

ALGORITMO DE BUSCA ORDENADA

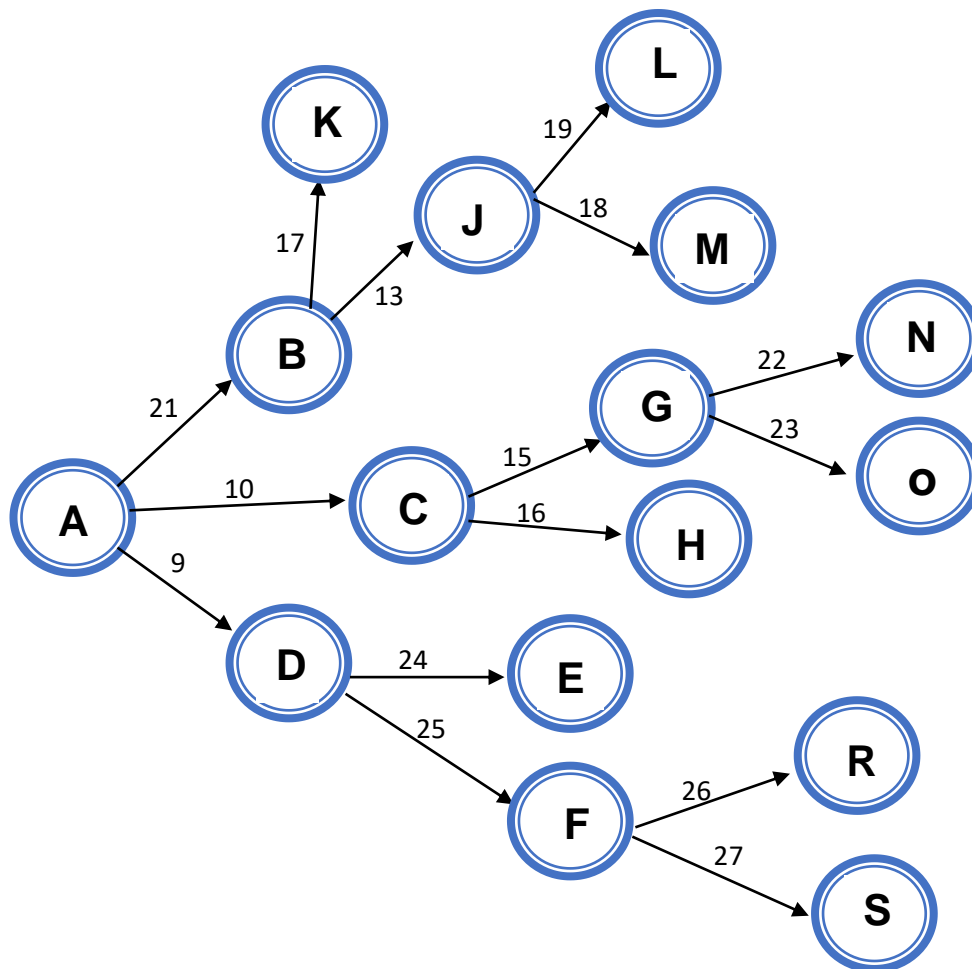
Inteligência Computacional I

Ana Cláudia Gomes Souza

Rio de Janeiro, 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 auxílio 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.

```
1 class Letras:
2     def __init__(self,nome,distancia):
3         self.nome = nome
4         self.distancia = distancia
5         self.visitado = False
6         self.adjacentes = []
7
8     def AddLetrasAdjacentes(self,letras):
9         self.adjacentes.append(letras)
10
```

Arquivo Adjacentes.py

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

```
1 class Adjacentes:
2     def __init__(self,letras):
3         self.letras = letras
4
```

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.

```

1 from Letras import Letras
2 from Adjacentes import Adjacentes
3
4 class Mapa:
5     A = Letras('A',0)
6     B = Letras('B',21)
7     K = Letras('K',17)
8     J = Letras('J',13)
9     L = Letras('L',19)
10    M = Letras('M',18)
11    C = Letras('C',10)
12    G = Letras('G',15)
13    H = Letras('H',16)
14    N = Letras('N',22)
15    O = Letras('O',23)
16    D = Letras('D',9)
17    E = Letras('E',24)
18    F = Letras('F',25)
19    R = Letras('R',26)
20    S = Letras('S',27)
21
22    '''Nós Adjacentes do nó raiz A'''
23    A.AddLetrasAdjacentes(Adjacentes(B))
24    A.AddLetrasAdjacentes(Adjacentes(C))
25    A.AddLetrasAdjacentes(Adjacentes(D))
26
27    '''Nós Adjacentes do nó B'''
28    B.AddLetrasAdjacentes(Adjacentes(K))
29    B.AddLetrasAdjacentes(Adjacentes(J))
30
31    '''Nós Adjacentes do nó C'''
32    C.AddLetrasAdjacentes(Adjacentes(G))
33    C.AddLetrasAdjacentes(Adjacentes(H))
34
35    '''Nós Adjacentes do nó D'''
36    D.AddLetrasAdjacentes(Adjacentes(E))
37    D.AddLetrasAdjacentes(Adjacentes(F))
38
39
40    '''Nós Adjacentes do nó J'''
41    J.AddLetrasAdjacentes(Adjacentes(L))
42    J.AddLetrasAdjacentes(Adjacentes(M))
43
44    '''Nós Adjacentes do nó G'''
45    G.AddLetrasAdjacentes(Adjacentes(N))
46    G.AddLetrasAdjacentes(Adjacentes(O))
47
48
49    '''Nós Adjacentes do nó F'''
50    F.AddLetrasAdjacentes(Adjacentes(R))
51    F.AddLetrasAdjacentes(Adjacentes(S))
52

```

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 a função `def buscar(self)` que verifica se a letra digitada pelo usuário existe ou não no mapa de letras e qual a distância (custo) total percorrida para chegar até o nó.

```
1 class VetorOrdenado:
2     def __init__(self,tamanho):
3         self.numeroElementos = 0
4         self.letras = [None] * tamanho
5
6     def inserir(self,Letras):
7         if self.numeroElementos == 0:
8             self.letras[0] = Letras
9             self.numeroElementos = 1
10        else:
11            self.letras[self.numeroElementos] = Letras
12            self.numeroElementos += 1
13
14    def mostrar(self):
15        print(' \n\n Nó    CUSTO ')
16        for i in range(0,self.numeroElementos):
17            print(' {}    {}          '.format(self.letras[i].nome,self.letras[i].distancia))
18
19
20    def buscar(self):
21        l = input('\nDigite uma letra para buscar no mapa: ')
22        busca = l.upper()
23        if busca == 'A':
24            custo = self.letras[0].distancia
25            print('\nA letra ',busca,'foi encontrada na posição 0 com um custo',custo )
26
27        elif busca == 'B':
28            custo += self.letras[1].distancia
29            print('\nA letra ',busca,'foi encontrada na posição 1 com um custo',custo )
30
31        elif busca == 'C':
32            custo += self.letras[1].distancia
33            custo += self.letras[2].distancia
34            print('\nA letra ',busca,'foi encontrada na posição 2 com um custo',custo )
35
36        elif busca == 'D':
37            custo += self.letras[1].distancia
38            custo += self.letras[2].distancia
39            custo += self.letras[3].distancia
40            print('\nA letra ',busca,'foi encontrada na posição 3 com um custo',custo )
41
42        elif busca == 'E':
43            custo += self.letras[1].distancia
44            custo += self.letras[2].distancia
45            custo += self.letras[3].distancia
46            custo += self.letras[4].distancia
47            custo += self.letras[5].distancia
48            custo += self.letras[6].distancia
49            custo += self.letras[7].distancia
50            custo += self.letras[8].distancia
51            custo += self.letras[9].distancia
52            custo += self.letras[10].distancia
53            custo += self.letras[11].distancia
54            custo += self.letras[12].distancia
55            print('\nA letra ',busca,'foi encontrada na posição 12 com um custo',custo )
```

```

57     elif busca == 'F':
58         custo += self.letras[1].distancia
59         custo += self.letras[2].distancia
60         custo += self.letras[3].distancia
61         custo += self.letras[4].distancia
62         custo += self.letras[5].distancia
63         custo += self.letras[6].distancia
64         custo += self.letras[7].distancia
65         custo += self.letras[8].distancia
66         custo += self.letras[9].distancia
67         custo += self.letras[10].distancia
68         custo += self.letras[11].distancia
69         custo += self.letras[12].distancia
70         custo += self.letras[13].distancia
71         print('\nA letra ',busca,'foi encontrada na posição 13 com um custo',custo )
72
73     elif busca == 'G':
74         custo += self.letras[1].distancia
75         custo += self.letras[2].distancia
76         custo += self.letras[3].distancia
77         custo += self.letras[4].distancia
78         custo += self.letras[5].distancia
79         custo += self.letras[6].distancia
80         custo += self.letras[7].distancia
81         custo += self.letras[8].distancia
82         print('\nA letra ',busca,'foi encontrada na posição 8 com um custo',custo )
83
84     elif busca == 'H':
85         custo += self.letras[1].distancia
86         custo += self.letras[2].distancia
87         custo += self.letras[3].distancia
88         custo += self.letras[4].distancia
89         custo += self.letras[5].distancia
90         custo += self.letras[6].distancia
91         custo += self.letras[7].distancia
92         custo += self.letras[8].distancia
93         custo += self.letras[9].distancia
94         print('\nA letra ',busca,'foi encontrada na posição 9 com um custo',custo )
95
96     elif busca == 'J':
97         custo += self.letras[1].distancia
98         custo += self.letras[2].distancia
99         custo += self.letras[3].distancia
100        custo += self.letras[4].distancia
101        custo += self.letras[5].distancia
102        print('\nA letra ',busca,'foi encontrada na posição 5 com um custo',custo )
103
104    elif busca == 'K':
105        custo += self.letras[1].distancia
106        custo += self.letras[2].distancia
107        custo += self.letras[3].distancia
108        custo += self.letras[4].distancia
109        print('\nA letra ',busca,'foi encontrada na posição 4 com um custo',custo )
110
111    elif busca == 'M':
112        custo += self.letras[1].distancia
113        custo += self.letras[2].distancia
114        custo += self.letras[3].distancia
115        custo += self.letras[4].distancia
116        custo += self.letras[5].distancia
117        custo += self.letras[6].distancia
118        custo += self.letras[7].distancia
119        print('\nA letra ',busca,'foi encontrada na posição 7 com um custo',custo )
120
121    elif busca == 'L':
122        custo += self.letras[1].distancia
123        custo += self.letras[2].distancia
124        custo += self.letras[3].distancia
125        custo += self.letras[4].distancia
126        custo += self.letras[5].distancia
127        custo += self.letras[6].distancia
128        print('\nA letra ',busca,'foi encontrada na posição 6 com um custo',custo )
129

```

```

130 elif busca == 'N':
131     custo += self.letras[1].distancia
132     custo += self.letras[2].distancia
133     custo += self.letras[3].distancia
134     custo += self.letras[4].distancia
135     custo += self.letras[5].distancia
136     custo += self.letras[6].distancia
137     custo += self.letras[7].distancia
138     custo += self.letras[8].distancia
139     custo += self.letras[9].distancia
140     custo += self.letras[10].distancia
141     print('\nA letra ',busca,'foi encontrada na posição 10 com um custo',custo )
142
143 elif busca == 'O':
144     custo += self.letras[1].distancia
145     custo += self.letras[2].distancia
146     custo += self.letras[3].distancia
147     custo += self.letras[4].distancia
148     custo += self.letras[5].distancia
149     custo += self.letras[6].distancia
150     custo += self.letras[7].distancia
151     custo += self.letras[8].distancia
152     custo += self.letras[9].distancia
153     custo += self.letras[10].distancia
154     custo += self.letras[11].distancia
155     print('\nA letra ',busca,'foi encontrada na posição 11 com um custo',custo )
156
157 elif busca == 'R':
158     custo += self.letras[1].distancia
159     custo += self.letras[2].distancia
160     custo += self.letras[3].distancia
161     custo += self.letras[4].distancia
162     custo += self.letras[5].distancia
163     custo += self.letras[6].distancia
164     custo += self.letras[7].distancia
165     custo += self.letras[8].distancia
166     custo += self.letras[9].distancia
167     custo += self.letras[10].distancia
168     custo += self.letras[11].distancia
169     custo += self.letras[12].distancia
170     custo += self.letras[13].distancia
171     custo += self.letras[14].distancia
172     print('\nA letra ',busca,'foi encontrada na posição 14 com um custo',custo )
173
174 elif busca == 'S':
175     custo += self.letras[1].distancia
176     custo += self.letras[2].distancia
177     custo += self.letras[3].distancia
178     custo += self.letras[4].distancia
179     custo += self.letras[5].distancia
180     custo += self.letras[6].distancia
181     custo += self.letras[7].distancia
182     custo += self.letras[8].distancia
183     custo += self.letras[9].distancia
184     custo += self.letras[10].distancia
185     custo += self.letras[11].distancia
186     custo += self.letras[12].distancia
187     custo += self.letras[13].distancia
188     custo += self.letras[14].distancia
189     custo += self.letras[15].distancia
190     print('\nA letra ',busca,'foi encontrada na posição 14 com um custo',custo )
191
192 else:
193     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.

```
1 from VetorOrdenado import VetorOrdenado
2 from Mapa import Mapa
3 mapa = Mapa()
4 vetor = VetorOrdenado(16)
5
6 def CriarMapaOrdenado():
7     vetor.inserir(mapa.A)
8     vetor.inserir(mapa.B)
9     vetor.inserir(mapa.C)
10    vetor.inserir(mapa.D)
11    vetor.inserir(mapa.K)
12    vetor.inserir(mapa.J)
13    vetor.inserir(mapa.L)
14    vetor.inserir(mapa.M)
15    vetor.inserir(mapa.G)
16    vetor.inserir(mapa.H)
17    vetor.inserir(mapa.N)
18    vetor.inserir(mapa.O)
19    vetor.inserir(mapa.E)
20    vetor.inserir(mapa.F)
21    vetor.inserir(mapa.R)
22    vetor.inserir(mapa.S)
23
24
25 CriarMapaOrdenado()
26 vetor.buscar()
27 vetor.mostrar()
28
```


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!