**Прізвище:** Дуда

**Ім'я:** Констянтин

**Група:** КН-407

**Варіант:** 11

**Кафедра:** САПР

**Дисципліна:** Дискретні моделі в системному проектуванні

**Перевірив:** Кривий Р. З.

**Дата:**

**ЗВІТ**

до лабораторної роботи №2

на тему "Алгоритм рішення задачі листоноші"

**Мета роботи:** Метою даної лабораторної роботи є вивчення алгоритмів рішення задачі листоноші.

**Короткі теоретичні відомості:**

Будь-який листоноша перед тим, як відправитись в дорогу повинен підібрати на пошті листи, що відносяться до його дільниці, потім він повинен рознести їх адресатам, що розмістились вздовж маршрута його проходження, і повернутись на пошту. Кожен листоноша, бажаючи втратити якомога менше сил, хотів би подолати свій маршрут найкоротшим шляхом. Загалом, задача листоноші полягає в тому, щоб пройти всі вулиці маршрута і повернутися в його початкову точку, мінімізуючи при цьому довжину пройденого шляху. Перша публікація, присвячена рішенню подібної задачі, появилась в одному з китайських журналів, де вона й була названа задачею листоноші. Очевидно, що така задача стоїть не тільки перед листоношею. Наприклад, міліціонер хотів би знати найбільш ефективний шлях патрулювання вулиць свого району, ремонтна бригада зацікавлена у виборі найкоротшого шляху переміщення по всіх дорогах. Задача листоноші може бути сформульована в термінах теорії графів. Для цього побудуємо граф G = (X , E), в якому кожна дуга відповідає вулиці в маршруті руху листоноші, а кожна вершина - стик двох вулиць. Ця задача являє собою задачу пошуку найкоротшого маршруту, який включає кожне ребро хоча б один раз і закінчується у початковій вершині руху. Нехай S-початкова вершина маршруту і a(i,j)>0 - довжина ребра (і, j). В графі існує декілька шляхів, по яким листоноша може обійти всі ребра і повернутись у вершину S. Ейлеревим циклом в графі називається шлях, який починається і закінчується в тій самій вершині, при чому всі ребра графа проходяться тільки один раз. Ейлеревим шляхом називається шлях, який починається в вершині А, а закінчується в вершині Б, і всі ребра проходяться лише по одному разу. Граф, який включає в себе ейлерів цикл називається ейлеревим. Якщо у графа всі вершини мають парну степінь, то цей граф Ейлерів, тобто має Ейлерів цикл.

**Код програми:**

[**https://github.com/Kruvyi/Lab\_2DS.git**](https://github.com/Kruvyi/Lab_2DS.git)

package com.kostya.postman\_lab2

import com.kostya.{Edge, Graph}

import scala.collection.mutable.ListBuffer

class PostMan(graph: Graph) {

private val cycles = new ListBuffer[ListBuffer[Edge]]

private val edges = graph.edges

def postManPath(): Unit = {

while (edges.nonEmpty)

cycles += makeCycle()

println("Cycles")

cycles.foreach(println)

if (cycles.length > 1)

mergeCycles()

}

def makeCycle(): ListBuffer[Edge] = {

val cycle = new ListBuffer[Edge]

cycle += edges.head

edges -= edges.head

while(cycle.last.vertex2 != cycle.head.vertex1) {

val edgesCopy = edges.clone()

addToCycle(cycle, edgesCopy)

}

cycle

}

def addToCycle(cycle: ListBuffer[Edge], edgesCopy: ListBuffer[Edge]): Unit = {

if (edgesCopy.nonEmpty) if (!cycle.contains(edgesCopy.head) && areConnected(cycle.last, edgesCopy)) {

cycle += edgesCopy.head

edges -= edgesCopy.head

addToCycle(cycle, edgesCopy.tail)

} else addToCycle(cycle, edgesCopy.tail) else println()

}

def mergeCycles(): ListBuffer[Edge] = {

val finalCycle = new ListBuffer[Edge]

val commonVertex = findCommonVertex(cycles(0), cycles(1))

if (commonVertex.contains("ERROR")) {

return finalCycle

}

var isDeleted = false

for (edge <- cycles(1)) {

if (!isDeleted && edge.vertex2 == commonVertex) {

finalCycle += edge

finalCycle ++= cycles(0)

cycles -= cycles(0)

isDeleted = true

}

else finalCycle += edge

}

if (cycles.length > 1) mergeCycles()

println("Final cycle: " + finalCycle)

finalCycle

}

def findCommonVertex(cycle1: ListBuffer[Edge], cycle2: ListBuffer[Edge]): String = {

for (edge1 <- cycle1) {

for (edge2 <- cycle2) {

if (edge1.vertex2 == edge2.vertex1) return edge1.vertex2

}

}

"No common vertex. ERROR"

}

def areConnected(edge1: Edge, edgesCopy: ListBuffer[Edge]): Boolean = {

val edge2 = edgesCopy.head

if (edge1.vertex2 == edge2.vertex1) {

true

} else if (edge1.vertex2 == edge2.vertex2) {

edges -= edge2

edgesCopy(0) = Edge(edge2.vertex2, edge2.vertex1, 1)

true

} else false

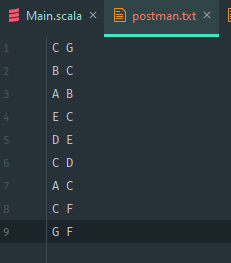
}

}

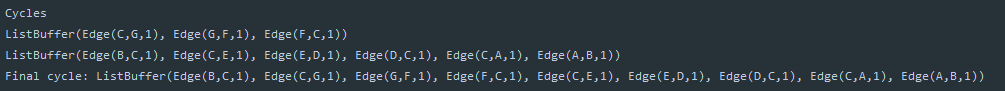
**Інструкція користувача:**

Для роботи програмі потрібно текстовий файл з списком ребер графа. Кожне ребро записується з нового рядка, через пробіл записується перша вершина ребра, друга вершина і вага дуги. Коли вхідний файл буде готовий потрібно запустити програму, після того як програма завершить свою роботу на екран буде виведено заданий список ребер і рішення задачі листоноші.

**Вхідний файл:**



**Результати:**



**Висновок:** у цій лабораторній роботі, я вивчив алгоритми рішення задачі листоноші. На практиці закріпив отримані знання. Використовуючи об'єктно-орієнтовану мову програмування Scala, реалізував алгоритм рішення задачі листоноші.