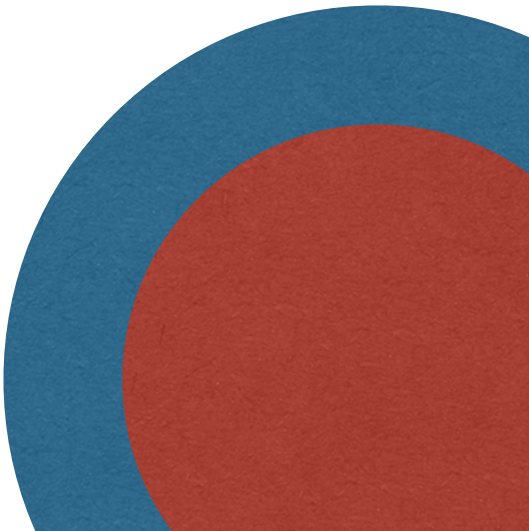
42. Bundeswettbewerb Informatik

Team Deutscher Qualitätscode
Max Eckstein, Nina Hochecker
Team-ID: 00538



Aufgabe 3: Zauberschule

Inhaltsverzeichnis

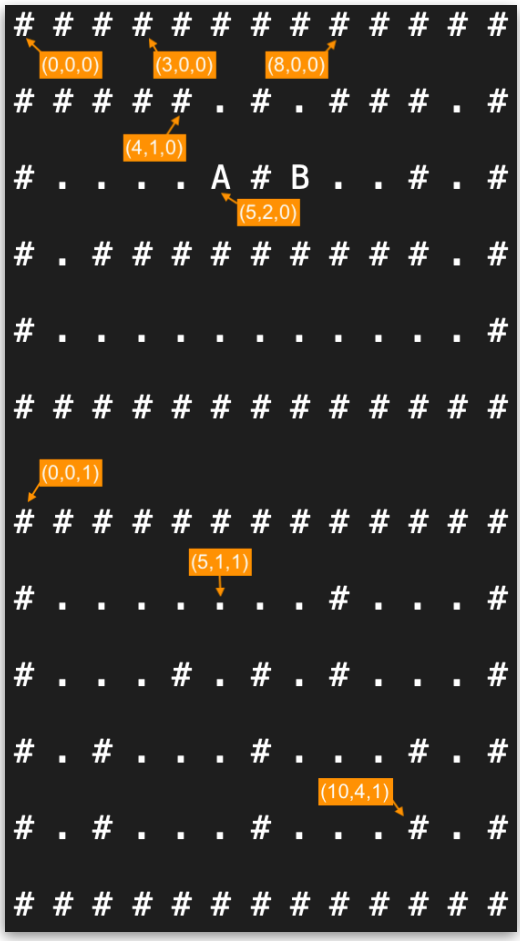
Lösungsidee

Unsere Idee ist es, die Karte der zwei Stockwerke als Graph mit gewichteten bidirektionalen Kanten zwischen den verschiedenen Knoten darzustellen. Dabei soll jeder freie Punkt (also jeder „." aus der Datei) einen Knoten darstellen. Ein Punkt kann Kanten zu allen 4 Punkten um ihn herum haben, welche das Gewicht 1 haben sollen. Außerdem soll er eine Kante zu dem Punkt „über" oder „unter" ihm haben, falls dieser überhaupt existiert (an der Stelle könnte auch eine Wand sein). Diese Kante soll das Gewicht 3 (-> 3 Sekunden Wegzeit) haben. Ein Graphensuch-Algorithmus (in unserem Fall A*) findet nun in diesem Graphen den schnellsten Weg vom Punkt A zum Punkt B.
Wir haben uns für den A*-Algorithmus entschieden, weil er im Gegensatz zu Dijkstra's Algorithmus zielgerichtet nach Wegen zum Ziel sucht (in dem er prüft, ob er sich grade prinzipiell von seinem Ziel entfernt und dies einkalkuliert), und nicht in alle Richtungen ermittelt. Er findet somit mit einer hohen Wahrscheinlichkeit also schneller das Ziel als Dijkstra's Algorithmus oder andere vergleichbare Algorithmen.

Umsetzung

Als Erstes wird die übergebene Datei eingelesen, in ein zweidimensionales Array von Buchstaben konvertiert. Dieses Array wird nach den Buchstaben „A" und „B" durchsucht. Die Position dieser Buchstaben wird als **Punkt** (mit x_1-, x_2- und x_3-Koordinate) gesichert.

- Die x_1-Koordinate entspricht dem Index in der zweiten Ebene des zweidimensionalen Arrays. -> Erste Koordinate innerhalb einer Etage
- Die x_2-Koordinate entspricht dem Index in der ersten Ebene des Arrays modulo Anzahl der Zeilen in einer Etage. -> Zweite Koordinate innerhalb einer Etage
- Die x_3-Koordinate gibt mit 0 oder 1 an, ob der Punkt sich in der ersten oder zweiten Etage befindet. -> Etage



Das Bild rechts zeigt den Zusammenhang nochmals. Wenn Start- und Endpunkt gefunden sind, übernimmt der A*-Algorithmus die Suche nach der optimalen Route. Der A*-Algorithmus ist ein schon bestehender Graphen-Algorithmus, welcher die Fähigkeit hat, den schnellsten

Weg von Start zum Ziel zu finden, ohne dabei alle verfügbaren Punkte, und Routen über diese, testen zu müssen. Dazu braucht er eine „heuristische Funktion“, welche zu jedem Punkt angeben kann, wie groß der Abstand zum Ziel noch mindestens sein muss. In unserem Fall kann hierfür einfach die gewichtete (x_3-Koordinate mit Faktor drei) Differenz der einzelnen Koordinaten benutzt werden, um die Mindest-Zeit zum Ziel zu errechnen. Der Algorithmus selbst ist von Wikipedia abgeschrieben und in die Programmiersprache Swift übersetzt. Die einzige Ergänzung, die wir noch vorgenommen haben ist, dass die Kanten von / zu einem Punkt erst dynamisch generiert werden, wenn der Algorithmus diesen Punkt gerade besucht. So kann verhindert werden, dass anfangs sehr viele Kanten errechnet werden müssen. Dieses dynamische generieren von Kanten übernimmt die Funktion **sucheNachVerbindungen**. Dazu prüft diese, welche der 4 Felder um den Punkt herum begehbar sind (also „.“ beinhalten und kein „#“). Außerdem wird noch das Feld „über“ oder „unter“ dem Punkt (also im anderen Stockwerk) von der Funktion betrachtet. Der A*-Algorithmus kann nun die schnellste Route bestimmen. Abschließend werden die Ergebnisse noch in die Konsole ausgegeben.

Ausführung des Algorithmus:
In der Kommandozeile wird (auf MacOS) “./main” eingegeben, gefolgt von einem Leerzeichen und der Nummer der Datei, die ausgelesen werden soll. Bei „./main 4“ wird beispielsweise die Datei „../Daten/zauberschule4.txt“ ausgelesen.

Beispiele

Datei	Dauer	Route
zauberschule0.txt	8 Sekunden	Punkt(6|10) auf dem ersten Stockwerk Punkt(6|10) auf dem zweiten Stockwerk Punkt(7|10) auf dem zweiten Stockwerk Punkt(8|10) auf dem zweiten Stockwerk Punkt(8|10) auf dem ersten Stockwerk
zauberschule1.txt	4 Sekunden	Punkt(14|8) auf dem ersten Stockwerk Punkt(13|8) auf dem ersten Stockwerk Punkt(12|8) auf dem ersten Stockwerk Punkt(12|7) auf dem ersten Stockwerk Punkt(12|6) auf dem ersten Stockwerk
zauberschule2.txt	14 Sekunden	Punkt(22|4) auf dem ersten Stockwerk Punkt(23|4) auf dem ersten Stockwerk Punkt(24|4) auf dem ersten Stockwerk Punkt(24|4) auf dem zweiten Stockwerk Punkt(25|4) auf dem zweiten Stockwerk Punkt(26|4) auf dem zweiten Stockwerk Punkt(26|4) auf dem ersten Stockwerk Punkt(26|5) auf dem ersten Stockwerk Punkt(26|6) auf dem ersten Stockwerk Punkt(27|6) auf dem ersten Stockwerk Punkt(28|6) auf dem ersten Stockwerk
zauberschule3.txt	28 Sekunden	Punkt(4|28) auf dem ersten Stockwerk Punkt(4|29) auf dem ersten Stockwerk Punkt(4|30) auf dem ersten Stockwerk Punkt(5|30) auf dem ersten Stockwerk Punkt(6|30) auf dem ersten Stockwerk Punkt(6|29) auf dem ersten Stockwerk Punkt(6|28) auf dem ersten Stockwerk Punkt(7|28) auf dem ersten Stockwerk Punkt(8|28) auf dem ersten Stockwerk Punkt(9|28) auf dem ersten Stockwerk Punkt(10|28) auf dem ersten Stockwerk Punkt(10|29) auf dem ersten Stockwerk Punkt(10|30) auf dem ersten Stockwerk Punkt(11|30) auf dem ersten Stockwerk Punkt(12|30) auf dem ersten Stockwerk Punkt(13|30) auf dem ersten Stockwerk Punkt(14|30) auf dem ersten Stockwerk Punkt(14|29) auf dem ersten Stockwerk Punkt(14|28) auf dem ersten Stockwerk Punkt(15|28) auf dem ersten Stockwerk Punkt(16|28) auf dem ersten Stockwerk Punkt(17|28) auf dem ersten Stockwerk Punkt(18|28) auf dem ersten Stockwerk Punkt(19|28) auf dem ersten Stockwerk Punkt(20|28) auf dem ersten Stockwerk Punkt(20|27) auf dem ersten Stockwerk Punkt(20|26) auf dem ersten Stockwerk Punkt(21|26) auf dem ersten Stockwerk Punkt(22|26) auf dem ersten Stockwerk

Datei	Dauer	Route
zauberschule4.txt	84 Sekunden	Punkt(74|68) auf dem ersten Stockwerk Punkt(74|67) auf dem ersten Stockwerk Punkt(74|66) auf dem ersten Stockwerk Punkt(73|66) auf dem ersten Stockwerk Punkt(72|66) auf dem ersten Stockwerk Punkt(72|66) auf dem zweiten Stockwerk Punkt(72|65) auf dem zweiten Stockwerk Punkt(72|64) auf dem zweiten Stockwerk Punkt(71|64) auf dem zweiten Stockwerk Punkt(70|64) auf dem zweiten Stockwerk Punkt(69|64) auf dem zweiten Stockwerk Punkt(68|64) auf dem zweiten Stockwerk Punkt(68|63) auf dem zweiten Stockwerk Punkt(68|62) auf dem zweiten Stockwerk Punkt(68|62) auf dem ersten Stockwerk Punkt(67|62) auf dem ersten Stockwerk Punkt(66|62) auf dem ersten Stockwerk Punkt(65|62) auf dem ersten Stockwerk Punkt(64|62) auf dem ersten Stockwerk Punkt(63|62) auf dem ersten Stockwerk Punkt(62|62) auf dem ersten Stockwerk Punkt(61|62) auf dem ersten Stockwerk Punkt(60|62) auf dem ersten Stockwerk Punkt(60|63) auf dem ersten Stockwerk Punkt(60|64) auf dem ersten Stockwerk Punkt(59|64) auf dem ersten Stockwerk Punkt(58|64) auf dem ersten Stockwerk Punkt(58|63) auf dem ersten Stockwerk Punkt(58|62) auf dem ersten Stockwerk Punkt(57|62) auf dem ersten Stockwerk Punkt(56|62) auf dem ersten Stockwerk Punkt(55|62) auf dem ersten Stockwerk Punkt(54|62) auf dem ersten Stockwerk Punkt(53|62) auf dem ersten Stockwerk Punkt(52|62) auf dem ersten Stockwerk Punkt(51|62) auf dem ersten Stockwerk Punkt(50|62) auf dem ersten Stockwerk Punkt(50|62) auf dem zweiten Stockwerk Punkt(50|63) auf dem zweiten Stockwerk Punkt(50|64) auf dem zweiten Stockwerk Punkt(49|64) auf dem zweiten Stockwerk Punkt(48|64) auf dem zweiten Stockwerk Punkt(48|63) auf dem zweiten Stockwerk Punkt(48|62) auf dem zweiten Stockwerk Punkt(47|62) auf dem zweiten Stockwerk Punkt(46|62) auf dem zweiten Stockwerk Punkt(46|62) auf dem ersten Stockwerk Punkt(46|61) auf dem ersten Stockwerk Punkt(46|60) auf dem ersten Stockwerk Punkt(45|60) auf dem ersten Stockwerk Punkt(44|60) auf dem ersten Stockwerk Punkt(44|59) auf dem ersten Stockwerk Punkt(44|58) auf dem ersten Stockwerk Punkt(43|58) auf dem ersten Stockwerk Punkt(42|58) auf dem ersten Stockwerk Punkt(41|58) auf dem ersten Stockwerk Punkt(40|58) auf dem ersten Stockwerk Punkt(40|57) auf dem ersten Stockwerk Punkt(40|56) auf dem ersten Stockwerk Punkt(39|56) auf dem ersten Stockwerk Punkt(38|56) auf dem ersten Stockwerk Punkt(38|55) auf dem ersten Stockwerk Punkt(38|54) auf dem ersten Stockwerk Punkt(37|54) auf dem ersten Stockwerk Punkt(36|54) auf dem ersten Stockwerk Punkt(35|54) auf dem ersten Stockwerk Punkt(34|54) auf dem ersten Stockwerk Punkt(34|54) auf dem zweiten Stockwerk Punkt(34|55) auf dem zweiten Stockwerk Punkt(34|56) auf dem zweiten Stockwerk Punkt(33|56) auf dem zweiten Stockwerk Punkt(32|56) auf dem zweiten Stockwerk Punkt(32|56) auf dem ersten Stockwerk

Datei	Dauer	Route
zauberschule5.txt	124 Sekunden	Punkt(14|168) auf dem ersten Stockwerk Punkt(14|168) auf dem zweiten Stockwerk Punkt(14|169) auf dem zweiten Stockwerk Punkt(14|170) auf dem zweiten Stockwerk Punkt(15|170) auf dem zweiten Stockwerk Punkt(16|170) auf dem zweiten Stockwerk Punkt(16|171) auf dem zweiten Stockwerk Punkt(16|172) auf dem zweiten Stockwerk Punkt(17|172) auf dem zweiten Stockwerk Punkt(18|172) auf dem zweiten Stockwerk Punkt(19|172) auf dem zweiten Stockwerk Punkt(20|172) auf dem zweiten Stockwerk Punkt(21|172) auf dem zweiten Stockwerk Punkt(22|172) auf dem zweiten Stockwerk Punkt(22|172) auf dem ersten Stockwerk Punkt(23|172) auf dem ersten Stockwerk Punkt(24|172) auf dem ersten Stockwerk Punkt(25|172) auf dem ersten Stockwerk Punkt(26|172) auf dem ersten Stockwerk Punkt(27|172) auf dem ersten Stockwerk Punkt(28|172) auf dem ersten Stockwerk Punkt(29|172) auf dem ersten Stockwerk Punkt(30|172) auf dem ersten Stockwerk Punkt(30|173) auf dem ersten Stockwerk Punkt(30|174) auf dem ersten Stockwerk Punkt(30|174) auf dem zweiten Stockwerk Punkt(31|174) auf dem zweiten Stockwerk Punkt(32|174) auf dem zweiten Stockwerk Punkt(33|174) auf dem zweiten Stockwerk Punkt(34|174) auf dem zweiten Stockwerk Punkt(35|174) auf dem zweiten Stockwerk Punkt(36|174) auf dem zweiten Stockwerk Punkt(37|174) auf dem zweiten Stockwerk Punkt(38|174) auf dem zweiten Stockwerk Punkt(38|173) auf dem zweiten Stockwerk Punkt(38|172) auf dem zweiten Stockwerk Punkt(38|171) auf dem zweiten Stockwerk Punkt(38|170) auf dem zweiten Stockwerk Punkt(39|170) auf dem zweiten Stockwerk Punkt(40|170) auf dem zweiten Stockwerk Punkt(41|170) auf dem zweiten Stockwerk Punkt(42|170) auf dem zweiten Stockwerk Punkt(43|170) auf dem zweiten Stockwerk Punkt(44|170) auf dem zweiten Stockwerk Punkt(44|170) auf dem ersten Stockwerk Punkt(44|169) auf dem ersten Stockwerk Punkt(44|168) auf dem ersten Stockwerk Punkt(44|167) auf dem ersten Stockwerk Punkt(44|166) auf dem ersten Stockwerk Punkt(44|165) auf dem ersten Stockwerk Punkt(44|164) auf dem ersten Stockwerk Punkt(44|163) auf dem ersten Stockwerk Punkt(44|162) auf dem ersten Stockwerk Punkt(44|162) auf dem zweiten Stockwerk Punkt(45|162) auf dem zweiten Stockwerk Punkt(46|162) auf dem zweiten Stockwerk Punkt(47|162) auf dem zweiten Stockwerk Punkt(48|162) auf dem zweiten Stockwerk Punkt(48|162) auf dem ersten Stockwerk Punkt(49|162) auf dem ersten Stockwerk Punkt(50|162) auf dem ersten Stockwerk Punkt(51|162) auf dem ersten Stockwerk Punkt(52|162) auf dem ersten Stockwerk Punkt(52|161) auf dem ersten Stockwerk Punkt(52|160) auf dem ersten Stockwerk Punkt(52|159) auf dem ersten Stockwerk Punkt(52|158) auf dem ersten Stockwerk Punkt(53|158) auf dem ersten Stockwerk Punkt(54|158) auf dem ersten Stockwerk Punkt(54|157) auf dem ersten Stockwerk Punkt(54|156) auf dem ersten Stockwerk Punkt(54|155) auf dem ersten Stockwerk Punkt(54|154) auf dem ersten Stockwerk Punkt(55|154) auf dem ersten Stockwerk Punkt(56|154) auf dem ersten Stockwerk Punkt(56|154) auf dem zweiten Stockwerk Punkt(57|154) auf dem zweiten Stockwerk Punkt(58|154) auf dem zweiten Stockwerk Punkt(59|154) auf dem zweiten Stockwerk Punkt(60|154) auf dem zweiten Stockwerk Punkt(60|154) auf dem ersten Stockwerk Punkt(60|155) auf dem ersten Stockwerk Punkt(60|156) auf dem ersten Stockwerk Punkt(60|157) auf dem ersten Stockwerk Punkt(60|158) auf dem ersten Stockwerk Punkt(61|158) auf dem ersten Stockwerk Punkt(62|158) auf dem ersten Stockwerk Punkt(63|158) auf dem ersten Stockwerk Punkt(64|158) auf dem ersten Stockwerk Punkt(65|158) auf dem ersten Stockwerk Punkt(66|158) auf dem ersten Stockwerk Punkt(66|158) auf dem zweiten Stockwerk Punkt(67|158) auf dem zweiten Stockwerk Punkt(68|158) auf dem zweiten Stockwerk Punkt(69|158) auf dem zweiten Stockwerk Punkt(70|158) auf dem zweiten Stockwerk Punkt(70|158) auf dem ersten Stockwerk Punkt(71|158) auf dem ersten Stockwerk Punkt(72|158) auf dem ersten Stockwerk Punkt(73|158) auf dem ersten Stockwerk Punkt(74|158) auf dem ersten Stockwerk Punkt(74|157) auf dem ersten Stockwerk Punkt(74|156) auf dem ersten Stockwerk Punkt(73|156) auf dem ersten Stockwerk Punkt(72|156) auf dem ersten Stockwerk

Quellcode

```
import Foundation

struct Punkt: Hashable {
    let x1: Int
    let x2: Int
    let x3: Int
}

// Einlesen der externen Datei
let dateiNummer: String = CommandLine.arguments.last ?? "-"
let dateiNummerInt: Int = Int(dateiNummer) ?? 0
let pfad: URL = URL(fileURLWithPath: "../Daten/A3_Zauberschule/zauberschule\
(dateiNummerInt).txt")
guard let text: String = try? String(contentsOf: pfad) else {
    print("Datei konnte nicht gefunden / ausgelesen werden")
    exit(EXIT_FAILURE)
}
var zeilen = text.split(whereSeparator: \.isNewline)
guard let hoeheEinesStockwerksString = zeilen.removeFirst().split(separator: "
").first, let hoeheEinesStockwerks = Int(hoeheEinesStockwerksString) else {
    print("Höhe eines Stockwerks konnte nicht ermittelt werden.")
    exit(EXIT_FAILURE)
}

let stockwerkZeilen: [[Character]] = zeilen.map { zeile in
    return zeile.map({$0})
}

var start: Punkt?
var ziel: Punkt?

var hoehenIndex = 0

for zeile in stockwerkZeilen {
    for char in zeile.enumerated() {
        let x1 = char.offset
        let x2 = (hoehenIndex < hoeheEinesStockwerks) ? hoehenIndex :
(hoehenIndex - hoeheEinesStockwerks)
        let x3 = (hoehenIndex < hoeheEinesStockwerks) ? 0 : 1
        if char.element == "A" {
            start = Punkt(x1: x1, x2: x2, x3: x3)
        } else if char.element == "B" {
            ziel = Punkt(x1: x1, x2: x2, x3: x3)
        }
    }
    hoehenIndex += 1
}

guard let start, let ziel else {
    print("Es konnte kein Start- und/oder Endpunkt gefunden werden.")
    exit(EXIT_FAILURE)
}

// MARK: A* Algorithmus

// Heuristische Funktion (schätzt noch zu brauchende Zeit)
func h(_ punkt: Punkt) -> Int {
    return abs(ziel.x1 - punkt.x1) + abs(ziel.x2 - punkt.x2) + abs(ziel.x3 -
punkt.x3) * 3
}

func rekonstruiereWeg(_ kommtVon: [Punkt: Punkt], _ endPunkt: Punkt) -> [Punkt]
{
```

```
    var pfad = [endPunkt]
    var aktuellerPunkt = endPunkt
    while let neuerPunkt = kommtVon[aktuellerPunkt] {
        aktuellerPunkt = neuerPunkt
        pfad.insert(neuerPunkt, at: 0)
    }
    return pfad
}

func sucheNachVerbindungen(fuerPunkt punkt: Punkt) -> [(Punkt, Int)] {
    let moeglicheOffsets = [
        (1, 0, 0),
        (-1, 0, 0),
        (0, 1, 0),
        (0, -1, 0),
        (0, 0, punkt.x3 == 0 ? 1 : -1)
    ]
    var gefundeneVerbindungen = [(Punkt, Int)]()
    for i in moeglicheOffsets {
        let neueX1 = punkt.x1 + i.0
        let neueX2 = punkt.x2 + i.1
        let neueX3 = punkt.x3 + i.2
        if stockwerkZeilen[neueX2 + neueX3*hoeheEinesStockwerks][neueX1] !=
"#" {
            let neuerPunkt = Punkt(x1: neueX1, x2: neueX2, x3: neueX3)
            gefundeneVerbindungen.append((neuerPunkt, (i.2 != 0) ? 3 : 1))
        }
    }
    return gefundeneVerbindungen
}

func aStarPfad(von start: Punkt, zum ziel: Punkt) -> ([Punkt], Int)? {

    var zuPruefendeKnoten = [start]

    var kommtVon = [Punkt: Punkt]()

    var weglaengeZuPunkten = [start: 0]

    var zuErwartendeDauerVonPunkten = [start: h(start)]

    while !zuPruefendeKnoten.isEmpty {
        guard let current = zuPruefendeKnoten.sorted(by:
{zuErwartendeDauerVonPunkten[$0]! < zuErwartendeDauerVonPunkten[$1]!}).first
else { return nil }
        if current == ziel {
            return (rekonstruiereWeg(kommtVon, current),
weglaengeZuPunkten[current]!)
        }

        guard let index = zuPruefendeKnoten.firstIndex(of: current) else
{ continue }
        zuPruefendeKnoten.remove(at: index)

        for bindung in sucheNachVerbindungen(fuerPunkt: current) {
            let weglaengeZuPunkt = weglaengeZuPunkten[current]! + bindung.1
            if weglaengeZuPunkt < weglaengeZuPunkten[bindung.0] ??
9999999999999999 {
                kommtVon[bindung.0] = current
                weglaengeZuPunkten[bindung.0] = weglaengeZuPunkt
                zuErwartendeDauerVonPunkten[bindung.0] = weglaengeZuPunkt +
h(bindung.0)
                if !zuPruefendeKnoten.contains(bindung.0) {
                    zuPruefendeKnoten.append(bindung.0)
                }
            }
```

```
        }
    }

    return nil
}

guard let weg = aStarPfad(von: start, zum: ziel) else {
    print("Es wurde kein Weg von A nach B gefunden.")
    exit(EXIT_FAILURE)
}

print("\nBerechnung des Weges erfolgreich!")
print("Der Kürzeste Weg geht über folgende Punkte:")
for i in weg.0 {
    print("Punkt(\(i.x1 + 1)|\(i.x2 + 1)) auf dem \(i.x3 == 0 ? "ersten" :
"zweiten") Stockwerk")
}
print("Der Weg ist \(weg.1) Sekunden lang.")
```