UNIVERSIDAD MAYOR, REAL Y PONTIFICIA DE SAN FRANCISCO XAVIER DE CHUQUISACA

FACULTAD DE TECNOLOGÍA

INGENIERIA DE SISTEMAS

INFORME LABORATORIO 2

ESTUDIANTE: ACHU GABRIEL ARIEL

CARRERA: INGENIERIA DE SISTEMAS

MATERIA: INTELIGENCIA ARTIFICIAL (SIS 420)

SUCRE – BOLIVIA

TAREA A REALIZAR

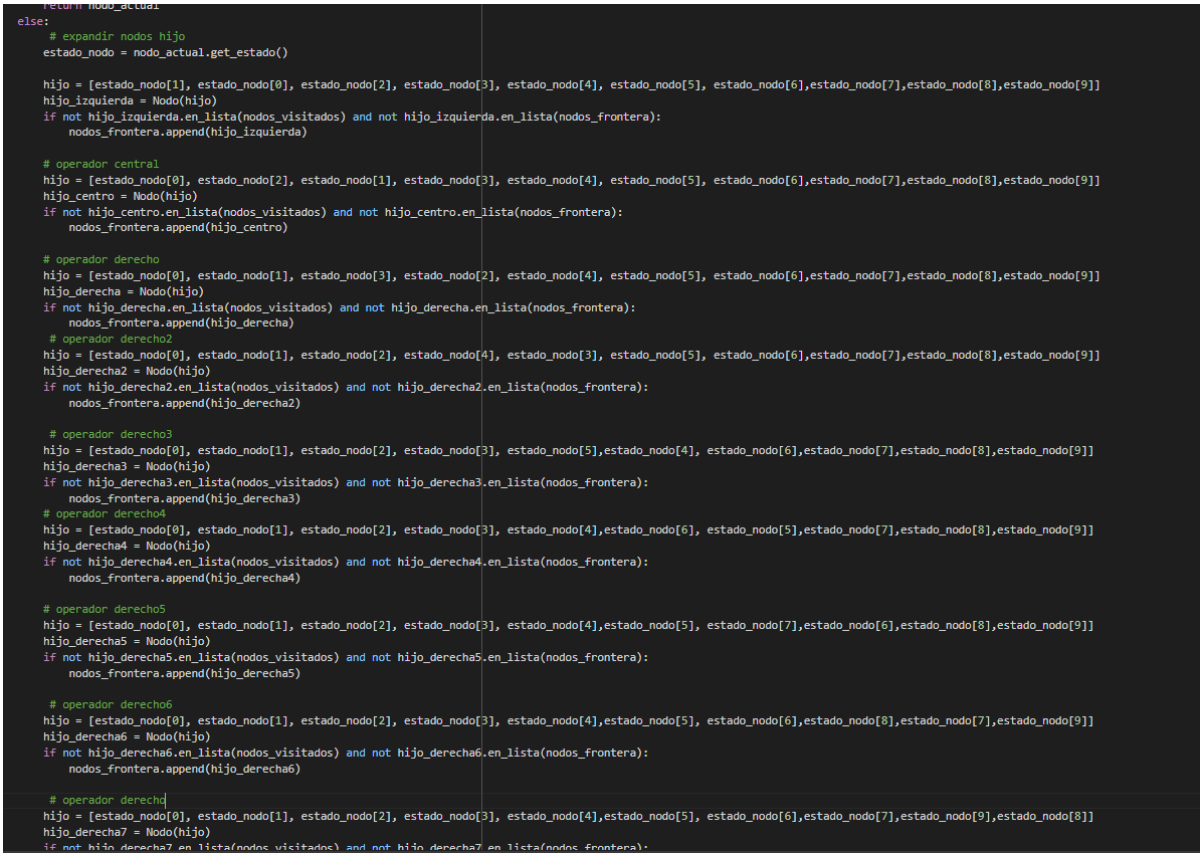
Elabore un programa para resolver un rompecabezas lineal de 10 dígitos en base al revisado en clase.

Debe informar que ocurre, en que tiempo lograr encontrar la solución, que dificultades identifico y cuales las soluciones o aportes.

Se logro resolver el problema de dos formas la primera forma se logro repitiendo los hijos y la segunda forma se logro optimizar la primera forma de repetir código y se utilizo 2 bubles while para poder llenar el hijo sin necesidad de repetir código

Primera forma:

El siguiente código está hecho para 10 elementos en un vector



```
    return nodo_actual
else:
     # expandir nodos hijo
    estado_nodo = nodo_actual.get_estado()

    hijo = [estado_nodo[1], estado_nodo[0], estado_nodo[2], estado_nodo[3], estado_nodo[4], estado_nodo[5], estado_nodo[6],estado_nodo[7],estado_nodo[8],estado_nodo[9]]
    hijo_izquierda = Nodo(hijo)
    if not hijo_izquierda.en_lista(nodos_visitados) and not hijo_izquierda.en_lista(nodos_frontera):
        nodos_frontera.append(hijo_izquierda)

    # operador central
    hijo = [estado_nodo[0], estado_nodo[2], estado_nodo[1], estado_nodo[3], estado_nodo[4], estado_nodo[5], estado_nodo[6],estado_nodo[7],estado_nodo[8],estado_nodo[9]]
    hijo_centro = Nodo(hijo)
    if not hijo_centro.en_lista(nodos_visitados) and not hijo_centro.en_lista(nodos_frontera):
        nodos_frontera.append(hijo_centro)

    # operador derecho
    hijo = [estado_nodo[0], estado_nodo[1], estado_nodo[3], estado_nodo[2], estado_nodo[4], estado_nodo[5], estado_nodo[6],estado_nodo[7],estado_nodo[8],estado_nodo[9]]
    hijo_derecha = Nodo(hijo)
    if not hijo_derecha.en_lista(nodos_visitados) and not hijo_derecha.en_lista(nodos_frontera):
        nodos_frontera.append(hijo_derecha)
     # operador derecho2
    hijo = [estado_nodo[0], estado_nodo[1], estado_nodo[2], estado_nodo[4], estado_nodo[3], estado_nodo[5], estado_nodo[6],estado_nodo[7],estado_nodo[8],estado_nodo[9]]
    hijo_derecha2 = Nodo(hijo)
    if not hijo_derecha2.en_lista(nodos_visitados) and not hijo_derecha2.en_lista(nodos_frontera):
        nodos_frontera.append(hijo_derecha2)

     # operador derecho3
    hijo = [estado_nodo[0], estado_nodo[1], estado_nodo[2], estado_nodo[3], estado_nodo[5],estado_nodo[4], estado_nodo[6],estado_nodo[7],estado_nodo[8],estado_nodo[9]]
    hijo_derecha3 = Nodo(hijo)
    if not hijo_derecha3.en_lista(nodos_visitados) and not hijo_derecha3.en_lista(nodos_frontera):
        nodos_frontera.append(hijo_derecha3)
    # operador derecho4
    hijo = [estado_nodo[0], estado_nodo[1], estado_nodo[2], estado_nodo[3], estado_nodo[4],estado_nodo[6], estado_nodo[5],estado_nodo[7],estado_nodo[8],estado_nodo[9]]
    hijo_derecha4 = Nodo(hijo)
    if not hijo_derecha4.en_lista(nodos_visitados) and not hijo_derecha4.en_lista(nodos_frontera):
        nodos_frontera.append(hijo_derecha4)

    # operador derecho5
    hijo = [estado_nodo[0], estado_nodo[1], estado_nodo[2], estado_nodo[3], estado_nodo[4],estado_nodo[5], estado_nodo[7],estado_nodo[6],estado_nodo[8],estado_nodo[9]]
    hijo_derecha5 = Nodo(hijo)
    if not hijo_derecha5.en_lista(nodos_visitados) and not hijo_derecha5.en_lista(nodos_frontera):
        nodos_frontera.append(hijo_derecha5)

     # operador derecho6
    hijo = [estado_nodo[0], estado_nodo[1], estado_nodo[2], estado_nodo[3], estado_nodo[4],estado_nodo[5], estado_nodo[6],estado_nodo[8],estado_nodo[7],estado_nodo[9]]
    hijo_derecha6 = Nodo(hijo)
    if not hijo_derecha6.en_lista(nodos_visitados) and not hijo_derecha6.en_lista(nodos_frontera):
        nodos_frontera.append(hijo_derecha6)

     # operador derecho
    hijo = [estado_nodo[0], estado_nodo[1], estado_nodo[2], estado_nodo[3], estado_nodo[4],estado_nodo[5], estado_nodo[6],estado_nodo[7],estado_nodo[9],estado_nodo[8]]
    hijo_derecha7 = Nodo(hijo)
    if not hijo_derecha7.en_lista(nodos_visitados) and not hijo_derecha7.en_lista(nodos_frontera):
```

Segunda forma:

El siguiente código está hecho para **n** elementos en un vector

```
    # expandir nodos hijo
    estado_nodo = nodo_actual.get_estado()
    cont=0
    i=0
    hijo_centro=[]
    while cont<=len(estado_nodo)-2:
      vector2=[]
      j=0
      while j<=len(estado_nodo)-1:
        if j==i and j<=len(estado_nodo)-2:
          vector2.append(estado_nodo[j+1])
          vector2.append(estado_nodo[j])
          j=j+2
        else:
          if i==len(estado_nodo)-1:
            vector2.append(estado_nodo[j-1])
            vector2.append(estado_nodo[j])
          else:
            vector2.append(estado_nodo[j])
            j=j+1
      i=i+1
      hijo_centro.append(Nodo(vector2))

      if not hijo_centro[cont].en_lista(nodos_visitados) and not hijo_centro[cont].en_lista(nodos_frontera):
        nodos_frontera.append(hijo_centro[cont])
      cont=cont+1

    nodo_actual.set_hijo(hijo_centro)
```

Problemas detectados

Tiempo: Las 2 formas se probaron el mismo vector [4, 2, 3, 1, 5, 6, 8, 7, 9,10]

La primera forma tardo 15.50s en encontrar la solución.

La segunda forma tardo 14.29s en encontrar la solución.

Limite:

La primera forma solo puede encontrar la solución para 10 elementos ni más ni menos.

La segunda forma puede encontrar la solución para N elementos.

Menos Código:

La primera forma para encontrar una solución de n elementos se tiene que ir repitiendo líneas de código.

La segunda forma puede encontrar una solución con la misma cantidad de código(optimizado).