```
#2 questao letra a
def vertices_nao_adjacentes(self):
  lista = []
  for x in range(len(self.N)):
    for y in range(len(self.N)):
      aresta = '{}-{}'.format(self.N[x], self.N[y])
      if self.M[x][y] != '-' and self.M[x][y] == 0:
        lista.append(aresta)
  return lista

#2 questao letra b
def ha_laco(self):
  contador = 0
  for x in self.M:
    if x[contador] >= 1:
      return True
    contador += 1
  return False

#2 questao letra c
def ha_paralelas(self):
  for x in range(len(self.N)):
    for y in range(len(self.N)):
      if self.M[x][y] != '-' and self.M[x][y] >= 2:
        return True
  return False

#2 questao letra d
def grau(self, vertice):
  somador = 0
  for x in range(len(self.N)):
    if self.N[x] == vertice:
      for y in range(len(self.N)):
        if self.M[x][y] != '-':
          somador += self.M[x][y]
        elif self.M[y][x] != '-':
          somador += self.M[y][x]
  return somador

#2 questao letra e
def arestas_sobre_vertice(self, vertice):
  lista = []
  for x in range(len(self.N)):
    if self.N[x] == vertice:
      for y in range(len(self.N)):
        aresta = '{}-{}'.format(self.N[x], self.N[y])
        aresta1 = '{}-{}'.format(self.N[y], self.N[x])
        if self.M[x][y] != '-' and self.M[x][y] > 0:
```

```
            if self.M[x][y] != '-' and self.M[x][y] > 1:
                for z in range(self.M[x][y]):
                    lista.append(aresta)
            else:
                lista.append(aresta)
        elif self.M[y][x] != '-' and self.M[y][x] > 0:
            if self.M[y][x] != '-' and self.M[y][x] > 1:
                for z in range(self.M[y][x]):
                    lista.append(aresta1)
            else:
                lista.append(aresta1)
    return lista

#2 questao letra f
def eh_completo(self):
    for x in range(len(self.N)):
        for y in range(len(self.N)):
            if self.M[x][y] != '-' and x != y and self.M[x][y] == 0:
                return False
    return True

#2 questao letra g
def ha_ciclo(self):
    contador = 0
    listpostion=[]
    new_x=0
    vertices=self.N
    listvertices=[]
    for x in range(len(self.M)):
        for y in range(len(self.M[x])):
            y+=1
            try:

                posicao = self.M[x][y]
                new_vertice = str(vertices[x]) + "-" + str(vertices[y])

                if posicao!= "-" and posicao>=1 :
                    listpostion.append(posicao)
                    listvertices.append(new_vertice)
                    new_x=x
                    if new_x>=1:
                        new_x=0
                        while new_x != x:

                            new_position = self.M[new_x][y]
                            new_vertice = str(vertices[new_x]) + "-" + str(vertices[y])
                            if new_position>=1:
                                listpostion.append(new_position)
```

```
                    listvertices.append(new_vertice)
                    return listvertices
                new_x+=1
            else:
                listvertices.clear()

        except IndexError:
            pass
        x+=1
        contador += 1

  return False


#2 questao letra h
def comprimento_de_tamanho_n(self, n, vertice=None, visitado=[], caminho=[], bordas=None,
contador=0):
    if vertice == None:
        vertice = self.N[0]
    if bordas == None:
        bordas = []
        for i in range(len(self.N)):
            for j in range(len(self.N)):
                if self.M[i][j] != '-' and self.M[i][j] > 0:
                    for k in range(self.M[i][j]):
                        bordas.append(self.N[i] + "-" + self.N[j])
    if vertice in visitado:
        return False
    else:
        contador += 1
        if contador == n:
            return True
        visitado.append(vertice)
        bordas_adjacentes = self.arestas_sobre_vertice(vertice)
        for i in bordas_adjacentes:
            if i in bordas:
                bordas.remove(i)
            proximo = i.split("-")
            if proximo[0] == vertice:
                caminho.append(vertice + "-" + proximo[1])
                result = self.comprimento_de_tamanho_n(n, proximo[1], visitado, caminho, bordas,
contador)
                if result:
                    return True
                else:
                    caminho.pop()
                    contador -= 1
            else:
```

```
            if len(bordas_adjacentes) != 1:
                caminho.append(vertice + "-" + proximo[0])
                result = self.comprimento_de_tamanho_n(n, proximo[0], visitado, caminho,
bordas, contador)
                if result:
                    return True
                else:
                    caminho.pop()
                    contador -= 1
    return False

#2 questao letra i
def eh_conexo(self, vertice=None, visitado=[], caminho=[], bordas=None):
    if vertice is None:
        vertice = self.N[0]
    if bordas is None:
        bordas = []
        for i in range(len(self.N)):
            for j in range(len(self.N)):
                if self.M[i][j] != '-' and self.M[i][j] > 0:
                    for k in range(self.M[i][j]):
                        bordas.append(self.N[i] + "-" + self.N[j])
    if vertice in visitado:
        if len(visitado) == len(self.N):
            return True
        else:
            return False
    else:
        visitado.append(vertice)
        bordas_adjacentes = self.arestas_sobre_vertice(vertice)
        for i in bordas_adjacentes:
            proximo = i.split("-")
            if proximo[0] == vertice:
                caminho.append(vertice + "-" + proximo[1])
                if self.eh_conexo(proximo[1], visitado, caminho, bordas):
                    return True
                else:
                    caminho.pop()
            else:
                if len(bordas_adjacentes) != 1:
                    caminho.append(vertice + "-" + proximo[0])
                    if self.eh_conexo(proximo[0], visitado, caminho, bordas):
                        return True
                    else:
                        caminho.pop()
    return False
```