LABORATORIO 03 (GRUPO 1 Y GRUPO 2)

Universitarios :

- Ceron Beimar Miguel -Ing. de Sistemas
- Martinez Rodriguez Milton -Ciencias de la computación

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.

- En búsqueda primero en amplitud es más adecuado para buscar vértices más cercanos a la fuente dada, pero no lejanos
- Considera primero a todos los vecinos y, por lo tanto, no es adecuado para árboles de toma de decisiones utilizados en juegos o rompecabezas.
- Es óptimo para encontrar el camino más corto.
- Cuando el objetivo está cerca de la fuente, BPA funciona mejor.
- Se recomienda usar búsqueda primero en profundidad, que utiliza la estructura de datos Stack / LIFO
- Tambien es recomendable usar Quicksort en lugar de .pop() que son algoritmos de clasificación más eficientes de ordenación rápida

BPA-Busqueda por Amplitud

In [ ]:

```python
# -*- coding: utf-8 -*-
"""
Created on Sun Oct 16 18:19:39 2022

@author: hp
"""

# datos = estado
class Nodo:
    def __init__(self, datos, hijo=None):
        self.datos = datos
        self.hijos = []
        self.padre = None
        self.costo = None
        self.set_hijo(hijo)

    def set_hijo(self, hijo):
        if (hijo is not None):
            self.hijos.append(hijo)
            if self.hijos is not None:
                for h in self.hijos:
                    h.padre = self

    def get_hijos(self):
        return self.hijos

    def set_padre(self, padre):
        self.padre = padre

    def get_padre(self):
        return self.padre

    def set_datos(self, datos):
        self.datos = datos

    def get_datos(self):
        return self.datos

    def set_costo(self, costo):
        self.costo = costo

    def get_costo(self):
        return self.costo

    def equal(self, nodo):
        if self.get_datos() == nodo.get_datos():
            return True
        else:
            return False

    def en_lista(self, lista_nodos):
        enlistado = False
        for n in lista_nodos:
            if self.equal(n):
                enlistado = True
        return enlistado

    def __str__(self):
        return str(self.get_datos())
```

```
60
61
62
63  def bpa(estado_inicio, estado_solucion):
64      resuelto = False
65      nodos_visitados = []
66      nodos_frontera = []
67      nodo_inicio = Nodo(estado_inicio)
68      nodos_frontera.append(nodo_inicio)
69
70      while resuelto == False and len(nodos_frontera) != 0:
71          #nodo_actual = nodos_frontera.pop(0)  # FIFO - cola
72          nodo_actual = nodos_frontera.pop() #LIFO - pila
73          nodos_visitados.append(nodo_actual)
74          if nodo_actual.get_datos() == estado_solucion:
75              resuelto = True
76              return nodo_actual
77          else:
78              for i in range(len(estado_inicio) - 1):
79                  for j in range(len(estado_inicio) - 1):
80                      hijo_datos = nodo_actual.get_datos().copy()
81                      temp = hijo_datos[j]
82                      hijo_datos[j] = hijo_datos[j + 1]
83                      hijo_datos[j + 1] = temp
84                      hijo = Nodo(hijo_datos)
85                      # print(f'hijo {j+1}',hijo_datos)
86                      if not hijo.en_lista(nodos_visitados) and not hijo.en_lista(nodos_f
87                          nodo_actual.set_hijo(hijo)
88                          nodos_frontera.append(hijo)
89
90
```

In [7]:

```
1
2  import time
3
4  if __name__ == "__main__":
5      estado_inicial = [6,5,4,3, 2, 1]
6      solucion = [1,2,3,4,5,6]
7      start = time.time()
8      nodo_solucion = bpa(estado_inicial, solucion)
9      end =time.time()
10
11     def convert(seconds):
12         return time.strftime("%H:%M:%S", time.gmtime(end-start))
13     print('Tiempo de ejecucion Horas:Minutos:Seg : ',convert(end-start), '.','\n')
14     # mostrar resultado
15     resultado = []
16     nodo_actual = nodo_solucion
17     while nodo_actual.get_padre() is not None:
18         resultado.append(nodo_actual.get_datos())
19         nodo_actual = nodo_actual.get_padre()
20
21     resultado.append(estado_inicial)
22     resultado.reverse()
23     print("*****"*10)
24     #print('\n',resultado)
25     print('Movimientos : \n')
26     c=0
27     s=0
28     for i in range(len(resultado)):
29       print(resultado[i])
30       c+= 1
31       print("Nro. de Movimiento :", c)
32     s=c
33
34     print("Total :", s)
```

```
Tiempo de ejecucion Horas:Minutos:Seg :  00:00:03 .

**************************************************
Movimientos :

[6, 5, 4, 3, 2, 1]
Nro. de Movimiento : 1
[5, 6, 4, 3, 2, 1]
Nro. de Movimiento : 2
[5, 4, 6, 3, 2, 1]
Nro. de Movimiento : 3
[4, 5, 6, 3, 2, 1]
Nro. de Movimiento : 4
[4, 5, 3, 6, 2, 1]
Nro. de Movimiento : 5
[4, 3, 5, 6, 2, 1]
Nro. de Movimiento : 6
[3, 4, 5, 6, 2, 1]
Nro. de Movimiento : 7
[3, 4, 5, 2, 6, 1]
Nro. de Movimiento : 8
[3, 4, 2, 5, 6, 1]
Nro. de Movimiento : 9
```

```
[3, 2, 4, 5, 6, 1]
Nro. de Movimiento : 10
[2, 3, 4, 5, 6, 1]
Nro. de Movimiento : 11
[2, 3, 4, 5, 1, 6]
Nro. de Movimiento : 12
[2, 3, 4, 1, 5, 6]
Nro. de Movimiento : 13
[2, 3, 1, 4, 5, 6]
Nro. de Movimiento : 14
[2, 1, 3, 4, 5, 6]
Nro. de Movimiento : 15
[1, 2, 3, 4, 5, 6]
Nro. de Movimiento : 16
Total : 16
```

In [ ]:

```
1
```

In [8]:

```
import time

if __name__ == "__main__":
    estado_inicial = [7,6,5,4,3, 2, 1]
    solucion = [1,2,3,4,5,6,7]
    start = time.time()
    nodo_solucion = bpa(estado_inicial, solucion)
    end =time.time()

    def convert(seconds):
        return time.strftime("%H:%M:%S", time.gmtime(end-start))
    print('Tiempo de ejecucion Horas:Minutos:Seg : ',convert(end-start), '.','\n')
    # mostrar resultado
    resultado = []
    nodo_actual = nodo_solucion
    while nodo_actual.get_padre() is not None:
        resultado.append(nodo_actual.get_datos())
        nodo_actual = nodo_actual.get_padre()

    resultado.append(estado_inicial)
    resultado.reverse()
    print("*****"*10)
    #print('\n',resultado)
    print('Movimientos : \n')
    c=0
    s=0
    for i in range(len(resultado)):
      print(resultado[i])
      c+= 1
      print("Nro. de Movimiento :", c)
    s=c

    print("Total :", s)
```

```
Tiempo de ejecucion Horas:Minutos:Seg :  00:04:46 .

**************************************************
Movimientos :

[7, 6, 5, 4, 3, 2, 1]
Nro. de Movimiento : 1
[6, 7, 5, 4, 3, 2, 1]
Nro. de Movimiento : 2
[6, 5, 7, 4, 3, 2, 1]
Nro. de Movimiento : 3
[5, 6, 7, 4, 3, 2, 1]
Nro. de Movimiento : 4
[5, 6, 4, 7, 3, 2, 1]
Nro. de Movimiento : 5
[5, 4, 6, 7, 3, 2, 1]
Nro. de Movimiento : 6
[4, 5, 6, 7, 3, 2, 1]
Nro. de Movimiento : 7
[4, 5, 6, 3, 7, 2, 1]
Nro. de Movimiento : 8
[4, 5, 3, 6, 7, 2, 1]
Nro. de Movimiento : 9
[4, 3, 5, 6, 7, 2, 1]
```

```
Nro. de Movimiento : 10
[3, 4, 5, 6, 7, 2, 1]
Nro. de Movimiento : 11
[3, 4, 5, 6, 2, 7, 1]
Nro. de Movimiento : 12
[3, 4, 5, 2, 6, 7, 1]
Nro. de Movimiento : 13
[3, 4, 2, 5, 6, 7, 1]
Nro. de Movimiento : 14
[3, 2, 4, 5, 6, 7, 1]
Nro. de Movimiento : 15
[2, 3, 4, 5, 6, 7, 1]
Nro. de Movimiento : 16
[2, 3, 4, 5, 6, 1, 7]
Nro. de Movimiento : 17
[2, 3, 4, 5, 1, 6, 7]
Nro. de Movimiento : 18
[2, 3, 4, 1, 5, 6, 7]
Nro. de Movimiento : 19
[2, 3, 1, 4, 5, 6, 7]
Nro. de Movimiento : 20
[2, 1, 3, 4, 5, 6, 7]
Nro. de Movimiento : 21
[1, 2, 3, 4, 5, 6, 7]
Nro. de Movimiento : 22
Total : 22
```

Algoritmo QuickSort

In [28]:

```python
# importing the module
from datetime import datetime
def partition(array, low, high):


    pivot = array[high]

    i = low - 1

    for j in range(low, high):
        if array[j] <= pivot:

            i = i + 1

            (array[i], array[j]) = (array[j], array[i])


    (array[i + 1], array[high]) = (array[high], array[i + 1])

    return i + 1


def quickSort(array, low, high):
    if low < high:

        pi = partition(array, low, high)
        quickSort(array, low, pi - 1)
        quickSort(array, pi + 1, high)


data = [10,9,8,7,6,5,4,3,2,1]

print("Matriz sin ordenar :")
print(data)
start = datetime.now()
size = len(data)
quickSort(data, 0, size - 1)
print('Matriz ordenada en orden ascendente:')
print(data)
end = datetime.now()
print('formato HH: MM: SS: MiliSeg',end-start)
```

```
Matriz sin ordenar :
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
Matriz ordenada en orden ascendente:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
formato HH: MM: SS: MiliSeg 0:00:00.001000
```

In [ ]:

```
1
```

In [ ]:

```
1
```

In [ ]:

```
1
```