UEZO
FUNDAÇÃO CENTRO UNIVERSITÁRIO
ESTADUAL DA ZONA OESTE
Rio de Janeiro / RJ / Brasil

ALGORIMO DE BUSCA LARGURA

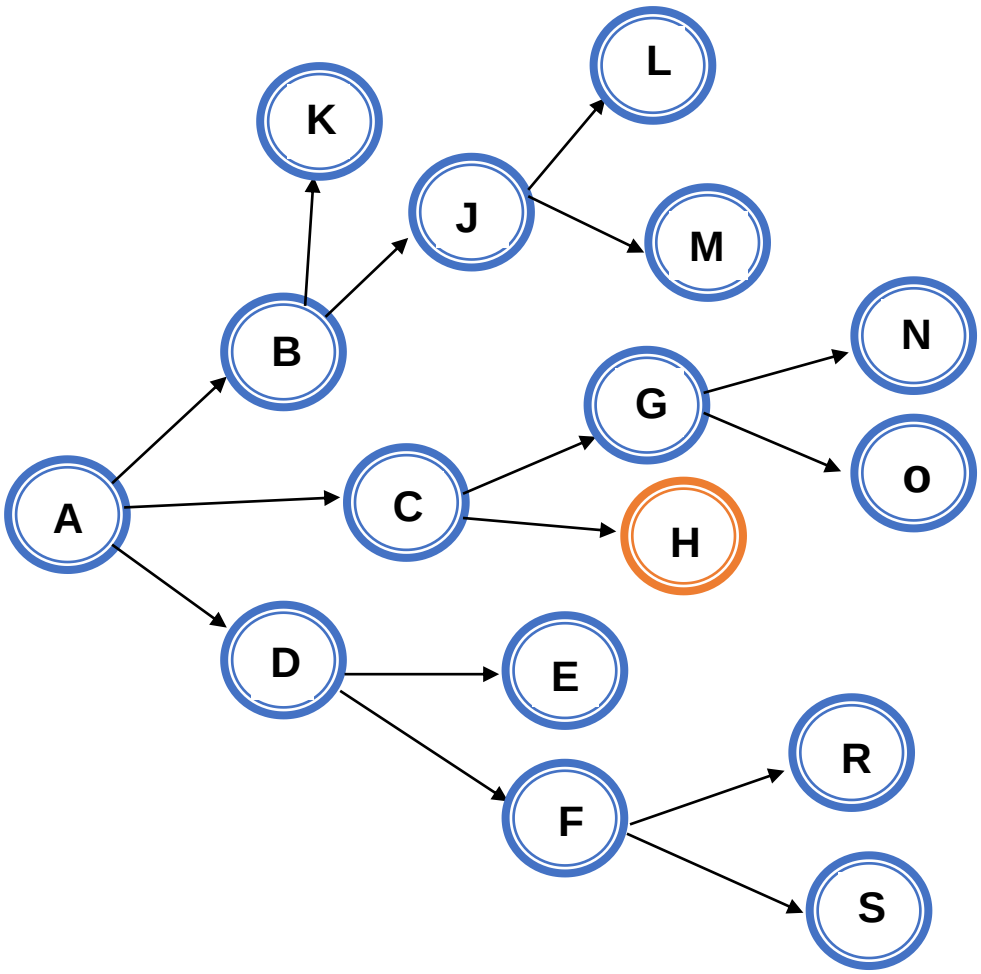
Inteligência Computacional I

Ana Cláudia Gomes Souza

Rio de Janiero, Maio 2019

Para a implementação do algoritmo de busca largura foi utilizada a linguagem Python. A estrutura do código é composta por um TAD de cinco arquivos. Com o auxilio de um mapa composto de letras do alfabeto será realizada a entrada de dados. O objetivo do algoritmo é percorrer os estados do mapa para encontrar a letra H.

Mapa de letras



Descrição dos arquivos do TAD

Arquivo Fila.py

```
class Fila:
    def __init__(self,tamanho):
        self.tamanho = tamanho
        self.letras = [None] * self.tamanho
        self.inicio = 0
        self.fim = -1
        self.numerosElementos = 0

    def enfileirar(self, letras):
        if not Fila.filaCheia(self):
            if self.fim == self.tamanho - 1:
                self.fim = -1
            self.fim +=1
            self.letras[self.fim] = letras
            self.numerosElementos += 1

        else:
            print('A fila está cheia!')


    def desenfileirar(self):
        if not Fila.filaVazia(self):
            temp = self.letras[self.inicio]
            self.inicio += 1
            if self.inicio == self.tamanho:
                self.inicio = 0
            self.numerosElementos -= 1
            return temp
        else:
            print('A fila já está vazia!')
            return None

    def getPrimeiro(self):
        return self.letras[self.inicio]
```

```
def filaVazia(self):
    return self.numerosElementos == 0

def filaCheia(self):
    return self.numerosElementos == self.tamanho
```

Neste arquivo foi criada a classe Fila, os objetos tamanho responsável por receber o tamanho da fila, letras responsável pelo armazenamento dos dados da fila, inicio responsável pela entrada de valores, fim responsável pela saída de valores (esvaziamento da fila) e numeroElementos responsável pela contagem de elementos contidos na fila. Também foram implementados os métodos filaCheia para verificar se a fila está cheia, filaVazia para verificar se a fila está vazia, getPrimeiro método que retorna o primeiro valor da fila, enfileirar método

utilizado para o preenchimento de dados na fila e o método desinfileirar utilizado para retirar os dados da fila

Arquivo Letras.py

```
class Letras:
    def __init__(self,nome):
        self.nome = nome
        self.visitado = False
        self.adjacentes = []

    def AddLetrasAdjacentes(self,letras):
        self.adjacentes.append(letras)
```

Neste arquivo foi criada a classe Letras e os objetos nome, visitado e adjacentes. Também foi implementado o método AddLetrasAdjacentes que é responsável por adicionar os valores que são adjacentes de um estado

Arquivo Adjacentes.py

```
class Adjacentes:
    def __init__(self,letras):
        self.letras = letras
```

Neste arquivo foi implementado o método que insere os valores adjacentes de um estado na lista letras

Arquivo Mapa.py

```
from Letras import Letras
from Adjacentes import Adjacentes

class Mapa:
    A = Letras("A")
    B = Letras("B")
    C = Letras("C")
    D = Letras("D")
    E = Letras("E")
    F = Letras("F")
    G = Letras("G")
    H = Letras("H")
    J = Letras("J")
    K = Letras("H")
    L = Letras("L")
    M = Letras("M")
    N = Letras("N")
    O = Letras("O")
    R = Letras("R")
    S = Letras("S")


    '''Adicionando as letras que são adjacentes a letra A'''
    A.AddLetrasAdjacentes(Adjacentes(B))
    A.AddLetrasAdjacentes(Adjacentes(C))
    A.AddLetrasAdjacentes(Adjacentes(D))

    '''Adicionando as letras que são adjacentes a letra B'''
    B.AddLetrasAdjacentes(Adjacentes(K))
    B.AddLetrasAdjacentes(Adjacentes(J))

    '''Adicionando as letras que são adjacentes a letra J'''
    J.AddLetrasAdjacentes(Adjacentes(L))
    J.AddLetrasAdjacentes(Adjacentes(M))

    '''Adicionando as letras que são adjacentes a letra C'''
    C.AddLetrasAdjacentes(Adjacentes(G))
    C.AddLetrasAdjacentes(Adjacentes(H))

    '''Adicionando as letras que são adjacentes a letra G'''
    G.AddLetrasAdjacentes(Adjacentes(N))
    G.AddLetrasAdjacentes(Adjacentes(O))

    '''Adicionando as letras que são adjacentes a letra D'''
    D.AddLetrasAdjacentes(Adjacentes(E))
    D.AddLetrasAdjacentes(Adjacentes(F))

    '''Adicionando as letras que são adjacentes a letra F'''
    F.AddLetrasAdjacentes(Adjacentes(R))
    F.AddLetrasAdjacentes(Adjacentes(S))
```

Neste arquivo foi criada a classe Mapa e os objetos que repesentam a letra de cada estado, também foi implementado o método AddLetrasAdjacentes que adiciona na lista os valores adjacentes de cada estado, para isso importamos os arquivos Letras e Adjacentes que contém as classes Letras e Adjacentes.

Arquivo Largura.py

```
from Fila import Fila

class Largura:
    def __init__(self,inicio,objetivo):
        self.inicio = inicio
        self.inicio.visitado = True
        self.objetivo = objetivo
        self.fronteira = Fila(16)
        self.fronteira.enfileirar(inicio)
        self.achou = False

    def buscar(self):
        primeiro = self.fronteira.getPrimeiro()
        print('Primeiro elemento da fila'.format(primeiro.nome))

        if primeiro == self.objetivo:
            self.achou = True
        else:
            temp = self.fronteira.desenfileirar()
            print('Desenfileirou {}'.format(temp.nome))

            for a in primeiro.adjacentes:
                print('Verificando se já foi visitado: {}'.format(a.letras.nome))
                if a.letras.visitado == False:
                    a.letras.visitado = True
                    self.fronteira.enfileirar(a.letras)
            if self.fronteira.numerosElementos > 0:
                Largura.buscar(self)

from Mapa import Mapa
mapa = Mapa()
largura = Largura(mapa.A,mapa.H)
largura.buscar()
```

Neste arquivo foi criada a classe Largura e os objetos inicio responsável por armazenar o valor inicial, inicio.visitado responsável por verificar se um estado foi ou não visitado, objetivo responsável por armazenar o valor do objetivo, fronteira responsável por armazenar os valores adjacentes de um estado e achou responsável por armazenar o valor lógico True ou False quanto a descoberta do valor objetivo. Também foi implementado o método buscar que faz uma verificação dos dados da lista do primeiro ao último elemento em busca do valor correspondente ao objetivo.

Impressão da saída do processamento dos dados da lista letras

```
Primeiro elemento da fila
Desenfileirou A
Verificando se já foi visitado: B
Verificando se já foi visitado: C
Verificando se já foi visitado: D
Primeiro elemento da fila
Desenfileirou B
Verificando se já foi visitado: H
Verificando se já foi visitado: J
Primeiro elemento da fila
Desenfileirou C
Verificando se já foi visitado: G
Verificando se já foi visitado: H
Primeiro elemento da fila
Desenfileirou D
Verificando se já foi visitado: E
Verificando se já foi visitado: F
Primeiro elemento da fila
Desenfileirou H
Primeiro elemento da fila
Desenfileirou J
Verificando se já foi visitado: L
Verificando se já foi visitado: M
Primeiro elemento da fila
Desenfileirou G
Verificando se já foi visitado: N
Verificando se já foi visitado: O
Primeiro elemento da fila
```