Universidade Federal Fluminense
Instituto de Computação
Programa de Pós-graduação em Ciência da Computação
TIC10002 – Estruturas de Dados e Algoritmos (2017.2)
Aluno: Bruno Olímpio
Matrícula: M048.216.005

Trabalho de implementação

O sistema de tráfego de uma região armazena o fluxo total de veículos por dia nas suas rodovias em arquivos organizados segundo o esquema Fluxo{setor,rodovia,dia,fluxo}. Faça um programa que apresente uma lista dos fluxos diários de tráfego por setor com esquema {setor,dia,fluxo} cujos fluxos sejam maiores que $min(fluxo)$ + 0.8Δ, onde Δ = max($fluxo$) − min($fluxo$). O programa deverá ser o mais eficiente possível. Explique as estruturas de dados utilizadas. O programa deverá considerar que novos arquivos são gerados a todo momento com novos dados. Nesse caso, novas listas de fluxo diário deverão ser produzidas.

Resoluções:

- Formato de entrada: CSV (Setor, Rodovia, Dia, Fluxo);
- Estruturas utilizadas:
    - Para processamento das entradas: Tabela hash com tamanho 50 e até 9 posições em lista para tratamento de colisões.
    - Para armazenamento e busca: Árvore AVL
- Formato da saída:
    - Impressão das linhas processadas;
    - Impressão da árvore gerada;
    - Impressão de fluxos mínimo e máximo, do delta e do limite de busca;
    - Impressão dos fluxos correspondentes à busca pelo limite
    - Impressão dos setores-dia correspondentes aos fluxos que satisfizerem ao limite.

Exemplo de arquivo de entrada (`fluxos.csv`)

```
NORTE,BR1,20170110,15
NORTE,BR3,20170110,35
SUL,BR2,20170111,45
LESTE,BR3,20170112,23
NORTE,BR1,20170111,43
LESTE,BR3,20170112,46
SUL,BR4,20170112,76
OESTE,BR1,20170111,35
NORTE,BR3,20170112,24
OESTE,BR2,20170114,23
SUL,BR4,20170110,45
SUL,BR2,20170112,67
NORTE,BR3,20170114,61
LESTE,BR1,20170114,72
NORTE,BR2,20170112,120
```

```
LESTE,BR4,20170110,84
LESTE,BR3,20170112,49
SUL,BR3,20170113,52
SUL,BR5,20170113,47
OESTE,BR3,20170111,46
SUL,BR5,20170113,41
OESTE,BR3,20170112,23
NORTE,BR2,20170114,22
SUL,BR1, 20170115,64
OESTE,BR2,20170115,71
NORTE,BR3,20170115,57
LESTE,BR3,20170116,51
NORTE,BR1,20170112,33
NORTE,BR3,20170115,56
SUL,BR2,20170115,24
LESTE,BR3,20170116,19
NORTE,BR1,20170116,29
LESTE,BR3,20170118,54
SUL,BR4,20170114,57
OESTE,BR1,20170117,34
NORTE,BR3,20170118,19
OESTE,BR2,20170123,12
SUL,BR4,20170112        ,75
SUL,BR2,20170113,23
NORTE,BR3,20170114,63
LESTE,BR1,20170115,72
NORTE,BR2,20170115,20
LESTE,BR4,20170113,43
LESTE,BR3,20170114,72
SUL,BR3,20170117,19
SUL,BR5,20170117,16
OESTE,BR3,20170115,72
SUL,BR5,2017016,25
OESTE,BR3,20170116,83
NORTE,BR2,20170116,26
SUL,BR1, 20170118,62
OESTE,BR2,20170119,71
NORTE,BR3,20170120,78
LESTE,BR3,20170121,36
```

Código do módulo construtor da AVL:

```python
# -*- coding: utf-8 -*-
"""
Created on Sun Nov  5 20:46:31 2017

@author: Bruno
"""

class No():
    def __init__(self, chave=None, valor=None, esquerda = None,
direita = None):
        self.valor = []
        self.chave = chave
        self.valor.append(valor)
        self.esquerda = esquerda
        self.direita = direita

    def retornaValor(self, chave):
```

```
        if self.chave == chave:
            print("Fluxo: {} --> Dia-setor: {}".format(self.chave,
self.valor))

        elif self.chave < chave:
            self.direita.retornaValor(chave)

        elif self.chave > chave:
            self.esquerda.retornaValor(chave)

    def retornaMin(self):
        if self.esquerda.no:
            self.esquerda.no.retornaMin()
        else:
            minFluxo = self.chave

    def retornaMax(self):
        if self.direita:
            self.direita.no.retornaMax(self)
        else:
            maxFluxo = self.chave
        return maxFluxo

class AVL():
    def __init__(self):
        self.no = None
        self.altura = -1
        self.saldo = 0

    def inclui(self, chave, valor):
        no = No(chave, valor)

        if self.no == None:
            self.no = no
            self.no.esquerda = AVL()
            self.no.direita = AVL()
        elif chave < self.no.chave:
            self.no.esquerda.inclui(chave, valor)
        elif chave > self.no.chave:
            self.no.direita.inclui(chave, valor)
        elif self.no.chave == chave:
            self.no.valor.append(valor)
        self.equilibra()

    def remove(self, chave):
        if self.no != None:
            if self.no.chave == chave:
                if not self.no.esquerda.no and not self.no.direita.no:
                    self.no = None

                elif not self.no.direita.no:
                    self.no = self.no.esquerda.no

                elif not self.no.esquerda.no:
                    self.no = self.no.direita.no

                else:
                    sucessor = self.no.direita.no
                    while sucessor and sucessor.esquerda.no:
                        sucessor = sucessor.esquerda.no

                    if sucessor:
                        self.no.chave = sucessor.chave
                        self.no.direita.remove(sucessor.chave)
```

```
            elif chave < self.no.chave:
                self.no.esquerda.remove(chave)

            elif chave > self.no.chave:
                self.no.direita.remove(chave)

            self.equilibra()

    def equilibra(self):
        self.atualizaAlturas(recursiva=False)
        self.atualizaSaldos(False)

        while self.saldo < -1 or self.saldo > 1:
            if self.saldo > 1:
                if self.no.esquerda.saldo < 0:
                    self.no.esquerda.trocaEsquerda()
                    self.atualizaSaldos()
                    self.atualizaAlturas()
                self.trocaDireita()
                self.atualizaSaldos()
                self.atualizaAlturas()
            if self.saldo < -1:
                if self.node.direita.saldo > 0:
                    self.no.direita.trocaDireita()
                    self.atualizaSaldos()
                    self.atualizaAlturas()
                self.trocaEsquerda()
                self.atualizaSaldos()
                self.atualizaAlturas()

    def atualizaSaldos(self, recursiva=True):
        if self.no:
            if recursiva:
                if self.no.esquerda:
                    self.no.esquerda.atualizaSaldos()
                if self.no.direita:
                    self.no.direita.atualizaSaldos()
            self.saldo = self.no.esquerda.saldo -
self.no.direita.saldo
        else:
            self.saldo = 0
#         print("Saldo = {}".format(self.saldo))

    def atualizaAlturas(self, recursiva=True):
        if self.no:
            if recursiva:
                if self.no.esquerda:
                    self.no.esquerda.atualizaAlturas()
                if self.no.direita:
                    self.no.direita.atualizaAlturas()
            self.altura = 1 + max(self.no.direita.altura,
self.no.esquerda.altura)
        else:
            self.altura = -1

    def trocaEsquerda(self):
        novo_raiz = self.no.direita.no
        novo_esquerda = novo_raiz.esquerda.no
        raiz = self.no
        self.no = novo_raiz
        raiz.direita.no = novo_esquerda
        novo_raiz.esquerda.no = raiz

    def trocaDireita(self):
        novo_raiz = self.no.esquerda.no
```

```
        novo_esquerda = novo_raiz.direita.no
        raiz = self.no
        self.no = novo_raiz
        raiz.esquerda.no = novo_esquerda
        novo_raiz.direita.no = raiz

    def percorre(self):
        resultado = []
        if not self.no:
            return resultado
        resultado.extend(self.no.esquerda.percorre())
        resultado.append(self.no.chave)
        resultado.extend(self.no.direita.percorre())
        return resultado

    def imprime(self, no=None, nivel=0):
        if not no:
            no = self.no

        if no.direita.no:
            self.imprime(no.direita.no, nivel + 1)
            print(('\t' * nivel), ('    /'))

        print(('\t' * nivel), no.chave)

        if no.esquerda.no:
            print(('\t' * nivel), ('    \\'))
            self.imprime(no.esquerda.no, nivel + 1)


    def relatorioLimite(self,limite):
        relatorio = []
        resultado = self.percorre()
        for chave in resultado:
            if chave > limite:
                relatorio.append(chave)
        print("Os fluxos maiores do que {} são:
{}".format(limite,relatorio))
        for chave in relatorio:
            self.retornaValor(chave)

    def retornaValor(self, chave):
        self.no.retornaValor(chave)

    def retornaMin(self):
        return self.no.retornaMin

    def retornaMax(self):
        self.no.retornaMax
```

Código do módulo principal:

```
# -*- coding: utf-8 -*-
"""
Created on Mon Oct 16 14:20:17 2017

@author: Bruno Olímpio
"""

### Trabalho 1 ED ###

import csv
from AVL import AVL

def processaFluxos(arquivo):
    with open(arquivo, newline='') as csvfile:
        fluxos = csv.reader(csvfile,delimiter=',');
        for row in fluxos:
            dia = int(row[2]);
            setor = row[0];
            fluxo = int(row[3]);
            atualizaMapaDia(dia, setor, fluxo);



def atualizaMapaDia(dia, setor, fluxo):
    setorDia = setor + str(dia)
    mapaFluxo = [[fluxo, setorDia]]
    i = hash(setorDia)
    tabelaHash[i] = mapaFluxo
    fluxoTotal = mapaFluxo[0][0]
    if not fluxoTotal:
        fluxoTotal = 0;
    fluxoTotal = fluxoTotal + fluxo


tabelaHash = [[0 for i in range(10)]for i in range(50)];
def hash(setorDia):
    soma = 0
    for i in range(len(setorDia)):
        soma = soma + ord(setorDia[i])
    hashSetorDia = soma%len(tabelaHash)
    return(hashSetorDia)

def calculaDelta(minFluxo, maxFluxo):
    delta = maxFluxo - minFluxo;
    print("minFluxo: {} ---- maxFluxo: {} ---- Delta: {}
\n".format(minFluxo,maxFluxo, delta))
    return delta

def criaLimite(minFluxo,delta):
    limite = minFluxo + int(0.8 * delta)
    return limite

############### EXECUTANDO #################

arvore = AVL()
processaFluxos('fluxo.csv')
for mapaFluxo in tabelaHash:
    if (mapaFluxo[0]) != 0:
        arvore.inclui(mapaFluxo[0][0], mapaFluxo[0][1])

arvore.imprime()
minFluxo = arvore.percorre()[0]
maxFluxo = arvore.percorre()[-1]
```

```
delta = calculaDelta(minFluxo, maxFluxo)
lim = criaLimite(minFluxo, delta)
arvore.relatorioLimite(lim)
```

Exemplo de saída com o arquivo de entrada apresentado:

```
84
     \
                            83
                                \
                                   78
                           /
                       72
                           \
                                   63
                                       \
                                          62
                                /
                            45
                    /
                43
            /
        36
            \
                            35
                           /
                       34
                           \
                            33
                                \
                                   26
                                       \
                                          25
                                             /
                                          24
                                /
                            20
                                \
                                   16
                    /
                12
minFluxo: 12 ---- maxFluxo: 84 ---- Delta: 72

Os fluxos maiores do que o limite de 69 são: [72, 78, 83, 84]
Fluxo: 72 --> Dia-setor: ['LESTE20170114', 'LESTE20170115', 'OESTE20170115']
Fluxo: 78 --> Dia-setor: ['NORTE20170120']
Fluxo: 83 --> Dia-setor: ['OESTE20170116']
Fluxo: 84 --> Dia-setor: ['LESTE20170110']
```