Aufgabe 3: Wortsuche

Team-ID: 00960

Team: Egge Q2

Bearbeiter/-innen dieser Aufgabe:
Benedikt Biedermann, Kevin Bah, Leonie-Sophie Ilyuk, Nils Kettler, Ole Kettler, Dominik Pfeiffer, Shalina Rohdenburg, Jule Schlifke, Rebekka Schmidt

16. November 2021

Inhaltsverzeichnis

Lösungsidee

Um das Rätsel zu erstellen, braucht man als erstes das Feld, worin die gesuchten Wörter enthalten sind. Darufhin muss das Feld mit den gewünschten Wörtern befüllt werden und danach werden die restlichen freien Felder mit zufälligen Buchstaben gefühlt.

Wie im Beispiel schon angegeben ist ein Kriterium für ein schwieriges Rätsel, welche Buchstaben als Füllbuchstaben verwendet werden. Als Füllbuchstaben können entweder Buchstaben verwendet werden, die nicht in den gesuchten Wörtern vorkommen (leicht), alle Buchstaben des Alphabets (normal) oder ausschließlich Buchstaben, aus denen die gesuchten Wörter bestehen (schwer). Es wäre denkbar, auch andere Charaktere als Füllmaterial zu verwenden, dies würde das Suchspiel jedoch zu stark vereinfachen.

Ein weiteres Kriterium ist die Textrichtung. Hier bietet sich an zwischen normaler Laufrichtung (links nach rechts, normal) und umgedrehter Laufrichtung (rechts nach links, schwer) zu unterscheiden.

Weitere Kriterien wären diagonal geschriebene Wörter (normal/schwer) oder absichtlich unvollständige Varianten der gesuchten Wörter (schwer). Interessant wären auch Rätselvarianten, die aus Hexagonen bestehen oder bei denen die Wörter über die Ränder hinausgehen – dies wäre

jedoch sicher nicht mehr von der Aufgabenstellung gedeckt. Leider hatten wir für die Umsetzung dieser Ideen keine Zeit mehr, so dass es letztlich drei Schwierigkeitsgrade geworden sind:

1. **Leicht**: Nur gewohnte Laufrichtung, ausschließlich andere Buchstaben
2. **Mittel**: Nur gewohnte Laufrichtung, alle Buchstaben des Alphabets
3. **Schwer**: Auch umgedrehte Laufrichtung, nur Buchstaben der Suchbegriffe

Umsetzung

Erstellung der Feldes

Das Feld (*matrix*) wird erstellt, indem die Größe der Zeilen und Spalten aus der gegebenen Datei ausgelesen wird. Mit *make_matrix()* wird dann das Feld erstellt:

```
def make_matrix(dimensions):
    matrix = [None] * dimensions[0]
    for idx, _ in enumerate(matrix):
        matrix[idx] = [None] * dimensions[1]
```

Hier kann man die Eigenschaft von Python nutzen, Listen mit bestimmter Länge mit Multiplikation zu erzeugen. In der *for*-Schleife muss der Index *idx* verwendet werden, da sonst nur eine Kopie befüllt wird. Wir nutzen hier die Funktion *enumerate()* zum Mitzählen, und den Unterstrich für Variablen die wir nicht brauchen.

Schreiben der Wörter

Daraufhin werden die gewünschten Wörter einzeln in das Spielfeld eingefügt, bis alle Wörter im Feld vorhanden sind. Wir schreiben in allen Schwierigkeitsgraden Wörter sowohl von links nach rechts als auch von oben nach unten. Wer Kreuzworträtsel kennt, für den sind beide Richtungen normal.

Für beide Fälle muss erst ein geeigneter Platz gefunden werden. Hier das Beispiel für die Suche nach rechts (die Funktionen ähneln sich aber sehr):

```
def find_word_space_right(word, matrix):
    """ Finde eine Zeile mit genug freiem Platz für das gegebene Wort
    """
    max_offset = len(matrix[0]) - len(word)
    random_column = random.randint(0, max_offset)
    random_rows = list(range(len(matrix)))
    random.shuffle(random_rows)
    for idx in random_rows:
        count = 0
        for column in range(random_column, len(matrix[0])):
            if matrix[idx][column] is not None:
                count = 0
            else:
                count += 1
                if count == len(word):
                    start_column = column - (len(word) - 1)
                    return idx, start_colum
```

Die ersten Zeilen legen einige zufällige Startpunkte fest, die jedoch den maximalen Index nicht überschreiten dürfen (deswegen *max_offset*). Wir gehen dann jede Spalte (*column)* in der aktuellen Zeile (*row*) durch und prüfen, ob bereits ein Buchstabe in dem Feld steht. Wenn nicht, zählen wir zu der Variable *count* 1 dazu. *Count* wird für jede neue Zufallsreihe wieder auf 0 gesetzt. Wenn genügend Spalten für das Wort gefunden wurden, geben wir die Start-Koordinaten zurück und schreiben das Wort mit *write_word_right()* bzw. *write_word_down()*.

Bei beiden Funktionen kann man mit *word.reverse()* die Laufrichtung für den schweren Schwierigkeitsgrad einfach umdrehen. Damit nicht alle Worte umgedreht werden, nehmen wir mit *random.random() > 0.5* nur die Hälfte:

```
if difficulty == 3 and random.random() > 0.5:
    word = list(word)
    word.reverse()
```

Auffüllen der leeren Felder

Danach werden die leeren Felder befüllt, je nach Schwierigkeit nur mit anderen Buchstaben (1), mit zufälligen Buchstaben (2) oder nur mit welchen aus den gesuchten Worten, wenn die Schwierigkeit 3 beträgt.

```
if matrix[idx][col] is None:
    word_list_letters = ''.join(word_list)
    alphabet = string.ascii_uppercase
    if difficulty == 1:
        filler = [letter for letter in alphabet if letter not in word_list_letters]
    elif difficulty == 3:
        filler = word_list_letters
    else:
        filler = alphabet
    matrix[idx][col] = random.choice(filler)
```

Die „List Comprehension” für die Schwierigkeitsstufe 1 wählt die Buchstaben aus, die sich nicht in der Liste der Worte befinden und erstellt eine neue Liste nur mit diesen Buchstaben. *random.choice(filler)* wählt aus der jeweils gegebenen Sammlung von Buchstaben dann zufällige aus und schreibt sie an Positionen, die nicht *None* sind.

Die Matrix wird dann mit *print_matrix()* angezeigt.

Beispiele

worte0.txt

Im ersten Beispiel müssen in einem 5x5 Feld die Worte *VOR, RAD, EVA* und *TORF* gefunden werden:

Y	Z	E	V	A
M	U	X	S	T
R	V	O	R	O
A	Y	X	P	R
D	K	G	L	F

worte0 (leicht)

U	W	A	G	B
T	D	P	X	G
O	R	T	A	N
R	A	E	V	A
F	D	V	O	R

worte0 (normal)

A	V	E	T	T
F	F	A	V	O
D	E	E	F	R
A	R	O	V	F
R	R	V	R	O

worte0 (schwer)

Mit ein wenig Eingewöhnung lassen sich die Worte im leichten Feld (Mitte, links unten, rechts oben, rechts mitte) besser finden als im mittleren oder im schweren Feld. Zwar ist *TORF* zunächst gut zu finden (rechts oben), aber durch die vielen gleichen Buchstaben und die umgedrehten Worte *ROV* (mitte unten), DAR (links unten nach oben) und insbesondere AVE (links oben) ist dieses Rätsel anspruchsvoller als die anderen. Das kleine Feld ist aber eher eine Übung, um sicherzustellen, dass das Programm funktioniert als ein richtiges Puzzle.

worte1.txt

Das zweite Beispiel mit 6x6 Feldern ist schon etwas schwieriger:

M	B	Y	K	B	G
I	B	I	G	W	C
N	P	D	E	I	N
F	F	A	D	U	U
O	E	R	M	N	N
A	L	V	F	S	D

worte1 (normal)

Hier sind im normalen Modus die Worte *EIN, DA, ER, INFO, DU* sowie *UND* versteckt.

worte2.txt

Das Problem bei der Datei ist, dass zwei Wörter (*BILDSCHIRM* und *FESTPLATTE*) so lang sind, dass die genau eine Zeile füllen. Sie blockieren dann vertikale Wörter. Zur besseren Sichtbarkeit hier die Darstellung in „schwer” mit und ohne Füllbuchstaben:

```
+---+---+---+---+---+---+---+---+---+---+
| B | I | L | D | S | C | H | I | R | M |
+---+---+---+---+---+---+---+---+---+---+
| O | F | T | L | R | U | S | B | B | T |
+---+---+---+---+---+---+---+---+---+---+
| U | A | L | S | E | R | T | R | S | A |
+---+---+---+---+---+---+---+---+---+---+
| I | U | T | S | U | U | A | A | B | S |
+---+---+---+---+---+---+---+---+---+---+
| C | C | M | L | P | U | A | I | H | T |
+---+---+---+---+---+---+---+---+---+---+
| P | U | S | B | U | U | U | A | P | A |
+---+---+---+---+---+---+---+---+---+---+
| E | S | S | U | A | M | E | M | T | T |
+---+---+---+---+---+---+---+---+---+---+
| R | E | T | U | P | M | O | C | U | U |
+---+---+---+---+---+---+---+---+---+---+
| T | B | F | H | A | D | T | U | A | R |
+---+---+---+---+---+---+---+---+---+---+
| E | T | T | A | L | P | T | S | E | F |
+---+---+---+---+---+---+---+---+---+---+
```

worte2 (schwer)

```
+---+---+---+---+---+---+---+---+---+---+
| B | I | L | D | S | C | H | I | R | M |
+---+---+---+---+---+---+---+---+---+---+
| . | . | . | . | . | U | S | B | . | T |
+---+---+---+---+---+---+---+---+---+---+
| . | . | . | . | . | . | . | . | . | A |
+---+---+---+---+---+---+---+---+---+---+
| . | . | . | . | . | . | . | . | . | S |
+---+---+---+---+---+---+---+---+---+---+
| . | . | . | . | . | . | . | . | . | T |
+---+---+---+---+---+---+---+---+---+---+
| . | . | . | . | . | . | . | . | . | A |
+---+---+---+---+---+---+---+---+---+---+
| . | . | S | U | A | M | . | . | . | T |
+---+---+---+---+---+---+---+---+---+---+
| R | E | T | U | P | M | O | C | . | U |
+---+---+---+---+---+---+---+---+---+---+
| . | . | . | . | . | . | . | . | . | R |
+---+---+---+---+---+---+---+---+---+---+
| E | T | T | A | L | P | T | S | E | F |
+---+---+---+---+---+---+---+---+---+---+
```

worte2 (schwer, ohne Füllbuchstaben)

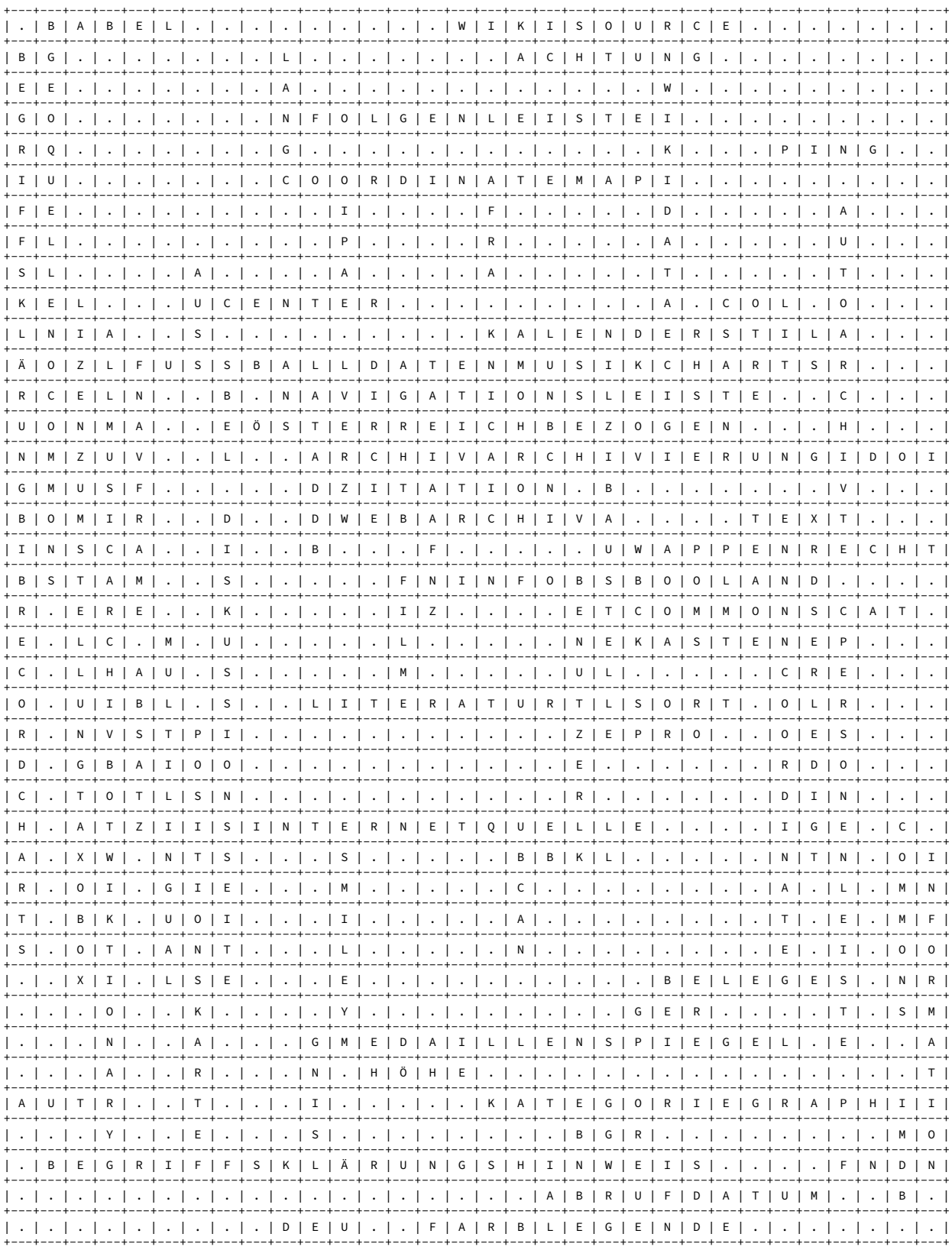
worte3.txt

Das vierte Beispiel ist mit 22x24 Feldern das zweitgrößte, mit langen, schwierigen Worten wie *LEGITIMATION* und *KONJUNKTUR*. Hier im Schwierigkeitsgrad „leicht”, ohne Füllbuchstaben:

```
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| . | . | . | V | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | R | . | . | . | . |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| . | I | . | E | I | N | T | U | I | T | I | O | N | . | . | . | . | . | . | E | . | . | . | . |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| . | N | . | R | . | . | . | . | C | . | . | . | . | . | . | . | . | . | . | V | . | . | . | . |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| . | F | . | S | . | . | . | . | H | . | . | . | . | . | . | . | . | . | . | O | . | . | . | . |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| . | E | . | . | . | . | . | . | R | . | . | . | . | . | . | . | . | . | . | L | . | . | . | . |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| . | K | . | . | . | . | . | . | O | . | . | . | . | . | . | . | . | . | . | U | . | . | E | . |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| . | T | . | . | . | . | . | . | N | . | . | . | . | . | . | . | . | . | . | T | . | . | M | . |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| . | I | . | . | . | . | . | . | I | . | . | . | . | . | . | . | . | . | . | I | . | . | P | . |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| . | O | . | . | . | . | . | . | K | E | M | I | S | S | I | O | N | . | . | O | . | . | A | . |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| . | N | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | N | . | . | T | . |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| . | D | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | H | . |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| . | E | . | . | K | O | N | J | U | N | K | T | U | R | . | . | . | . | . | . | . | . | I | . |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| . | K | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | E | . |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| . | O | . | . | . | . | . | . | . | . | . | . | . | M | O | N | O | G | R | A | M | M | . | . |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| . | R | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| . | A | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| . | T | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | R | E | F | E | R | A | T |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| . | I | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| . | O | . | . | . | . | . | . | . | . | . | . | L | E | G | I | T | I | M | A | T | I | O | N |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| . | N | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

worte4.txt

Beispiel fünf ist das anspruchsvollste Beispiel:



Es hat eine Fläche von 40x32 Feldern die 77 Worte beinhalten sollen. Die abgebildete Variante ist der Schwierigkeitsgrad „leicht”, ohne Füllwörter, und selbst so ist es schwer, das Rätsel zu debuggen, da hier auch Abkürzungen wie *FN*, *FNZ* oder *COL* vorkommen oder ausgefallene Worte wie *GNIS* und *PERSONENLEISTE*.

worte5.txt

Hier ist das Problem, dass dadurch, dass nur ein Wort in der Liste vorhanden ist, dies sich mehrfach auffinden lässt wenn man die dritte Schwierigkeit auswählt. Hier müsste eigentlich eine zusätzliche Funktion implementiert werden, die darauf achtet, dass das gesuchte, kurze Wort *DAS* tatsächlich nur an der gewünschten Stelle auftaucht. Im Grunde ist hier nur unsere erste Schwierigkeitsstufe, die ausschließlich andere Buchstaben, als die gesuchten nimmt, sinnvoll. Viel Spaß! :)

J	O	M	W	E	G	V	K	Z	O	C	Q	R	X	O	N	W	W	I	X	G	P	T	H	B	U	N	W	Q	M
B	K	O	P	J	Z	F	Y	J	P	F	I	P	G	K	X	O	L	Z	B	R	F	N	R	V	T	U	X	K	Y
N	M	L	W	I	H	Y	U	G	W	R	H	C	W	H	R	H	P	M	N	V	Y	Z	K	Z	X	Z	Q	O	W
Z	U	H	H	N	E	H	F	L	I	M	I	C	V	M	M	G	F	R	Z	H	H	P	D	A	S	Y	Q	U	V
W	G	O	G	P	T	K	Z	R	J	W	C	Z	W	R	C	J	H	F	Q	B	X	R	L	H	O	X	B	C	V
W	H	W	V	E	Y	C	N	L	F	I	U	F	Q	Y	J	F	C	Y	X	T	I	Q	C	V	T	B	O	G	B
E	L	Z	J	E	C	N	C	E	J	W	M	K	M	P	Q	Q	Z	B	H	V	V	Z	L	G	K	Z	K	B	F
T	R	V	B	X	V	T	K	C	L	M	G	V	G	O	I	Z	K	F	H	U	C	C	L	P	O	Q	Y	I	I
X	Z	N	O	B	L	V	G	N	N	K	N	B	V	F	Q	O	I	I	G	O	N	K	K	R	Q	G	P	K	T
T	Z	H	Z	Y	M	H	R	G	L	L	M	O	P	E	B	Z	O	C	L	H	Y	O	O	W	R	M	E	K	P
L	I	Q	Z	Q	R	R	N	U	L	J	G	Q	T	K	G	K	Y	P	U	B	W	F	O	T	I	R	K	B	Q
T	R	E	V	B	R	F	X	X	P	L	Y	K	N	J	F	L	M	Y	U	C	U	Y	Y	P	X	E	J	T	Y
Y	O	J	I	W	E	E	T	W	Y	O	J	K	J	F	H	M	F	T	O	N	J	E	B	V	R	Y	X	V	J
C	J	P	Z	L	R	W	U	J	L	K	H	W	E	B	Q	Y	Y	C	E	C	N	I	P	N	Q	W	L	N	F
Y	X	W	P	V	N	U	C	T	R	O	Q	F	R	E	C	Y	T	X	X	Q	R	E	W	J	Y	P	F	C	I
F	Y	P	I	Z	F	J	T	X	W	V	T	F	L	M	G	T	C	G	Y	C	C	T	U	T	U	I	Q	Q	R
L	L	T	R	H	L	X	G	M	R	U	I	K	M	Z	P	K	G	B	Y	H	N	M	C	E	X	B	K	P	O
C	Q	C	C	N	C	B	Y	T	W	G	W	M	M	H	M	F	Y	Y	J	F	F	G	V	X	J	I	H	X	I
Z	I	G	U	K	U	J	I	L	L	I	V	N	C	T	B	W	Y	T	U	L	V	Y	G	J	C	O	U	T	T
C	I	G	W	Y	X	W	F	B	I	Y	F	P	F	F	X	X	K	U	P	B	C	G	U	R	X	T	U	V	P
P	J	X	Q	N	N	U	M	Z	T	B	J	P	C	V	F	H	U	V	L	H	B	V	P	U	Z	C	O	H	R
H	Q	T	O	J	I	K	P	R	Z	F	V	U	E	K	L	M	Q	H	E	N	N	V	H	C	Y	E	T	T	Z
V	F	B	W	V	Y	B	Q	U	P	V	X	Y	C	E	U	G	H	C	B	O	U	O	O	X	K	Q	C	Q	C
M	Z	N	I	I	G	L	Y	J	J	Y	X	R	X	U	N	B	O	L	W	Q	W	X	L	L	Y	R	J	X	O
K	X	R	N	V	P	G	H	J	X	R	L	I	B	W	R	U	F	C	O	P	M	V	K	W	P	P	G	U	K
U	E	G	Y	L	U	H	M	M	M	F	I	U	Y	I	E	U	Q	C	K	I	O	G	V	H	R	V	E	Y	Z
B	Z	N	L	V	G	T	V	G	U	F	W	G	L	F	V	N	P	T	M	Z	B	C	W	Y	B	Q	O	G	H
P	E	Z	B	J	C	Q	Y	K	K	B	B	J	U	J	K	M	N	U	W	L	V	H	V	H	N	G	B	K	Q
U	C	Q	G	P	U	P	J	U	Z	K	G	G	E	N	W	H	X	H	F	X	W	O	N	X	P	G	M	G	X
J	T	U	Q	V	E	B	C	Y	X	M	C	T	T	O	W	K	L	I	B	X	Y	J	E	R	N	G	G	J	Q

Quellcode

```
from pathlib import Path
import random
import string

"""
Kriterien:
    1. Leicht: Nur gewohnte Laufrichtung, ausschließlich andere Buchstaben
    2. Mittel: Nur gewohnte Laufrichtung, alle Buchstaben des Alphabets
    3. Schwer: Auch umgedrehte Laufrichtung, nur Buchstaben der Suchbegriffe
"""

def read_input(filename='worte0.txt'):
    """ Beispieldatei einlesen
    Die Zeilen in Integer und List umwandeln und Zeilenumbrüche mit .strip() entfernen.
    Default ist das Aufgabenbeispiel parkplatz0.txt.
    """
    file = Path('beispieldaten', filename)
    with open(file, 'r') as file_in:
        dimensions = [int(num) for num in file_in.readline().strip().split()]
        word_count = file_in.readline().strip()
        word_list = [line.strip() for line in file_in.readlines()]

    return dimensions, word_list, word_count


def make_matrix(dimensions):
    matrix = [None] * dimensions[0]
    for idx, row in enumerate(matrix):
        matrix[idx] = [None] * dimensions[1]

    return matrix


def create_game(word_list, matrix):
    """ Erstelle ein neues Rechteck mit benutzerdefiniertem Schwierigkeitsgrad
    """
    difficulty = int(input("Wähle Schwierigkeitsgrad (1,2 oder 3):"))
    for idx, word in enumerate(word_list):
        if idx % 2 == 1:
            word_space_down = find_word_space_down(word, matrix)
            if word_space_down:
                write_word_down(matrix, word_space_down[0], word_space_down[1], word, difficulty)
                continue
        start_row, start_column = find_word_space_right(word, matrix)
        write_word_right(matrix, start_row, start_column, word, difficulty)
    fill_empty_spaces(matrix, difficulty, word_list)


def fill_empty_spaces(matrix, difficulty, word_list):
    """ Fülle die leeren Felder mit Buchstaben
    """
    for idx, row in enumerate(matrix):
        for col, _ in enumerate(row):
```

```
            if matrix[idx][col] is None:
                word_list_letters = ''.join(word_list)
                alphabet = string.ascii_uppercase
                if difficulty == 1:
                    filler = [letter for letter in alphabet if letter not in word_list_letters]
                elif difficulty == 3:
                    filler = word_list_letters
                else:
                    filler = alphabet
                matrix[idx][col] = random.choice(filler)
    return matrix


def find_word_space_right(word, matrix):
    """ Finde eine Zeile mit genug freiem Platz für das gegebene Wort
    """
    max_offset = len(matrix[0]) - len(word)
    random_column = random.randint(0, max_offset)
    random_rows = list(range(len(matrix)))
    random.shuffle(random_rows)
    for idx in random_rows:
        count = 0
        for column in range(random_column, len(matrix[0])):
            if matrix[idx][column] is not None:
                count = 0
            else:
                count += 1
                if count == len(word):
                    start_column = column - (len(word) - 1)
                    return idx, start_column


def find_word_space_down(word, matrix):
    """ Finde eine Spalte mit genug freiem Platz für das gegebene Wort
    """
    max_offset = len(matrix) - len(word)
    random_row = random.randint(0, max_offset)
    random_cols = list(range(len(matrix[0])))
    random.shuffle(random_cols)
    for idx in random_cols:
        count = 0
        for row in range(random_row, len(matrix)):
            if matrix[row][idx] is not None:
                count = 0
            else:
                count += 1
                if count == len(word):
                    start_row = row - (len(word) - 1)
                    return start_row, idx


def write_word_right(matrix, start_row, start_column, word, difficulty):
    """ Schreibe ein Wort in die Matrix
    """
    if difficulty == 3 and random.random() > 0.5:
        word = list(word)
        word.reverse()
```

```
    for letter in word:
        matrix[start_row][start_column] = letter
        start_column += 1


def write_word_down(matrix, start_row, start_column, word, difficulty):
    """ Schreibe ein Wort von oben nach unten in die Matrix
    """
    if difficulty == 3 and random.random() > 0.5:
        word = list(word)
        word.reverse()

    for letter in word:
        matrix[start_row][start_column] = letter
        start_row += 1


def print_matrix(matrix):
    print('+', end='')
    for _ in matrix[0]:
        print('---+', end='')
    print()
    for row in matrix:
        print('| ', end='')
        for letter in row:
            print(letter, end=' | ')
        print()
        print('+', end='')
        for _ in row:
            print('---+', end='')
        print()


def run_main():
    dimensions, word_list, word_count = read_input()
    matrix = make_matrix(dimensions)
    create_game(word_list, matrix)
    print_matrix(matrix)


if __name__ == '__main__':
    run_main()
```