Aufgabe 5: Stadtführung

Team-ID: 00265

Teilnahme-ID: 69306

Team-Name: Die Debugger

Bearbeiter dieser Aufgabe:
Abulfasl Ahmadi

19. November 2023

Inhaltsverzeichnis

1 Lösungsidee und Umsetzung

Vorweg sei gesagt, dass die Umsetzung in Java Script ES6 erfolgte. Die Lösungsidee ist zusätzich in Pseudocode beschrieben. Der Quellcode ist im Anhang zu finden.

Zunächst werden die Stops und die weiteren Informationen in einem Array gespeichert.

Im ersten Schritt werden die Orte vor und inkl. dem aller ersten essenziellen Stop („beforeEssential“), sowie die Orte nach und inkl. dem aller letzten essenziellen Stop („after essential“) in zwei Arrays gespeichert. Im array „beforeEssential“ werden die stops von hinten nach vorne mit dem „afterEssential“ Array abgeglichen, ob es dupplikate gibt. Es wird explizit von hinten nach vorne abgeglichen, da man die größt mögliche Teilroute entfernen möchte. Wenn ein Duplikat gefunden wird, wird die Teilroute zwischen den beiden Duplikaten entfernt.

Im zweiten schritt werden valide duplikate gefunden d.h. dass zwischen den gefunden duplikaten keine essenziellen Stops liegen. Diese werden dann in einem Array gespeichert.

Im dritten schritt wird zuerst die größte Teilroute aus dem Array der validen Duplikate entfernt. Dann wird der zweite schrtt wiederholt, da durch das entfernen der teilroute, andere duplikate entfernt wurden, als auch die indexe sich „verschoben“ haben. Dies wird solange wiederholt, bis keine validen Duplikate mehr gefunden werden.

2 Psuedocode:

starke vereinfachung des codes, da nur die lösungsidee beschrieben werden soll.

Schritt 1:

beforeEssential = [ ]

afterEssential = [ ]

for (len of beforeEssential; len >= 0 len- -) {

abgelich vom index, ob duplikat mit afterEssential

if (duplikat) {

entferne teilroute zwischen den beiden duplikaten

} }

Schritt 2:

let duplicate = [ ]

if (duplikat && keine essenziellen stops zwischen den duplikaten) {

duplicate.push(duplikat)

}

schritt 3:

let max = 0

for (each duplicate) {

if (duplicate differenz > max) {

max = duplicate differenz

}

}

größte teilroute entfernen mit js methoden (array.splice())

3 Beispiele

3.1 Beispiel 1

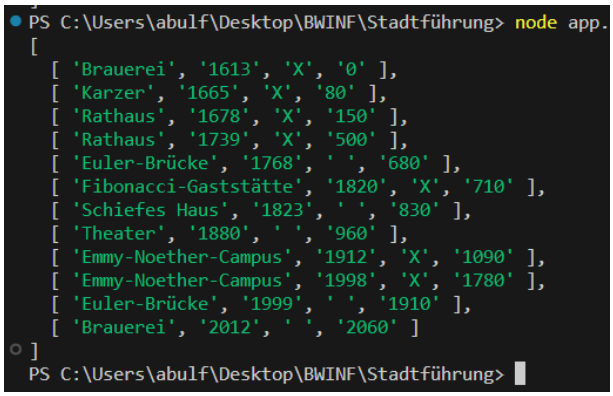

Abbildung 1: gekürzte Tour 1 aus der BwInf Seite

3.2 Beispiel 2

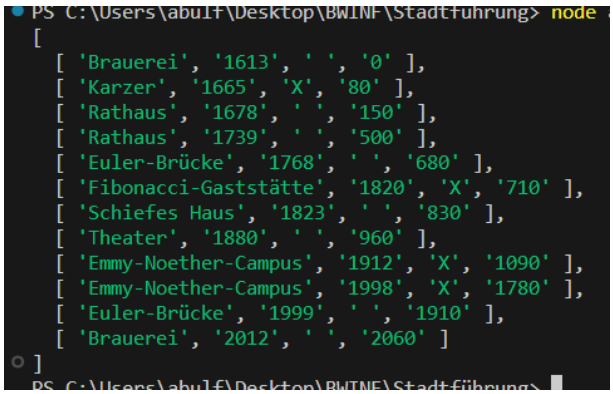

Abbildung 2: gekürzte Tour 2 aus der BwInf Seite

3.3 Beispiel 3

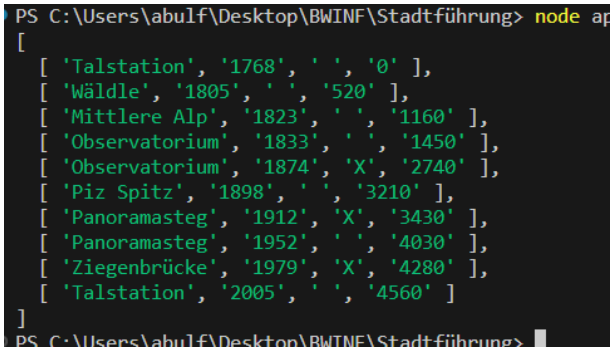

Abbildung 3: gekürzte Tour 3 aus der BwInf Seite

3.4 Beispiel 4

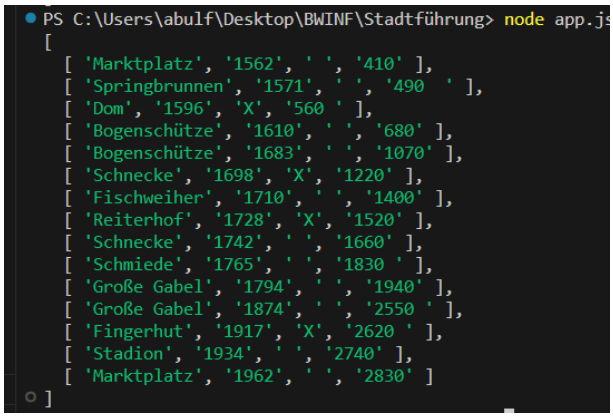

Abbildung 4: gekürzte Tour 4 aus der BwInf Seite

3.5 Beispiel 5

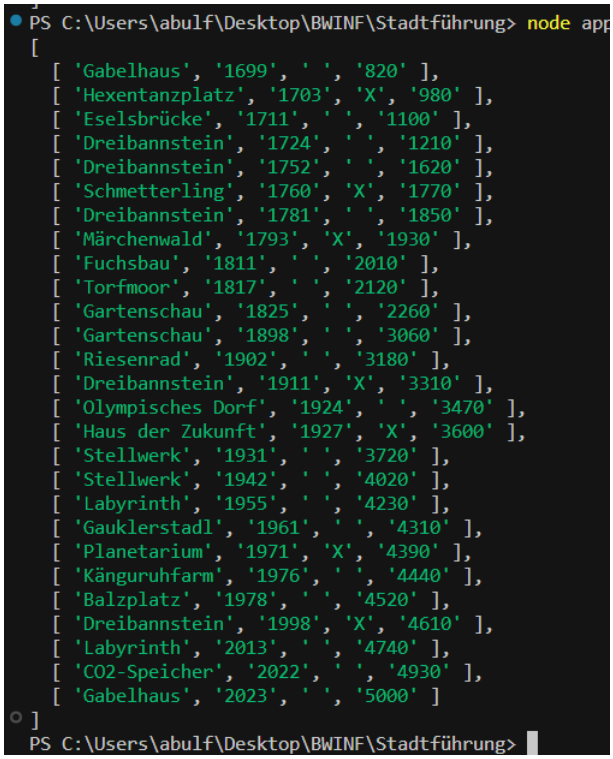

Abbildung 5: gekürzte Tour 5 aus der BwInf Seite

4 Quellcode

```
// laden des Filesystems und der Daten
const fs = require('fs');
const data = fs.readFileSync('./tests/tour5.txt', {encoding: "utf-8"});

// Hier wird die neue Liste gespeichert
let result = []

// Die Daten werden in ein Array gespeichert um die besser zugänglich/bearbeitbar
zu machen
let inputs = []
inputs = data.replace(/\r/g, "").split('\n').map(x => x.split(','));
result = inputs.slice(1, inputs.length)
```

```
// Gucken, ob man Start und Ziel verändern kann

function startAndEnd() {
    //Spiechern der essenziellen Stops in einem Array
    let listOfIndexOfEssential = []
    for(each in result) {
        if(result[each][2] === "X") {
            listOfIndexOfEssential.push(parseInt(each))
        }
    }

    // Orte VOR dem essentiellen Stop werden in beforeEssential gespeichert
    let beforeEssential = []
    for(let i=0; i<listOfIndexOfEssential[0]+1; i++) {
        beforeEssential.push(result[i][0])
    }

    // Orte NACH dem essentiellen Stop werden in afterEssential gespeichert
    let afterEssential = []
                                            for(let           i=result.length-1;
i>(listOfIndexOfEssential[listOfIndexOfEssential.length-1]-1); i--) {
        afterEssential.push(result[i][0])
    }

   // Gucken, ob ein gleicher Ort vor dem ersten essentiellen Stop und nach dem
letzten essentiellen Stop vorkommen
    for(let i=beforeEssential.length-1; i>=0; i--) {
        if(afterEssential.indexOf(beforeEssential[i]) !== -1) {

          // die Orte werden in den Array "list" gespeichert um eine js methode
zu benutzen (indexOf)
            let list = []
            for(each in result) {
                list.push(result[each][0])
            }

          // die tour wird angepasst an die kürzung (falls überhaupt möglich)
                        result  =  result.slice(list.indexOf(result[i][0],  i),
list.indexOf(result[i][0], result.length-afterEssential.length)+1)
            break
        }
    }
}
startAndEnd();


// Funktion um duplikate zu finden
function findValidDuplicates() {

    // speichern der validen duplicate
    let duplicate = []

    // Speichern aller Orte unter dem "list" Array
    let list = [];
    for(each of result) {
        list.push(each[0]);
    }
```

```
    // zählen, wie oft ein Ort vorkommt
    duplicate = list.map(item => {
        let counter = 0;
        for(each in list) {
            if(list[each].indexOf(item) !== -1) {
                counter++
            }
        }
        return [item, counter]
    })

    // Einzeln vorkommende Orte werden gefiltert
    duplicate = duplicate.filter(item => item[1] > 1)

    // Indexe mit im "duplicate" Array speichern
    for(each in duplicate) {
        let index = []
        for(let i=0; i<result.length; i++) {
            if(result[i][0] === duplicate[each][0]) {
                index.push(i)
            }
        }
        duplicate[each].push(index)
    }

    // mehr als 2 mal vorkommende Orte werden einzeln im array gespeichert
    let counterList = {};
    for(each of duplicate) {
        if(each[1] > 2) {
            if(!(each[0] in counterList)) {
                counterList[each[0]] = 0
            }
            each[2] = each[2].slice(counterList[each[0]], counterList[each[0]]
+2)
            counterList[each[0]]++;
        }

    }
    duplicate = duplicate.filter(item => item[2].length === 2)


    // Mehrfach vorkommende Orte werden vom Array gelöscht
    const resultMap = new Map();
    for (const item of duplicate) {
        const key = JSON.stringify(item[2]);
        if (!resultMap.has(key)) {
            resultMap.set(key, item);
        }
    }
    duplicate = Array.from(resultMap.values())

    // Überprüfen, ob ein essentieller Stop zwischen den Duplikaten liegt
    for(let j=duplicate.length-1; j>=0; j--) {
        if(duplicate[j]){
            let able = true
            for(let i=duplicate[j][2][0]+1; i<duplicate[j][2][1]; i++) {
                if(result[i][2] === "X") {
                    able = false
```

```
                    break
                }
            }
            if(!able) {
                duplicate.splice(j, 1)
            }
        }
    }

    // Verhindern eines Stack overflows/Max stack size errors bei dem rekursiven
aufrufen dieser fktn von "shortTheList()"
    for(let i=duplicate.length; i>=0; i--) {
        if(duplicate[i]) {
            for(each in duplicate) {
                if((duplicate[each][2][1] - duplicate[each][2][0]) === 1) {
                    duplicate.splice(each, 1)
                    break
                }
            }
        }
    }

    return duplicate
}

// kürzen der Tour durch weglassen der geschlossenen Teiltouren.
function shortTheList() {
    let duplicates = findValidDuplicates();

    // die größte kürzung finden und diese prorisieren
    let maxIndex = 0
    let deltaMax = 0
    duplicates.map((item, i) => {
         let delta = parseInt(result[item[2][1]][3]) - parseInt(result[item[2]
[0]][3]); // abstand
        if(delta > deltaMax) {
            maxIndex = i
            deltaMax = delta
        }
    })

    // entfernen der längsten teiltour
     result.splice(duplicates[maxIndex][2][0]+1, (duplicates[maxIndex][2][1] -
duplicates[maxIndex][2][0])-1)

    // wdh der fktn falls noch duplikate übrig sind
    if(findValidDuplicates().length >0) {
        shortTheList()
    }
}
shortTheList();

// Ausgenben in der Konsole
console.log(result)
```