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

ALGORITMO DE BUSCA ORDENADA

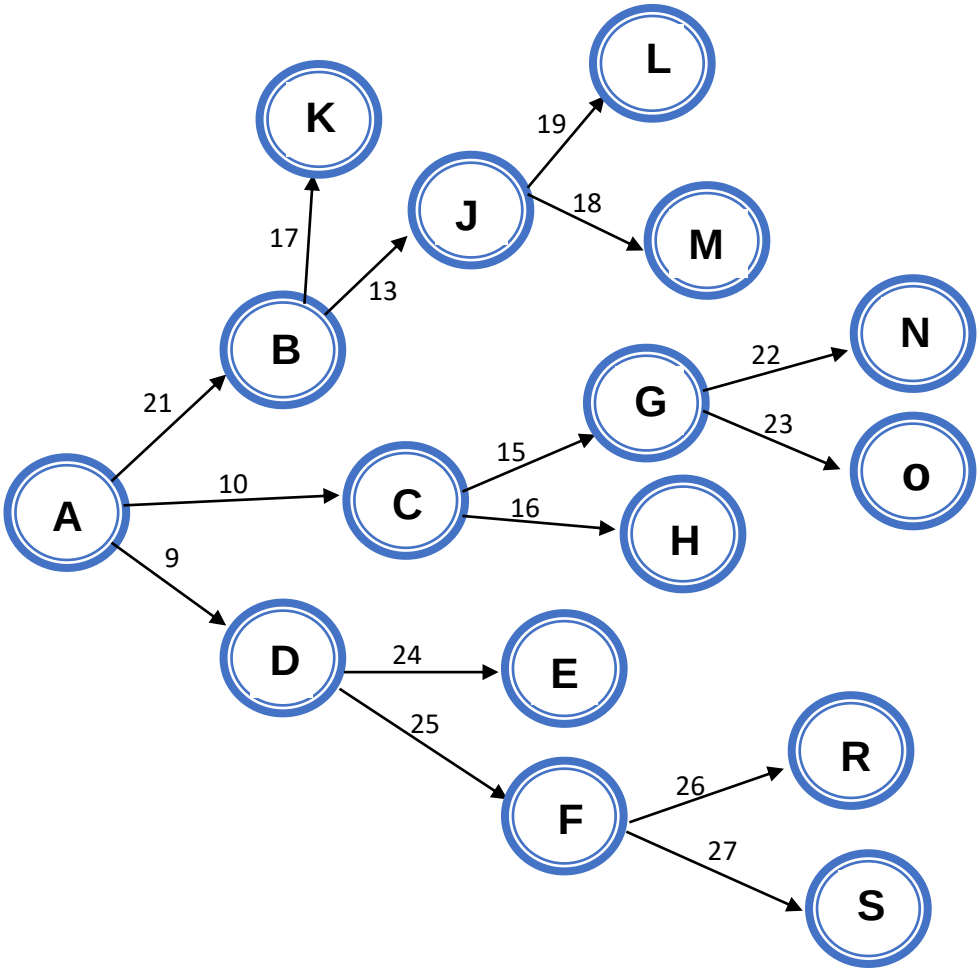
Inteligência Computacional I

Ana Cláudia Gomes Souza

Rio de Janiero, Maio 2019

Para a implementação do algoritmo de busca ordenada 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 informada pelo usuário via teclado e calcular o custo que esta busca gerou levando em consideração a distância entre os nós.

Mapa de letras



Descrição dos arquivos que compõem o TAD:

Arquivo Letras.py

Este arquivo contém duas funções def __init__(self,nome,distancia) sendo esta responsável por receber as letras e a distância de cada letra e a função def AddLetrasAdjacentes(self,letras) responsável por adicionar os nós adjacentes no mapa de letras.

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

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

```

Arquivo Adjacentes.py

Este arquivo implementa a classe Adjacentes utilizada no método AddLetrasAdjacentes para adicionar as letras no mapa.

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

```

Arquivo Mapa.py

Este arquivo é responsável por criar o mapa de letras como parte do desenvolvimento do código importamos os arquivos Letras e Adjacentes e suas classes e métodos para tornar possível a sua implementação.

```python
from Letras import Letras
from Adjacentes import Adjacentes

class Mapa:
    A = Letras('A',0)
    B = Letras('B',21)
    K = Letras('K',17)
    J = Letras('J',13)
    L = Letras('L',19)
    M = Letras('M',18)
    C = Letras('C',10)
    G = Letras('G',15)
    H = Letras('H',16)
    N = Letras('N',22)
    O = Letras('O',23)
    D = Letras('D',9)
    E = Letras('E',24)
    F = Letras('F',25)
    R = Letras('R',26)
    S = Letras('S',27)

    '''Nós Adjacentes do nó raiz A'''
    A.AddLetrasAdjacentes(Adjacentes(B))
    A.AddLetrasAdjacentes(Adjacentes(C))
    A.AddLetrasAdjacentes(Adjacentes(D))

    '''Nós Adjacentes do nó B'''
    B.AddLetrasAdjacentes(Adjacentes(K))
    B.AddLetrasAdjacentes(Adjacentes(J))

    '''Nós Adjacentes do nó C'''
    C.AddLetrasAdjacentes(Adjacentes(G))
    C.AddLetrasAdjacentes(Adjacentes(H))

    '''Nós Adjacentes do nó D'''
    D.AddLetrasAdjacentes(Adjacentes(E))
    D.AddLetrasAdjacentes(Adjacentes(F))


    '''Nós Adjacentes do nó J'''
    J.AddLetrasAdjacentes(Adjacentes(L))
    J.AddLetrasAdjacentes(Adjacentes(M))

    '''Nós Adjacentes do nó G'''
    G.AddLetrasAdjacentes(Adjacentes(N))
    G.AddLetrasAdjacentes(Adjacentes(O))


    '''Nós Adjacentes do nó F'''
    F.AddLetrasAdjacentes(Adjacentes(R))
    F.AddLetrasAdjacentes(Adjacentes(S))
```

Arquivo VetorOrdenado.py

Este arquivo contém quatro funções responsáveis pela ordenação e busca dos valores do mapa de letras a função def __init__(self,tamanho) recebe o tamanho do vetor e o cria, a função def inserir(self,Letras) insere os valores no vetor, no caso são as letras e suas distâncias e os armazena no vetor de forma ordenada obedecendo a precedência que foram inseridos no mapa, a função def mostrar(self) responsável por mostrar os dados do vetor ordenado e afunção def buscar(self) que verifica se a letra digitada pelo usuário existe ou não no mapa de letras e qual a distância (custo) total precorrida para cheagar até o nó.

```
class VetorOrdenado:
    def __init__(self,tamanho):
        self.numeroElementos = 0
        self.letras = [None] * tamanho

    def inserir(self,Letras):
        if self.numeroElementos == 0:
            self.letras[0] = Letras
            self.numeroElementos = 1
        else:
            self.letras[self.numeroElementos] = Letras
            self.numeroElementos += 1

    def mostrar(self):
        print(' \n\n Nó  CUSTO ')
        for i in range(0,self.numeroElementos):
            print('  {}    {}      '.format(self.letras[i].nome,self.letras[i].distancia))

```

```
     def buscar(self):
         l = input('\nDigite uma letra para buscar no mapa: ')
         busca = l.upper()
         if busca == 'A':
             custo = self.letras[0].distancia
             print('\nA letra ',busca,'foi encontrada na posição 0 com um custo',custo )

         elif busca == 'B':
             custo =+ self.letras[1].distancia
             print('\nA letra ',busca,'foi encontrada na posição 1 com um custo',custo )

         elif busca == 'C':
             custo =+ self.letras[1].distancia
             custo += self.letras[2].distancia
             print('\nA letra ',busca,'foi encontrada na posição 2 com um custo',custo )

         elif busca == 'D':
             custo =+ self.letras[1].distancia
             custo += self.letras[2].distancia
             custo += self.letras[3].distancia
             print('\nA letra ',busca,'foi encontrada na posição 3 com um custo',custo )

         elif busca == 'E':
             custo =+ self.letras[1].distancia
             custo += self.letras[2].distancia
             custo += self.letras[3].distancia
             custo += self.letras[4].distancia
             custo += self.letras[5].distancia
             custo += self.letras[6].distancia
             custo += self.letras[7].distancia
             custo += self.letras[8].distancia
             custo += self.letras[9].distancia
             custo += self.letras[10].distancia
             custo += self.letras[11].distancia
             custo += self.letras[12].distancia
             print('\nA letra ',busca,'foi encontrada na posição 12 com um custo',custo )
```

```
        elif busca == 'F':
            custo =+ self.letras[1].distancia
            custo += self.letras[2].distancia
            custo += self.letras[3].distancia
            custo += self.letras[4].distancia
            custo += self.letras[5].distancia
            custo += self.letras[6].distancia
            custo += self.letras[7].distancia
            custo += self.letras[8].distancia
            custo += self.letras[9].distancia
            custo += self.letras[10].distancia
            custo += self.letras[11].distancia
            custo += self.letras[12].distancia
            custo += self.letras[13].distancia
            print('\nA letra ',busca,'foi encontrada na posição 13 com um custo',custo )

        elif busca == 'G':
            custo =+ self.letras[1].distancia
            custo += self.letras[2].distancia
            custo += self.letras[3].distancia
            custo += self.letras[4].distancia
            custo += self.letras[5].distancia
            custo += self.letras[6].distancia
            custo += self.letras[7].distancia
            custo += self.letras[8].distancia
            print('\nA letra ',busca,'foi encontrada na posição 8 com um custo',custo )

        elif busca == 'H':
            custo =+ self.letras[1].distancia
            custo += self.letras[2].distancia
            custo += self.letras[3].distancia
            custo += self.letras[4].distancia
            custo += self.letras[5].distancia
            custo += self.letras[6].distancia
            custo += self.letras[7].distancia
            custo += self.letras[8].distancia
            custo += self.letras[9].distancia
            print('\nA letra ',busca,'foi encontrada na posição 9 com um custo',custo )


        elif busca == 'J':
            custo =+ self.letras[1].distancia
            custo += self.letras[2].distancia
            custo += self.letras[3].distancia
            custo += self.letras[4].distancia
            custo += self.letras[5].distancia
            print('\nA letra ',busca,'foi encontrada na posição 5 com um custo',custo )

        elif busca == 'K':
            custo =+ self.letras[1].distancia
            custo += self.letras[2].distancia
            custo += self.letras[3].distancia
            custo += self.letras[4].distancia
            print('\nA letra ',busca,'foi encontrada na posição 4 com um custo',custo )

        elif busca == 'M':
            custo =+ self.letras[1].distancia
            custo += self.letras[2].distancia
            custo += self.letras[3].distancia
            custo += self.letras[4].distancia
            custo += self.letras[5].distancia
            custo += self.letras[6].distancia
            custo += self.letras[7].distancia
            print('\nA letra ',busca,'foi encontrada na posição 7 com um custo',custo )

        elif busca == 'L':
            custo =+ self.letras[1].distancia
            custo += self.letras[2].distancia
            custo += self.letras[3].distancia
            custo += self.letras[4].distancia
            custo += self.letras[5].distancia
            custo += self.letras[6].distancia
            print('\nA letra ',busca,'foi encontrada na posição 6 com um custo',custo )

```

```
        elif busca == 'N':
            custo =+ self.letras[1].distancia
            custo += self.letras[2].distancia
            custo += self.letras[3].distancia
            custo += self.letras[4].distancia
            custo += self.letras[5].distancia
            custo += self.letras[6].distancia
            custo += self.letras[7].distancia
            custo += self.letras[8].distancia
            custo += self.letras[9].distancia
            custo += self.letras[10].distancia
            print('\nA letra ',busca,'foi encontrada na posição 10 com um custo',custo )

        elif busca == 'O':
            custo =+ self.letras[1].distancia
            custo += self.letras[2].distancia
            custo += self.letras[3].distancia
            custo += self.letras[4].distancia
            custo += self.letras[5].distancia
            custo += self.letras[6].distancia
            custo += self.letras[7].distancia
            custo += self.letras[8].distancia
            custo += self.letras[9].distancia
            custo += self.letras[10].distancia
            custo += self.letras[11].distancia
            print('\nA letra ',busca,'foi encontrada na posição 11 com um custo',custo )

        elif busca == 'R':
            custo =+ self.letras[1].distancia
            custo += self.letras[2].distancia
            custo += self.letras[3].distancia
            custo += self.letras[4].distancia
            custo += self.letras[5].distancia
            custo += self.letras[6].distancia
            custo += self.letras[7].distancia
            custo += self.letras[8].distancia
            custo += self.letras[9].distancia
            custo += self.letras[10].distancia
            custo += self.letras[11].distancia
            custo += self.letras[12].distancia
            custo += self.letras[13].distancia
            custo += self.letras[14].distancia
            print('\nA letra ',busca,'foi encontrada na posição 14 com um custo',custo )

        elif busca == 'S':
            custo =+ self.letras[1].distancia
            custo += self.letras[2].distancia
            custo += self.letras[3].distancia
            custo += self.letras[4].distancia
            custo += self.letras[5].distancia
            custo += self.letras[6].distancia
            custo += self.letras[7].distancia
            custo += self.letras[8].distancia
            custo += self.letras[9].distancia
            custo += self.letras[10].distancia
            custo += self.letras[11].distancia
            custo += self.letras[12].distancia
            custo += self.letras[13].distancia
            custo += self.letras[14].distancia
            custo += self.letras[15].distancia
            print('\nA letra ',busca,'foi encontrada na posição 14 com um custo',custo )

        else:
            print('\nA letra ',busca,'não pertence ao mapa!')
```

Arquivo BuscaOrdenada.py

Neste arquivo é criada a função def CriarMapaOrdenado() responsável por criar um novo mapa de letras de forma que as distâncias estejam ordenadas na precedência em que as letras foram inseridas no mapa. Também é aplicado o método buscar() que retorna os resultados da busca e mostrar() que apresenta a nova ordenação dos dados do mapa de letras.

```
from VetorOrdenado import VetorOrdenado
from Mapa import Mapa
mapa = Mapa()
vetor = VetorOrdenado(16)

def CriarMapaOrdenado():
    vetor.inserir(mapa.A)
    vetor.inserir(mapa.B)
    vetor.inserir(mapa.C)
    vetor.inserir(mapa.D)
    vetor.inserir(mapa.K)
    vetor.inserir(mapa.J)
    vetor.inserir(mapa.L)
    vetor.inserir(mapa.M)
    vetor.inserir(mapa.G)
    vetor.inserir(mapa.H)
    vetor.inserir(mapa.N)
    vetor.inserir(mapa.O)
    vetor.inserir(mapa.E)
    vetor.inserir(mapa.F)
    vetor.inserir(mapa.R)
    vetor.inserir(mapa.S)


CriarMapaOrdenado()
vetor.buscar()
vetor.mostrar()

```

Dados ordenados do MapaOrdenado

Nó	CUSTO
A	0
B	21
C	10
D	9
K	17
J	13
L	19
M	18
G	15
H	16
N	22
O	23
E	24
F	25
R	26
S	27

Realizando buscas no mapa

```
Digite uma letra para buscar no mapa: g

A letra  G foi encontrada na posição 8 com um custo 122
```

```
Digite uma letra para buscar no mapa: x

A letra  X não pertence ao mapa!
```