Aufgabe 4: Schrebergärten

Team-ID: 00108

Team-Name: Los Paper Maker‘s

Bearbeiter/-innen dieser Aufgabe:   
Stefan Herwig

18. November 2019

Inhaltsverzeichnis

[Lösungsidee 1](#_Toc530939095)

[Umsetzung 1](#_Toc530939096)

[Beispiele 2](#_Toc530939097)

[Quellcode 3](#_Toc530939098)

# Lösungsidee

In dieser Aufgabe muss eine gegebene Nummer, beispielsweise eine Telefonnummer in Blöcke von einer Länge von 2-4 Zahlen aufgeteilt werden. Dafür bedient man sich der dynamischen Programmierung, welche sehr typisch für solche Optimierungsprobleme ist.

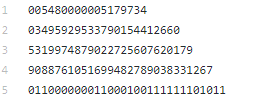
Dabei teilt man das Problem in immer kleinere Teil Probleme bis man sie lösen kann.

# Umsetzung

Ich habe mich bei meiner Lösung für eine Implementierung in Kotlin entschieden da ich sehr vertraut mit dieser Programmiersprache bin. Die Anwendung ist eine Konsolenanwendung da keine besondere optische Aufbereitung für die Lösung notwendig ist.

Mithilfe mehrerer Funktionen teilt man erst die Nummer in verschieden Blöcke auf, dabei teilt man immer den Block der mehr Nullen am Anfang hat. Dies macht man solange bis der Block bei einer Größe von 2 angekommen ist, oder ein anderer mehr nullen hat. Damit erreicht man die optimale Verteilung von den 0 auf die Blöcke entsprechend der Aufgabenstellung.

# Beispiele



# Quellcode

package dev.diekautz.bwinf38.nummernmerker

import java.io.File

fun main() {

File("src/dev/diekautz/bwinf38/nummernmerker/nummern.txt").forEachLine { //einlesen der nummern

println(splitNumber(it))

}

}

fun splitNumber(num: String): SplitSolution { //aufteilen der nummern in blöcke

if(num.length <= 4){

return SplitSolution(arrayOf(num))

} else {

val len2 = splitNumber(num.substring(2, num.lastIndex))

val len3 = splitNumber(num.substring(3, num.lastIndex))

val len4 = splitNumber(num.substring(4, num.lastIndex))

if(len2.getLeadingZeros() < len3.getLeadingZeros()){

if (len2.getLeadingZeros() < len4.getLeadingZeros()){

return len2.prepend(num.substring(0, 2))

} else {

return len4.prepend(num.substring(0, 4))

}

} else {

if(len3.getLeadingZeros() < len4.getLeadingZeros()){

return len3.prepend(num.substring(0, 3))

} else {

return len4.prepend(num.substring(0, 4))

}

}

}

}

class SplitSolution(val num: Array<String>) { //prüfen das möglichst wenig 0 am anfang stehen

fun getLeadingZeros(): Int {

var count = 0

num.forEach {

if(it.startsWith("0")){

count++

}

}

return count

}

override fun toString(): String {

return num.joinToString(" ")

}

fun prepend(string: String): SplitSolution {

return SplitSolution(arrayOf(string) + num)

}

operator fun plus(other: SplitSolution): SplitSolution {

return SplitSolution(num + other.num)

}

}