Aufgabe 2: Simultane Labyrinthe

Teilnahme-ID: 74749

Bearbeiter/-in dieser Aufgabe:
Christian Krause

28. Januar 2025

Inhaltsverzeichnis

Anleitung: Trage oben in den Zeilen 8 bis 10 die Aufgabennummer, die Teilnahme-ID und die/den Bearbeiterin/Bearbeiter dieser Aufgabe mit Vor- und Nachnamen ein. Vergiss nicht, auch den Aufgabennamen anzupassen (statt „LaTeX-Dokument“)!

Dann kannst du dieses Dokument mit deiner LaTeX-Umgebung übersetzen.

Die Texte, die hier bereits stehen, geben ein paar Hinweise zur Einsendung. Du solltest sie aber in deiner Einsendung wieder entfernen!

1 Lösungsidee

. . . Einleitung

Annahmen: Das Labyrinth ist von Wänden umgeben . . .

1.1 Optimale Lösung

Zuerst habe ich mich damit beschäftigt, einen Algorithmus zu entwickeln, der eine optimale Lösung für die Aufgabenstellung berechnen kann. Eine optimale Lösung ist hier die kürzeste Anweisungssequenz, die Anton und Bea ins Ziel bringt. In diesem Abschnitt wird direkt der Aufgabenteil b) behandelt (TOOD besser formulieren).

TODO vlt section Modellierung hier?

Jeder Zustand, in dem sich Anton und Bea befinden, kann als Tupel der jeweiligen Koordinaten dargestellt werden: $((x_0, y_0), (x_1, y_1))$. (x_0, y_0) ist hier die Position von Anton (der sich in Labyrinth 1 befindet), (x_1, y_1) beschreibt die Position von Bea im zweiten Labyrinth. Am Anfang herscht der Zustand $S_0 = ((0, 0), (0, 0))$, da sich beide auf ihrem Startfeld befinden. Chris (uii, er heißt ja fast wie ich!!!) kann nun vier mögliche Anweisungen geben, die einen neuen Zustand herbeiführen würden. Wenn Anton und Bea beide ihr Zielfeld erreicht haben, befindet wir uns in dem Zustand $S_{end} = ((n-1, m-1), (n-1, m-1))$. Formal können die verschiedenen Positionen von Anton und Bea als Graph dargestellt werden, dessen Knoten alle Möglichen Zustände sind.

$$V = \{((x_1, y_1), (x_1, y_1)) \mid 0 \leq x_1, x_2 < nand 0 \leq y_1, y_2 < m\}$$

(TODO muss man hier n und m vertauschen?) Von jedem Knoten gehen vier Kanten aus, eine für jede Anweisung, die Chris geben könnte. TODO

TODO Grube

Alle Kanten haben die Länge 1, da sie einer Anweisung entsprechen.

TODO das vlt später irgendwo: In der Implementierung können alle Schleifen, also Kanten die einen Knoten mit sich selbst verbinden, ignoriert werden, da sie mit Anweisungen zusammenhängen, die den Zustand nicht ändern (z.B. da Anton und Bea beide gegen eine Wand laufen).

Jeder Pfad von einem Knoten A zu einem anderen Knoten B repräsentiert eine Sequenz von Anweisungen, die Anton und Bea von ihren Positionen bei A zu ihren Positionen bei B bringt. Die Länge eines solchen Pfades entspricht der Anzahl der durchlaufenen Anweisungen.
Der kürzeste Pfad von S_0 zu S_end entspricht also einer kürzesten Anweisungssequenz, die Anton und Bea ins Ziel bringt. Die Aufgabenstellung lässt sich also darauf reduzieren, den kürzesten Pfad von S_0 zu S_{end} in dem oben beschriebenen Graph zu finden.
Dies kann zum Beispiel mit dem bekannten Dijstra-Algorithmus (TODO Name richtig?) (TODO Quelle) bewerkstelligt werden.

1.1.1 Laufzeit

Wenn Dijkstra mit einer Liste oder einem Array implementiert wird, entspricht die Zeitkomplexität dem Quadrat der Knotenanzahl: $O(\|E\| + \|V\|^2) = O(\|V\|^2)$. (TODO Quelle Wikipedia). Der oben beschriebene Graph besitzt einen Knoten für jede Mögliche Kombination an Positionen von Anton und Bea. Bea und Anton können jeweils $n \cdot m$ verschiedene Positionen einnehmen, d.h. insgesamt hat der Graph $\|V\| = n^2m^2$ Knoten. Damit entspricht die worst-case Laufzeit für die naive Implementierung $O(n^4 \cdot m^4)$.

2 Erweiterungen

TODO gibt es mehrere mögliche kürzeste Anweisungssequenzen