KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Programavimo kalbų teorija (P175B124)

Laboratorinių darbų ataskaita

Atliko:

IFF – 6/8 gr. studentas

Tadas Laurinaitis

2019 m. balandžio 14 d.

Priėmė:

Lekt. Evaldas Guogis

KAUNAS 2019

TURINYS

1. Python arba Ruby (L1) 3

1.1. Darbo užduotis 3

1.2. Programos tekstas 3

1.3. Pradiniai duomenys ir rezultatai 4

2. Scala (L2) 4

3. Haskell (L3) 4

4. Prolog (L4) 4

#### 1.1 Darbo užduotis

Pav #1 Darbo užduotis

#### 1.2 Programos tekstas

#Number class used to store a number and a number which shows a cumulative sum of each numbers divisors from 1 to number

class Number:

def \_\_init\_\_(self, number, cumulativeSum):

self.number = number

self.cumulativeSum = cumulativeSum

def get\_number(self):

return self.number

#finds sum of all viable divisors of number n

def findSumOfDivisors(n):

sum = 0

for x in range(2, int(n)):

z = n / x #temporary result of division

if z == int(z):

sum = sum + z

return sum

#finds cumulative sum of divisors for numbers 1 to Number.number

def findCumulativeSumOfDivisors(Number):

for x in range(0, Number.number + 1):

Number.cumulativeSum = Number.cumulativeSum + findSumOfDivisors(x)

print("Cumulative sum of divisors of number n: " + str(Number.number) + " is: " + str(Number.cumulativeSum))

return Number

#reads data from file into integer array

def readIntoArray(fileName):

array = []

with open('data.txt') as f:

for line in f: # read all lines

array.append(int(line))

return array

#finds results for all integers in array

def findResults(array):

numberArray = []

for x in array:

temp = Number(x, 0)

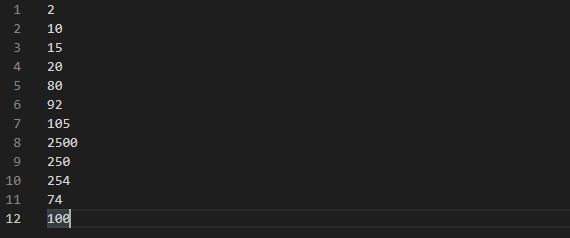
temp = findCumulativeSumOfDivisors(temp)

numberArray.append(temp)

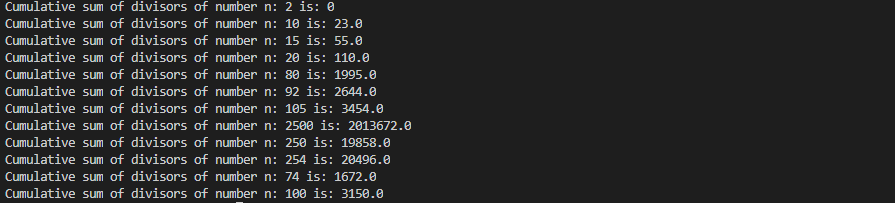
array = readIntoArray("data.txt")

findResults(array)

#### 1.3 Pradiniai duomenys ir rezultatai



Pav #2 Duomenų failas



Pav #3 Gauti rezultatai

#### Darbo užduotis

Pav #1 Darbo užduotis

#### 1.2 Programos tekstas

// Tadas Laurinaitis, IFF - 6/8, uzduoties nr. - 294, Divisors https://uva.onlinejudge.org/index.php?option=com\_onlinejudge&Itemid=8&category=4&page=show\_problem&problem=230

open System

open System.IO

let readDataFromFile file =

File.ReadAllLines(file)

let writeResultToFile file L U maxNum maxCount =

let file = File.AppendText(file)

Console.WriteLine("Between {0} and {1}, {2} has a maximum of {3} divisors", L, U, maxNum, maxCount)

file.WriteLine("Between {0} and {1}, {2} has a maximum of {3} divisors", L, U, maxNum, maxCount)

file.Close()

//let rec findDivisionsOfNumber(number : int, divisionCount : int, current : int) =

let rec findDivisionsOfNumber number divisionCount current =

if (number % current = 0 && current <= number) then

let temp1 = divisionCount + 1

let temp2 = current + 1

findDivisionsOfNumber number temp1 temp2

else if (number % current <> 0 && current <= number) then

let temp2 = current + 1

findDivisionsOfNumber number divisionCount temp2

else

let temp = divisionCount

temp

let rec findNumber L U current maxNum maxCount =

let divisionCount = findDivisionsOfNumber current 0 1

//Console.WriteLine(divisionCount)

if(divisionCount > maxCount && current <= U) then

let nextStep = current + 1

let currentMaxNum = current

let currentMaxCount = divisionCount

findNumber L U nextStep currentMaxNum divisionCount

else if (divisionCount <= maxCount && current <= U) then

let nextStep = current + 1

findNumber L U nextStep maxNum maxCount

else

let temp = maxNum

writeResultToFile "Results.txt" L U maxNum maxCount

temp

let rec doStuff current =

let numbers = readDataFromFile("D:\Tadas\KALBUTEORIJA\Fsharp\Lab3\Lab3\Data.txt")

let firstLine = numbers.[0].Split(' ')

let NN = Int32.Parse(firstLine.[0])

if(current > NN) then

printf "Job is done "

else

let strings = numbers.[current].Split(' ')

let L = Int32.Parse(strings.[0])

let U = Int32.Parse(strings.[1])

Console.WriteLine(NN)

Console.WriteLine(current)

Console.WriteLine(L)

Console.WriteLine(U)

if (U - L >= 0 && U - L <= 10000 && current <= NN) then

printf "Data looks fine "

let num = findNumber L U L 0 0

let nextStep = current + 1

doStuff nextStep

else if(U - L < 0 || U - L > 10000 && current <= NN) then

printf "The Data is incorrect "

let nextStep = current + 1

doStuff nextStep

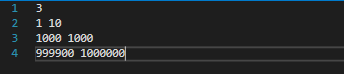
[<EntryPoint>]

let main argv =

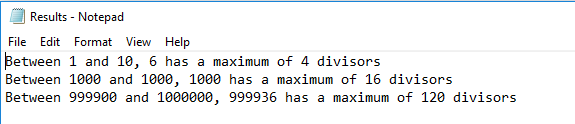
let k = doStuff 1

0

#### 1.3 Pradiniai duomenys ir rezultatai



Pav #2 Duomenų failas



Pav #3 Gauti rezultatai