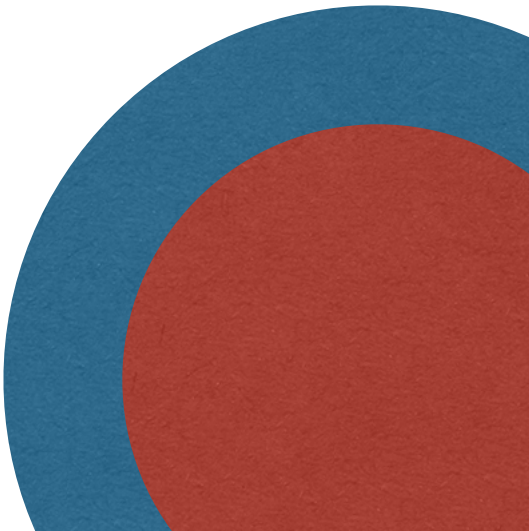
42. Bundeswettbewerb Informatik

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



Aufgabe 5: Stadtführung

Inhaltsverzeichnis

Lösungsidee

Um, wie in die Aufgabenstellung gefordert, die kürzeste Route zu finden, haben wir uns überlegt, durch die verschiedenen Tourpunkte zu iterieren und die nicht nötigen „Schleifen" herauszufiltern. Um solche „Schleifen" zu erkennen, soll der Algorithmus ausgehend von einem Startpunkt in der restlichen Tour nach einem Punkt suchen, welcher den gleichen Ort hat. Falls zwischen dem Startpunkt und diesem gefundenen Punkt kein essenzieller Punkt liegt, so können die Punkte dazwischen (und in manchen Fällen auch einer von diesen beiden Punkten) gelöscht werden. Dieser Vorgang soll nun mit allen Tourpunkten als Startpunkt einmal ausgeführt werden.
Dabei mussten wir allerdings noch verschiedene Umstände berücksichtigen:

1. Nicht nötige „Schleifen" können auch über das Ende der Tour hinausgehen. Diese müssen auch vom Algorithmus erkannt werden. Ein Beispiel hierfür stellt die Datei tour4.txt dar. Hier muss die Schleife von dem Tourpunkt „Marktplatz, 1962" bis zum Punkt „Marktplatz, 1562" weggelöscht werden, um die schnellste Route zu erhalten.
2. Außerdem muss beachtet werden, dass nicht jede Kürzung produktiv ist. Wenn jede gefundene, nicht nötige „Schleife" direkt gelöscht werden würde, würde dies bedeuten, dass andere, möglicherweise effektivere Wege nicht mehr entdeckt werden können. Dies wollen wir lösen, in dem jedes Mal, wenn eine unnötige „Schleife" gelöscht wird, eine Kopie von der bisherigen Tour erstellt wird, welche als eigene Variation betrachtet wird. Insgesamt sollen also alle möglichen Kombinationen von Tour-Kürzungen gesichert werden, damit die effektivste Kombination ausgewählt werden kann.
3. Um zu ermöglichen, dass alle Kombinationen von Tour-Kürzungen entdeckt werden, haben wir uns zudem überlegt, die Iteration durch die Tourpunkte nicht beim chronologisch ersten Tourpunkt, sondern beim chronologisch ersten essenziellen Tourpunkt zu beginnen. Selbstverständlich werden trotzdem noch alle Tourpunkte eingebunden, nur der Start- und Endpunkt der Schleife, welche durch alle Tourpunkte iteriert, wird verschoben.

Umsetzung

Der Code liest die Tour aus einer externen Datei ein.
Er transformiert den Inhalt der Datei zu einem Array von Tourpunkt(en) namens **urspruenglicheTour**.
Ein Tourpunkt hat folgende Attribute:

- jahr: Integer, gibt das Jahr an
- ort: String, gibt den Ort an
- essenziell: Boolean, ist wahr, wenn der Tourpunkt essenziell ist

- abstandVomStart: Integer, Distanz vom Start der Tour (in Metern)

Es wird sichergestellt, dass die Tour mindestens drei Tourpunkte hat, da sonst nichts gekürzt werden kann. Die ursprüngliche Länge der Tour wird gesichert, in dem der Abstand zum Start des letzten Punktes ausgelesen wird. Nun wird der chronologisch erste essenzielle Tourpunkt gefunden. Falls keiner vorhanden ist, so kann die Route auf den Start- und den Endpunkt reduziert werden und der Algorithmus ist fertig, denn dann ist ja die ganze Route eine redundante Schleife.
Die Variable **neueTouren** speichert alle Tour-Variationen in Kombination mit der Länge dieser. Initial beinhaltet sie nur die **ursprünglicheTour** zusammen mit der **urspruenglichenLaenge**.
Nun wird durch die ursprüngliche Route iteriert, um mögliche unnötige Schleifen aufzuspüren. Betrachtet wird immer der Tourpunkt **untersuchtesObjekt**, mit dem Index **untersuchterIndex** in der Liste **urspruenglicheTour**, welcher initial auf den Index des ersten essenziellen Punktes gesetzt wird. Am Anfang jedes Durchlaufes der Schleife wird geprüft, ob der betrachtete Punkt essenziell ist. Falls dies gegeben ist, so kann die Liste der bestehenden Tour-Variationen auf ihre kürzeste Tour-Variation reduziert werden, was die Funktion **reduziereAufKuerzesteTour** übernimmt. Diese Reduzierung ist zulässig, weil alle Tour-Variationen diesen Punkt enthalten müssen, und somit an diesem Punkt vergleichbar sind. Jetzt wird für alle Tour-Variationen geprüft, ob das **untersuchtesObjekt** noch enthalten ist. Wenn dies der Fall ist, so wird in der Tour-Variation nach einer Schleife gesucht, in dem die Orte der Tourpunkte, die chronologisch nach dem **untersuchtenObjekt** kommen, mit dem Ort des **untersuchtenObjekt**es abgeglichen werden. Falls ein essenzielles Element gefunden wird, bevor eine Schleife gefunden wird, wird die Suche nach Schleifen für diese Tour-Variation abgebrochen. Falls aber eine Schleife gefunden wird, so wird diese entfernt, mithilfe der **loescheTourpunkteAusTour**-Funktion. Diese gibt die Tour-Variation ohne die gelöschten Tourpunkte wieder zurück, wobei auch bei allen Tourpunkten das **abstandVomStart**-Attribut aktualisiert wurde, zusammen mit der neuen Gesamtlänge der Tour-Variation.

Abschließend wird die Funktion **reduziereAufKuerzesteTour** (nochmals) aufgerufen, um den kürzesten Weg auszuwählen. Der kürzeste Weg und einige Informationen über den Erfolg des Algorithmus werden 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/tour4.txt" ausgelesen.

Beispiele

Datei	Ursprüngliche Tour	Neue Tour	Eingesparter Weg
tour1.txt	Brauerei - 1613 - Essenziell - 0m Karzer - 1665 - Essenziell - 80m Rathaus - 1678 - Essenziell - 150m Gründungsstein - 1685 - Nicht essenziell - 310m Wallanlage - 1690 - Nicht essenziell - 410m Rathaus - 1739 - Essenziell - 500m Euler-Brücke - 1768 - Nicht essenziell - 680m Fibonacci-Gaststätte - 1820 - Essenziell - 710m Schiefes Haus - 1823 - Nicht essenziell - 830m Theater - 1880 - Nicht essenziell - 960m Emmy-Noether-Campus - 1912 - Essenziell - 1090m Fibonacci-Gaststätte - 1923 - Nicht essenziell - 1180m Hilbert-Raum - 1945 - Nicht essenziell - 1300m Schiefes Haus - 1950 - Nicht essenziell - 1400m Gauß-Turm - 1952 - Nicht essenziell - 1450m Emmy-Noether-Campus - 1998 - Essenziell - 1780m Euler-Brücke - 1999 - Nicht essenziell - 1910m Brauerei - 2012 - Nicht essenziell - 2060m	Brauerei - 1613 - Essenziell - 0m Karzer - 1665 - Essenziell - 80m Rathaus - 1678 - Essenziell - 150m Rathaus - 1739 - Essenziell - 150m Euler-Brücke - 1768 - Nicht essenziell - 330m Fibonacci-Gaststätte - 1820 - Essenziell - 360m Schiefes Haus - 1823 - Nicht essenziell - 480m Theater - 1880 - Nicht essenziell - 610m Emmy-Noether-Campus - 1912 - Essenziell - 740m Emmy-Noether-Campus - 1998 - Essenziell - 740m Euler-Brücke - 1999 - Nicht essenziell - 870m Brauerei - 2012 - Nicht essenziell - 1020m	1040m
tour2.txt	Brauerei - 1613 - Nicht essenziell - 0m Karzer - 1665 - Essenziell - 80m Rathaus - 1678 - Nicht essenziell - 150m Gründungsstein - 1685 - Nicht essenziell - 310m Wallanlage - 1690 - Nicht essenziell - 410m Rathaus - 1739 - Nicht essenziell - 500m Euler-Brücke - 1768 - Nicht essenziell - 680m Fibonacci-Gaststätte - 1820 - Essenziell - 710m Schiefes Haus - 1823 - Nicht essenziell - 830m Theater - 1880 - Nicht essenziell - 960m Emmy-Noether-Campus - 1912 - Essenziell - 1090m Fibonacci-Gaststätte - 1923 - Nicht essenziell - 1180m Hilbert-Raum - 1945 - Nicht essenziell - 1300m Schiefes Haus - 1950 - Nicht essenziell - 1400m Gauß-Turm - 1952 - Nicht essenziell - 1450m Emmy-Noether-Campus - 1998 - Essenziell - 1780m Euler-Brücke - 1999 - Nicht essenziell - 1910m Brauerei - 2012 - Nicht essenziell - 2060m	Brauerei - 1613 - Nicht essenziell - 0m Karzer - 1665 - Essenziell - 80m Rathaus - 1678 - Nicht essenziell - 150m Euler-Brücke - 1768 - Nicht essenziell - 330m Fibonacci-Gaststätte - 1820 - Essenziell - 360m Schiefes Haus - 1823 - Nicht essenziell - 480m Theater - 1880 - Nicht essenziell - 610m Emmy-Noether-Campus - 1912 - Essenziell - 740m Emmy-Noether-Campus - 1998 - Essenziell - 740m Euler-Brücke - 1999 - Nicht essenziell - 870m Brauerei - 2012 - Nicht essenziell - 1020m	1040m
tour3.txt	Talstation - 1768 - Nicht essenziell - 0m Wäldle - 1805 - Nicht essenziell - 520m Mittlere Alp - 1823 - Nicht essenziell - 1160m Observatorium - 1833 - Nicht essenziell - 1450m Wäldle - 1841 - Nicht essenziell - 1700m Bergstation - 1866 - Nicht essenziell - 2370m Observatorium - 1874 - Essenziell - 2740m Piz Spitz - 1898 - Nicht essenziell - 3210m Panoramasteg - 1912 - Essenziell - 3430m Bergstation - 1928 - Nicht essenziell - 3690m Ziegenbrücke - 1935 - Nicht essenziell - 3870m Panoramasteg - 1952 - Nicht essenziell - 4030m Ziegenbrücke - 1979 - Essenziell - 4280m Talstation - 2005 - Nicht essenziell - 4560m	Talstation - 1768 - Nicht essenziell - 0m Wäldle - 1805 - Nicht essenziell - 520m Mittlere Alp - 1823 - Nicht essenziell - 1160m Observatorium - 1874 - Essenziell - 1450m Piz Spitz - 1898 - Nicht essenziell - 1920m Panoramasteg - 1912 - Essenziell - 2140m Ziegenbrücke - 1979 - Essenziell - 2390m Talstation - 2005 - Nicht essenziell - 2670m	1890m

Datei	Ursprüngliche Tour	Neue Tour	Eingesparter Weg
tour4.txt	Blaues Pferd - 1523 - Nicht essenziell - 0m Alte Mühle - 1544 - Nicht essenziell - 110m Marktplatz - 1549 - Nicht essenziell - 210m Zeughaus - 1559 - Nicht essenziell - 310m Marktplatz - 1562 - Nicht essenziell - 410m Springbrunnen - 1571 - Nicht essenziell - 490m Dom - 1596 - Essenziell - 560m Bogenschütze - 1610 - Nicht essenziell - 680m Marktplatz - 1622 - Nicht essenziell - 820m Ruhiges Eck - 1625 - Nicht essenziell - 850m Frauentor - 1636 - Nicht essenziell - 910m Ruhiges Eck - 1651 - Nicht essenziell - 980m Springbrunnen - 1675 - Nicht essenziell - 1000m Bogenschütze - 1683 - Nicht essenziell - 1070m Schnecke - 1698 - Essenziell - 1220m Fischweiher - 1710 - Nicht essenziell - 1400m Reiterhof - 1728 - Essenziell - 1520m Schnecke - 1742 - Nicht essenziell - 1660m Schmiede - 1765 - Nicht essenziell - 1830m Große Gabel - 1794 - Nicht essenziell - 1940m Schnecke - 1829 - Nicht essenziell - 2050m Europapark - 1852 - Nicht essenziell - 2280m Große Gabel - 1874 - Nicht essenziell - 2550m Fingerhut - 1917 - Essenziell - 2620m Stadion - 1934 - Nicht essenziell - 2740m Marktplatz - 1962 - Nicht essenziell - 2830m Baumschule - 1974 - Nicht essenziell - 2910m Polizeipräsidium - 1991 - Nicht essenziell - 3050m Blaues Pferd - 2004 - Nicht essenziell - 3200m	Marktplatz - 1562 - Nicht essenziell - 0m Springbrunnen - 1571 - Nicht essenziell - 80m Dom - 1596 - Essenziell - 150m Bogenschütze - 1610 - Nicht essenziell - 270m Schnecke - 1698 - Essenziell - 420m Fischweiher - 1710 - Nicht essenziell - 600m Reiterhof - 1728 - Essenziell - 720m Schnecke - 1742 - Nicht essenziell - 860m Schmiede - 1765 - Nicht essenziell - 1030m Große Gabel - 1794 - Nicht essenziell - 1140m Fingerhut - 1917 - Essenziell - 1210m Stadion - 1934 - Nicht essenziell - 1330m Marktplatz - 1962 - Nicht essenziell - 1420m	1780m
tour5.txt	Gabelhaus - 1638 - Nicht essenziell - 0m Burgruine - 1654 - Nicht essenziell - 140m Labyrinth - 1667 - Nicht essenziell - 220m Hängepartie - 1672 - Nicht essenziell - 470m Hexentanzplatz - 1681 - Nicht essenziell - 730m Gabelhaus - 1699 - Nicht essenziell - 820m Hexentanzplatz - 1703 - Essenziell - 980m Eselsbrücke - 1711 - Nicht essenziell - 1100m Dreibannstein - 1724 - Nicht essenziell - 1210m Alte Wache - 1733 - Nicht essenziell - 1340m Palisadenhaus - 1740 - Nicht essenziell - 1490m Dreibannstein - 1752 - Nicht essenziell - 1620m Schmetterling - 1760 - Essenziell - 1770m Dreibannstein - 1781 - Nicht essenziell - 1850m Märchenwald - 1793 - Essenziell - 1930m Fuchsbau - 1811 - Nicht essenziell - 2010m Torfmoor - 1817 - Nicht essenziell - 2120m Gartenschau - 1825 - Nicht essenziell - 2260m Märchenwald - 1840 - Nicht essenziell - 2340m Eselsbrücke - 1855 - Nicht essenziell - 2420m Heimatmuseum - 1863 - Nicht essenziell - 2490m Eselsbrücke - 1877 - Nicht essenziell - 2590m Reiterdenkmal - 1880 - Nicht essenziell - 2730m Riesenrad - 1881 - Nicht essenziell - 2910m Hochsitz - 1885 - Nicht essenziell - 2980m Gartenschau - 1898 - Nicht essenziell - 3060m Riesenrad - 1902 - Nicht essenziell - 3180m Dreibannstein - 1911 - Essenziell - 3310m Olympisches Dorf - 1924 - Nicht essenziell - 3470m Haus der Zukunft - 1927 - Essenziell - 3600m Stellwerk - 1931 - Nicht essenziell - 3720m Dreibannstein - 1938 - Nicht essenziell - 3890m Stellwerk - 1942 - Nicht essenziell - 4020m Labyrinth - 1955 - Nicht essenziell - 4230m Gauklerstadl - 1961 - Nicht essenziell - 4310m Planetarium - 1971 - Essenziell - 4390m Känguruhfarm - 1976 - Nicht essenziell - 4440m Balzplatz - 1978 - Nicht essenziell - 4520m Dreibannstein - 1998 - Essenziell - 4610m Labyrinth - 2013 - Nicht essenziell - 4740m CO2-Speicher - 2022 - Nicht essenziell - 4930m Gabelhaus - 2023 - Nicht essenziell - 5000m	Gabelhaus - 1699 - Nicht essenziell - 0m Hexentanzplatz - 1703 - Essenziell - 160m Eselsbrücke - 1711 - Nicht essenziell - 280m Dreibannstein - 1724 - Nicht essenziell - 390m Schmetterling - 1760 - Essenziell - 540m Dreibannstein - 1781 - Nicht essenziell - 620m Märchenwald - 1793 - Essenziell - 700m Eselsbrücke - 1855 - Nicht essenziell - 780m Reiterdenkmal - 1880 - Nicht essenziell - 920m Riesenrad - 1881 - Nicht essenziell - 1100m Dreibannstein - 1911 - Essenziell - 1230m Olympisches Dorf - 1924 - Nicht essenziell - 1390m Haus der Zukunft - 1927 - Essenziell - 1520m Stellwerk - 1931 - Nicht essenziell - 1640m Labyrinth - 1955 - Nicht essenziell - 1850m Gauklerstadl - 1961 - Nicht essenziell - 1930m Planetarium - 1971 - Essenziell - 2010m Känguruhfarm - 1976 - Nicht essenziell - 2060m Balzplatz - 1978 - Nicht essenziell - 2140m Dreibannstein - 1998 - Essenziell - 2230m Labyrinth - 2013 - Nicht essenziell - 2360m CO2-Speicher - 2022 - Nicht essenziell - 2550m Gabelhaus - 2023 - Nicht essenziell - 2620m	2380m

Im Aufgabentext wird erzählt, dass der Tourbegleiter die Geschichte der weggekürzten Tourpunkte trotzdem erzählen will, es nur nicht für nötig hält dazu bei diesen zu sein. Bei der von uns neu errechneten Tour handelt es sich also nicht mehr um eine vollständige Auflistung der erzählten Geschichten, sondern nur der besuchten Orte.

Quellcode

```
import Foundation

struct Tourpunkt: Equatable {
    let jahr: Int
    let ort: String
    let essenziell: Bool
    var abstandVomStart: Int

    static func == (_ lhs: Tourpunkt, _ rhs: Tourpunkt) -> Bool {
        lhs.jahr == rhs.jahr && lhs.ort == rhs.ort
    }
}

var urspruenglicheTour: [Tourpunkt] = []

// Tour aus der externen Datei einlesen
let dateiNummer: String = CommandLine.arguments.last ?? "-"
let dateiNummerInt: Int = Int(dateiNummer) ?? 1
let pfad: URL = URL(fileURLWithPath: "../Daten/A5_Stadtfuehrung/tour\(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)
let ersteZeile = zeilen.removeFirst()

for zeile in zeilen {
    let parameter = zeile.split(separator: ",")
    guard parameter.count == 4, let jahr = Int(parameter[1].replacing(" ", with: "")), let abstand =
Int(parameter[3].replacing(" ", with: "")) else {
        print("Die Zeile \"\(zeile)\" hat nicht das erwartete Format.")
        exit(EXIT_FAILURE)
    }
    let neuerTourpunkt = Tourpunkt(jahr: jahr, ort: String(parameter[0]), essenziell: parameter[2]
== "X", abstandVomStart: abstand)
    urspruenglicheTour.append(neuerTourpunkt)
}

guard urspruenglicheTour.count > 2 else {
    print("Die Route muss mindestens 3 Tourpunkte enthalten, um möglicherweise optimiert werden zu
können.")
    exit(EXIT_FAILURE)
}

// Force-Unwrap wird immer gelingen, da in Z.38 gesichert wurde, dass die Liste mindestens 3
Elemente hat.
let urspruenglicheLaenge = urspruenglicheTour.last!.abstandVomStart

// Eine Liste möglicher neue Touren, mit ihrer jeweiligen Länge
// TODO: Könnte zu einem Dictinary (Map) umgeformt werden
var neueTouren: [([Tourpunkt], Int)] = [(urspruenglicheTour, urspruenglicheLaenge)]

// Diese Funktion sucht von allen Routen-Varianten die aktuell kürzeste heraus,
// und löscht alle anderen.
func reduziereAufKuerzesteTour(){
    var kuerzesteTour = (urspruenglicheTour, urspruenglicheLaenge)
    for i in neueTouren {
        if i.1 < kuerzesteTour.1 {
            kuerzesteTour = i
        }
    }
    neueTouren = [kuerzesteTour]
}

func loescheTourpunkteAusTour(von startIndex: Int, bis endIndex: Int, bei tour: ([Tourpunkt], Int))
-> ([Tourpunkt], Int) {
    var veraenderbareTour = tour // Veränderbare Kopie der Tour
    var schonGeloeschte = 0
    var entfernteStrecke = 0
    if endIndex >= startIndex {
        for i in startIndex...endIndex {
            // Tourpunkt löschen
            veraenderbareTour.0.remove(at: i - schonGeloeschte)
            schonGeloeschte += 1
            // Länge anpassen
            if i != 0 {
                // Zu dem betrachteten Punkt hinführende Strecke löschen
```

```
                // Zum Anfangspunkt der Tour fürt kein Weg hin, deshalb kann auch keiner gelöscht
werden
                let streckeZuDiesemPunkt = tour.0[i].abstandVomStart - tour.0[i - 1].abstandVomStart
                entfernteStrecke += streckeZuDiesemPunkt
            }
            if i == endIndex && i != tour.0.count - 1 {
                // Von dem betrachteten Punkt wegführende Strecke löschen
                let streckeAbDiesemPunkt = tour.0[i + 1].abstandVomStart - tour.0[i].abstandVomStart
                entfernteStrecke += streckeAbDiesemPunkt
            }
        }
    } else {
        // Zu löschende Tourpunkte gehen über das Ende der Tour hinaus
        // Lösche zunächst die Tourpunkte vom _startIndex_ bis zum Ende der Liste
        for i in startIndex...(tour.0.count - 1) {
            veraenderbareTour.0.remove(at: i - schonGeloeschte)
            schonGeloeschte += 1
            // Verändern der Länge
            if i != 0 {
                // Zu dem betrachteten Punkt hinführende Strecke löschen
                // Zum Anfangspunkt der Tour fürt kein Weg hin, deshalb kann auch keiner gelöscht
werden
                let streckeZuDiesemPunkt = tour.0[i].abstandVomStart - tour.0[i - 1].abstandVomStart
                entfernteStrecke += streckeZuDiesemPunkt
            }
            if i == endIndex && i != tour.0.count - 1 {
                // Von dem betrachteten Punkt wegführende Strecke löschen
                let streckeAbDiesemPunkt = tour.0[i + 1].abstandVomStart - tour.0[i].abstandVomStart
                entfernteStrecke += streckeAbDiesemPunkt
            }
        }
        // _schonGeloeschte_ muss zurückgesetzt werden, da sonst das remove Statement in der
        // folgenden Schleife nicht die richtigen Tourpunkte löscht
        schonGeloeschte = 0
        // Verrechne die entfernte Strecke mit der ingesamten Streckenlänge und setzte sie
anschließend zurück.
        // Begründung: Später wird die Entfernung zum Anfang der Tour für alle Punkte neu berechnet.
Dabei darf nicht
        // ins Gewicht fallen, wie viel Strecke grade am Ende der Liste gelöscht wurde, sondern nur
wieviel gleich
        // vom Anfang der Liste weggenommen wird.
        veraenderbareTour.1 -= entfernteStrecke
        entfernteStrecke = 0
        for i in 0...endIndex {
            veraenderbareTour.0.remove(at: i - schonGeloeschte)
            schonGeloeschte += 1
            // Verändern der Länge
            if i != 0 {
                // Zu dem betrachteten Punkt hinführende Strecke löschen
                // Zum Anfangspunkt der Tour fürt kein Weg hin, deshalb kann auch keiner gelöscht
werden
                let streckeZuDiesemPunkt = tour.0[i].abstandVomStart - tour.0[i - 1].abstandVomStart
                entfernteStrecke += streckeZuDiesemPunkt
            }
            if i == endIndex && i != tour.0.count - 1 {
                // Von dem betrachteten Punkt wegführende Strecke löschen
                let streckeAbDiesemPunkt = tour.0[i + 1].abstandVomStart - tour.0[i].abstandVomStart
                entfernteStrecke += streckeAbDiesemPunkt
            }
        }
    }
    veraenderbareTour.1 -= entfernteStrecke
    if endIndex != veraenderbareTour.0.count - 1 {
        for tourpunkt in veraenderbareTour.0[(endIndex - schonGeloeschte) + 1 ..<
veraenderbareTour.0.count].enumerated() {
            veraenderbareTour.0[tourpunkt.offset + (endIndex - schonGeloeschte) + 1].abstandVomStart
-= entfernteStrecke
        }
    }
    return veraenderbareTour
}

// Ersten essenziellen Punkt in der Route finden
if let indexVomErstenEssenziellenPunkt = urspruenglicheTour.firstIndex(where: {$0.essenziell}) {

    // Die Suche startet beim ersten essenziellen Punkt, damit keine Schleifen übersehen werden
können.
    // TODO: Diese Entscheidung in der Dokumentation ausführlicher erklären
    var untersuchterIndex = indexVomErstenEssenziellenPunkt

    for _ in 0..<urspruenglicheTour.count {
```

```
        let untersuchtesObjekt = urspruenglicheTour[untersuchterIndex]
        if untersuchtesObjekt.essenziell {
            // Bei essenziellen Punkten kann der bisher schnellste Weg als generell schnellster Weg
deklariert werden
            reduziereAufKuerzesteTour()
        }
        // Alle verschiedenen neuen Touren absuchen
        for neueTour in neueTouren {
            // Index des untersuchten Tourpunktes in der betrachteten neuen Tour.
            // Muss nicht mit _untersuchterIndex_ übereinstimmen, da bei der neuenTour ja Elemente
fehlen können.
            // Wenn der Tourpunkt nicht in der neuen Tour ist, wird zur nächsten neuen Tour
geskippt.
            guard let StartIndexInNeuerTour = neueTour.0.firstIndex(of: untersuchtesObjekt) else
{continue }
            // Durchlaufender Index, mit welchem nun nach Schleifen, die von _StartIndexInNeuerTour_
ausgehen, gesucht wird.
            var untersuchterIndexInNeuerTour = (StartIndexInNeuerTour + 1) % neueTour.0.count
            // Breche den Loop ab, wenn er die ganze Liste einmal durchgesucht hat.
            while StartIndexInNeuerTour != untersuchterIndexInNeuerTour {
                let untersuchtesObjektNeueTour = neueTour.0[untersuchterIndexInNeuerTour]
                if untersuchtesObjektNeueTour.ort == untersuchtesObjekt.ort {
                    // Die zwei betrachteten Tourpunkte sind am gleichen Ort
                    let keinOrtDazwischen = (StartIndexInNeuerTour + 1) % neueTour.0.count ==
untersuchterIndexInNeuerTour
                    if !keinOrtDazwischen {
                        print((StartIndexInNeuerTour + 1))
                        print(untersuchterIndexInNeuerTour - 1)
                        let neueneueTour = loescheTourpunkteAusTour(von: (StartIndexInNeuerTour + 1)
% neueTour.0.count, bis: (untersuchterIndexInNeuerTour - 1) % neueTour.0.count, bei: neueTour)
                        neueTouren.append(neueneueTour)
                    }
                }
                if untersuchtesObjektNeueTour.essenziell {
                    // Breche den Suchvorgang ab wenn du auf einen essenziellen Punkt gestoßen bist.
                    // Loop wird allerdings erst hier unten (nach dem Suchvorgang) abgebrochen,
damit Schleifen zwischen
                    // essenziellen Punkten gefunden werden können.
                    break
                } else {
                    // Erhöhe den Index, und fang von vorne an wenn du am Ende der Liste bist
                    untersuchterIndexInNeuerTour = (untersuchterIndexInNeuerTour + 1) %
neueTour.0.count
                }
            }
        }
        // Erhöhe den untersuchten Index. Wenn der Index schon das Ende der Liste war, fang wieder
von vorne an.
        untersuchterIndex = (untersuchterIndex + 1) % urspruenglicheTour.count
    }
    reduziereAufKuerzesteTour()
} else {
    // Die Tour enthält keinen einzigen essenziellen Punkt, kann also auf Start- und Endpunkt
reduziert werden.
    neueTouren = [([urspruenglicheTour[0], urspruenglicheTour.last!], 0)]
}

// Ausgabe
print("------\nUrsprüngliche Tour:")
for tour in urspruenglicheTour {
    print(tour.ort + " - " + tour.jahr.description + " - " + (tour.essenziell ? "Essenziell - " :
"Nicht essenziell - ") + tour.abstandVomStart.description + "m" )
}

print("------\nNeue Tour:")
for tour in neueTouren[0].0 {
    print(tour.ort + " - " + tour.jahr.description + " - " + (tour.essenziell ? "Essenziell - " :
"Nicht essenziell - ") + tour.abstandVomStart.description + "m" )
}

print("------\nAnalyse:")
print("- Die Tour hat nach der Kürzung \(urspruenglicheTour.count - neueTouren[0].0.count) weniger
Tourpunkte.")
print("Ursprünglich waren es \(urspruenglicheTour.count), jetzt sind es \(neueTouren[0].0.count).")
print("- Die Tour ist \(urspruenglicheLaenge - neueTouren[0].1)m kürzer geworden.")
print("Ursprünglich waren es \(urspruenglicheLaenge)m, jetzt sind es \(neueTouren[0].1)m.")
```