Aufgabe 5: Hüpfburg

Team-ID: 00543

Team: pip install knowledge (Imanuel Fehse)

Bearbeiter dieser Aufgabe: Imanuel Fehse

15. November 2022

Inhaltsverzeichnis

Aufgabe

Überlege dir, wie man feststellen kann, ob Sasha und Mika einen Parcours erfolgreich absolvieren können. Schreibe ein Programm das einen Parcours einliest und dann ausgibt, ob dieser Parcours erfolgreich absolviert werden kann. Falls es möglich ist, soll außerdem eine erfolgreiche Folge von Sprüngen für Sasha und Mika ausgegeben werden.

Lösungsidee

Das Programm erstellt aus der gegebenen Datei ein Register, in dem vermerkt ist, an welche Positionen der Spieler im nächsten Schritt von seiner aktuellen Position kommen kann.
Diese möglichen aktuellen Positionen speichert das Programm ab, prüft ob es eine Möglichkeit gibt, an der sich die beiden Spieler (Sasha und Mika) treffen können. Falls ja, gebe die Position aus, an der sie sich treffen können und wie viele Züge (Sprünge) die beiden bis dahin gemacht haben.

Umsetzung

Im ersten Schritt wird mithilfe eines Userinputs die Datei eingelesen. Falls die Datei existiert, fährt das Programm fort, falls nicht, frägt das Programm erneut nach einer Datei.
Aus dieser Datei liest das Programm die Daten ein und formatiert sie so, dass jede Verbindung mit der der Spieler von einem Feld zum nächsten kommt separat in einer Liste steht. Diese Liste wird dann mittels eines Unterprogramms in ein Dictionary umgewandelt, aus dem das Programm einfacher an die folgenden Felder für eine aktuelle Position erhalten kann.
Nun wird für 100 Züge/Sprünge (da es sehr unwahrscheinlich ist, dass die Spieler mehr als 100 Felder springen können, da ihre Kräfte irgendwann zu Ende sein werden) geprüft, ob es eine Lösung gibt. Dazu überprüft das Programm für jede aktuell mögliche Position, welche Positionen der Spieler im nächsten Schritt erreichen kann und speichert diese möglichen, nächsten Positionen ab, damit diese für den nächsten Schritt durch dasselbe Verfahren genutzt werden können.
Nach jedem Schritt wird zusätzlich überprüft, ob es eine Möglichkeit gibt, dass die beiden Spieler sich auf demselben Feld befinden.
Falls es ein solches Feld gibt, gibt das Programm dieses Feld sowie die dafür benötigten Schritte/Sprünge der Spieler aus. Falls nicht, gibt das Programm aus, dass es keine Lösung gibt, an der sich die beiden Spieler treffen.

Beispiele

Dies sind die Ausgaben, die das Programm in der Konsole macht, nachdem das Programm den Namen und Speicherort der Datei erhalten hat.

huepfburg0.txt

Solution found after 3 steps
Solution: 10

huepfburg1.txt

No solution found

huepfburg2.txt

Solution found after 8 steps
Solution: 51

huepfburg3.txt

No solution found

huepfburg4.txt

Solution found after 16 steps
Solution: 12

Wichtige Teile des Codes

```
# get following connections
for item in x:
    # get for every possible last position every possible current positions and add them to
list
    for ix in [item]:
        try:
            p = connections[ix]
        except KeyError:
            continue
        for subx in connections[ix]:
            dx.append(subx)

for item in y:
    # get for every possible last position every possible current positions and add them to
list
    for iy in [item]:
        try:
            p = connections[iy]
        except KeyError:
            continue
        for suby in connections[iy]:
            dy.append(suby)

# save possible positions at current step
x_list.append(dx)
y_list.append(dy)
```

Ermittelt alle möglichen Positionen der Spieler beim aktuellen Zug und speichert diese.

```
def check_solution(x_pos, y_pos):
    for x in x_pos:
        for y in y_pos:
            if x == y:
                return True, x
    return False, None
```

Überprüft, ob die Spieler sich an der gleichen Position befinden können.