>=0:
                moves.append([[i,j],[0,-1]])#unten
                #...okay ist. wenn ja, liste diese auf und...
    return moves#gib die Liste zurück

#####################
#Greedy step analyse#
#####################

#Funktion: pusten_predict
#Argumente: position, richtung, hof
#Gibt zurück: den neuen Zustand des Schulhofs nach dem Pusten
#Die Funktion simuliert das Pusten der Blätter auf dem Schulhof und gibt den neuen
Zustand zurück.
#Dabei arbeitet diese mit den berechneten Erwartungswerten und rundet diese auf 2
Nachkommastellen.
#Abgesehen davon ist sie der Hausmeisterfunktion vom Aufbau her ähnlich. Wir
kommentieren diese Funktion also nur spärlich aus.
def pusten_predict(position, richtung, hof):# richtung: als vektor [a,b] angegeben
a=0 oder b=0


    def istImHof(feld):
        if 0 <= feld[0] < groessen[0] and 0<= feld[1] < groessen[1]:
            return 1
        else:
            return 0

    pustfeld = np.add(position, richtung)
    ziel = np.add(pustfeld, richtung)#ziel bestimmt.
    ziel_links = np.add(ziel, [richtung[1], richtung[0]])
    ziel_rechts = np.add(ziel, [richtung[1]*-1, richtung[0]*-1])
    neuesziel = np.add(ziel, richtung)


    if istImHof(pustfeld)==0:

        print(f"FEHLER!!!: das pustfeld {pustfeld} ist nicht im Hof")
        exit()#wenn sich das gegebene Feld nicht im Hof befindet, dann ist ein Fehler
aufgetreten.

    #ab hier ist der Aufbau der Hausmeisterfunktion sehr ähnlich.
```

```
    if not istImHof(ziel):#wenn das Ziel nicht im hof ist, werden 50% werden zur
Seite geblasen
        haelfte_der_blaetter = hof[tuple(pustfeld)]
        if istImHof(np.add(pustfeld, [richtung[1], richtung[0]])):#ein seitfeld vom
IstImHof??
            hof[tuple(np.add(pustfeld, [richtung[1], richtung[0]]))]+=
haelfte_der_blaetter
            hof[tuple(pustfeld)]-= haelfte_der_blaetter
        if istImHof(np.add(pustfeld, [richtung[1]*-1, richtung[0]*-1])):
            hof[tuple(np.add(pustfeld, [richtung[1]*-1, richtung[0]*-1]))]+=
haelfte_der_blaetter
            hof[tuple(pustfeld)]-= haelfte_der_blaetter
        return hof

    if istImHof (neuesziel):#ansonsten, puste wie normal.
        hof[tuple(neuesziel)] += np.round(hof[tuple(ziel)]*0.1, decimals = 2)
        hof[tuple(ziel)] = np.round(hof[tuple(ziel)]*0.9, decimals = 2)

    if istImHof(ziel_links):#pusten der Ränder legitimieren!
        hof[tuple(ziel_links)] += np.round(hof[tuple(pustfeld)]*0.1, decimals = 2)
    else:
        hof[tuple(ziel)] += np.round(hof[tuple(pustfeld)]*0.1, decimals = 2)

    if istImHof(ziel_rechts):
        hof[tuple(ziel_rechts)] += np.round(hof[tuple(pustfeld)]*0.1, decimals = 2)
    else:
        hof[tuple(ziel)] += np.round(hof[tuple(pustfeld)]*0.1, decimals = 2)

    hof[tuple(ziel)] += np.round(hof[tuple(pustfeld)]*0.8, decimals = 2)
    hof[tuple(pustfeld)] = 0

    return hof

#Funktion eval: evaluiert den Schulhof und bewertet diesen nach Erwartungswerten
#Argumente: hof
#Gibt zurück: die Bewertung des Schulhofs
#Eval orientiert sich an der in der Doku erwähnten Heuristik und gibt entsprechend
nach der
#Bewertung einen Score zu einem Pustfeld, das als Argument eingegeben wird, zurück.
def eval(hof = None):
    score = 0
    for i in range(groessen[0]):
        for j in range(groessen[1]):#für jedes Feld im eigegebenen Hof:
            dis = np.subtract([i,j], zielquadrat)#Distanz berechnen
            dis = np.absolute(dis)
            if not(i==0 or i==groessen[0]-1) and not(j==0 or j==groessen[1]-1):#wenn
nicht am Rand liegend, distanz zur Bewertung addieren.
```

```
                score += dis[0]*hof[i][j]
                score += dis[1]*hof[i][j]
                continue
            if i==0 or i==groessen[0]-1:#sonst: extra viel dazuaddieen als "Strafe"
                score += (g)*hof[i][j]
            if j==0 or j==groessen[1]-1:
                score += (g)*hof[i][j]
    return score#score zurückgeben

#Funktion greedy_step
#Argumente: Schulhof
#Gibt zurück: den besten Zug, den der Hausmeister machen kann
#Die Greedy-Step funktion ist die Hauptfunktion des Greedy-Algorithmus.
#Sie berechnet die besten Züge, die der Hausmeister machen kann
def greedy_step(schulhof):

    evals = []
    for i in moves:#suche für jeden Zug nach der Evaluation und füge sie einer Liste
an
        evals.append(eval(pusten_predict(i[0], i[1],
np.copy(schulhof).astype(float))))

    best_eval = min(evals)#ermittle die beste Bewertung
    best_move = moves[evals.index(best_eval)]#ermittle daraus den besten zug und...

    if eval(schulhof) <= best_eval:#gib diesen zurück wenn dieser Zug den Zustand des
Schulhofes verbessert
        return 0#beende das Programm
    else:
        return best_move#sonst gib den besten Zug zurück

###############
#Hauptprogramm#
###############

moves = enumerate_moves()#enumeriere alle möglichen Züger zunächst
count = 0
while True:#wiederhole bis returned wird
    info = greedy_step(schulhof)
    #print(info)
    if info:
        hausmeister_pustet(info[0], info[1])
        print(schulhof)
        count += 1
    else:
        print(schulhof)
        print(f"count: {count}")
        print(f"eval: {eval(schulhof)}")
```

```
        print(pustpfad)
        end_time = time.time()
        print(f"Time: {end_time-start_time}")
        sys.exit()
```