Actividad Grupal

Resolución del problema del puzzle-8 mediante búsqueda heurística A*

Integrantes:

- **Godoy Bonillo Giocrisrai**
- **Ordoñez Marín, César Augusto**
- **Álvarez González, Fátima del Rosario**
- **Castillo Quenaya, Luis Miguel**
- **Santana Chavez, Erick Eduardo**

Con esta actividad vas a conseguir implementar la estrategia de búsqueda heurística A* para la resolución del problema del puzzle-8.

In [13]:
```
import numpy as np
```

In [14]:
```
class Nodos:
    def __init__(self,datos,nivel,fvalor,movimiento):
#Inicializa el nodo con los datos, nivel, movimientos y el valor calculado
        self.datos = datos
        self.nivel = nivel
        self.fvalor = fvalor
        self.movimiento = movimiento
#Genera nodos secundarios a partir del nodo dado, moviendo el espacio en blanco en las cuatro direcciones {
arriba, abajo, izquierda, derecha} """
    def crear_hijos(self):
        # Se busca la posición del espacio en blanco.
        x,y = np.where(self.datos == "_")
        #lista contiene valores de posición para mover el espacio en blanco
        lista = np.array([[x,y-1],[x,y+1],[x-1,y],[x+1,y]])
        hijos = np.array([])
        for i in lista:
            hijo, movimiento = self.resolver(self.datos,x,y,i[0],i[1])
        #Se verifica las coordenadas de la lista existe y se crea un nodo hijo.
            if hijo is not None:
                nodos_hijos = Nodos(hijo, self.nivel + 1, 0, movimiento)
                hijos = np.append(hijos, nodos_hijos)
        return hijos

    def resolver(self,puzzle,x1,y1,x2,y2):
        x1 = x1[0];y1 = y1[0];x2 = x2[0];y2 = y2[0];
        # Mueva el espacio en blanco dentro de los límites de la dirección dada
        if x2 >= 0 and x2 < len(self.datos) and y2 >= 0 and y2 < len(self.datos):
            temp_puzzle = np.copy(puzzle) #Se crea una copia temporal del nodo
            temp_puzzle[x1][y1] = temp_puzzle[x2][y2] # Asigna el nuevo valor
            temp_puzzle[x2][y2] = "_" # Asignar a la posicion cambiada el valor "_"
            # guarda el dato a reemplazar temporalmente
            if (y2 - y1) == -1:
                movimiento = temp_puzzle[x1][y1] + " a la derecha"
            if (y2 - y1) == 1:
                movimiento = temp_puzzle[x1][y1] + " a la izquierda"
            if (x2 - x1) == -1:
                movimiento = temp_puzzle[x1][y1] + " hacia abajo"
            if (x2 - x1) == 1:
                movimiento = temp_puzzle[x1][y1] + " hacia arriba"
            temp_puzzle
            return temp_puzzle, movimiento # Retorna la nueva matriz hijo y su movimiento
        else:
            return None, None
```

In [15]:
```
#Clase puzzle que recibe las parametrización inicial
class Puzzle_8:

    puzzle_inicial = np.array([])
    puzzle_final = np.array([])

    def __init__(self, dimension): # Inicia el tamaño del puzzle por el tamaño especificado
        self.n = dimension
        self.n_abierto = []
        self.n_cerrado = np.array([])

    def imprimir_metricas(self,estado_inicial,estado_final, indent=0):
        h_val = self.h(estado_inicial.datos,estado_final)
        f_val = h_val+estado_inicial.nivel
        print(f'f(n): {f_val}, g(n):{estado_inicial.nivel} , h(n): {h_val}, M: {estado_inicial.movimiento}'
)

    def f(self,estado_inicial,estado_final): #Función heurística que cálcula f(x)=h(x)+g(x)
        return self.h(estado_inicial.datos,estado_final)+estado_inicial.nivel

    def h(self,estado_inicial,estado_final): #Cálcula la diferencia entre los puzzle
        heuristica = (estado_inicial != estado_final).sum()
        if(heuristica < 0):
            heuristica = 0
        return heuristica

    def imprimir_matriz(self, matriz, msg='', metricas=True, isnodo=True):#Función para salida de datos
        metricas_string = ''
        indent_string = ''
        if isnodo:
            nivel = matriz.nivel
            datos = matriz.datos
        else:
            datos = matriz
        if msg != '':
            print(indent_string + msg)
        if metricas and isnodo:
            self.imprimir_metricas(matriz, self.puzzle_final, nivel)
        for i in datos:
            aux = ""
            for j in i:
                aux = aux+j+" "
            print(indent_string + aux)

    def procesar_puzzle(self): # Acepta el estado de Puzzle de inicio y objetivo
        estado_inicial = self.puzzle_inicial
        self.imprimir_matriz(estado_inicial, 'Matriz Inicial', False, False)
        estado_final = self.puzzle_final
        self.imprimir_matriz(estado_final, 'Matriz Final', False, False)
        estado_inicial = Nodos(estado_inicial,0,0,"")
        estado_inicial.fvalor = self.f(estado_inicial,estado_final)
        self.n_abierto.append(estado_inicial) # Agregar a la cola del nodo de inicio en la lista abierta
        iteracion = 1
        while True:
            cur = self.n_abierto[0] #Nos quedamos siempre con el primer elemento de la lista de hijos.
            # Si la diferencia entre el nodo actual y el objetivo es 0, hemos alcanzado el nodo objetivo
            if(self.h(cur.datos,estado_final) == 0):
                self.imprimir_matriz(cur, 'Resultado')
                break
            contador = 1
            print(f"\n---- ITERACION {iteracion}---:")
            print('----Nodos Abiertos---:')
            for i in self.n_abierto:
                self.imprimir_matriz(i, '')

            for i in cur.crear_hijos():
                i.fvalor = self.f(i, estado_final)
                self.n_abierto.append(i) # Agrega nuevo hijo a la lista de nodos abiertos
                contador += 1

            self.imprimir_matriz(cur, '---- Nodo Expandido ----')

            if len(self.n_cerrado) > 0:
                print('----Nodos Cerrados---:')
                for i in self.n_cerrado:
                    self.imprimir_matriz(i, '')

            self.n_cerrado = np.append(self.n_cerrado, cur) # Listada de nodos Cerrados
            del self.n_abierto[0]
            #Orderna los hijos por el valor de F más bajo.
            self.n_abierto.sort(key = lambda x:x.fvalor,reverse=False)
            iteracion += 1
```

Ejecución del llamado de la clase para resolver el Algorimo A* para el puzzle 8

In [16]:
```
puzzle = Puzzle_8(3) # Se llama a la clase definiendo el tamaño de la matriz a resolver
# Se da el valor inicial del puzzle
puzzle.puzzle_inicial = np.array([["2", "8", "3"], ["1", "6", "4"], ["7", "_", "5"]])
# Se da el valor objetivo del puzzle para la correspondiente ejecución
puzzle.puzzle_final = np.array([["1", "2", "3"], ["8", "_", "4"], ["7", "6", "5"]])
puzzle.procesar_puzzle() # Ejecución del puzzle
```

```
Matriz Inicial
2 8 3
1 6 4
7 _ 5
Matriz Final
1 2 3
8 _ 4
7 6 5

---- ITERACION 1---:
----Nodos Abiertos---:
f(n): 5, g(n):0 , h(n): 5, M:
2 8 3
1 6 4
7 _ 5
---- Nodo Expandido ----
f(n): 5, g(n):0 , h(n): 5, M:
2 8 3
1 6 4
7 _ 5

---- ITERACION 2---:
----Nodos Abiertos---:
f(n): 4, g(n):1 , h(n): 3, M: 6 hacia abajo
2 8 3
1 _ 4
7 6 5
f(n): 7, g(n):1 , h(n): 6, M: 7 a la derecha
2 8 3
1 6 4
_ 7 5
f(n): 7, g(n):1 , h(n): 6, M: 5 a la izquierda
2 8 3
1 6 4
7 5 _
---- Nodo Expandido ----
f(n): 4, g(n):1 , h(n): 3, M: 6 hacia abajo
2 8 3
1 _ 4
7 6 5
----Nodos Cerrados---:
f(n): 5, g(n):0 , h(n): 5, M:
2 8 3
1 6 4
7 _ 5

---- ITERACION 3---:
----Nodos Abiertos---:
f(n): 6, g(n):2 , h(n): 4, M: 1 a la derecha
2 8 3
_ 1 4
7 6 5
f(n): 6, g(n):2 , h(n): 4, M: 8 hacia abajo
2 _ 3
1 8 4
7 6 5
f(n): 7, g(n):1 , h(n): 6, M: 7 a la derecha
2 8 3
1 6 4
_ 7 5
f(n): 7, g(n):1 , h(n): 6, M: 5 a la izquierda
2 8 3
1 6 4
7 5 _
f(n): 7, g(n):2 , h(n): 5, M: 4 a la izquierda
2 8 3
1 4 _
7 6 5
f(n): 7, g(n):2 , h(n): 5, M: 6 hacia arriba
2 8 3
1 6 4
7 _ 5
---- Nodo Expandido ----
f(n): 6, g(n):2 , h(n): 4, M: 1 a la derecha
2 8 3
_ 1 4
7 6 5
----Nodos Cerrados---:
f(n): 5, g(n):0 , h(n): 5, M:
2 8 3
1 6 4
7 _ 5
f(n): 4, g(n):1 , h(n): 3, M: 6 hacia abajo
```

```
2 8 3
1 _ 4
7 6 5

---- ITERACION 4---:
----Nodos Abiertos---:
f(n): 6, g(n):2 , h(n): 4, M: 8 hacia abajo
2 _ 3
1 8 4
7 6 5
f(n): 6, g(n):3 , h(n): 3, M: 1 a la izquierda
2 8 3
1 _ 4
7 6 5
f(n): 7, g(n):1 , h(n): 6, M: 7 a la derecha
2 8 3
1 6 4
_ 7 5
f(n): 7, g(n):1 , h(n): 6, M: 5 a la izquierda
2 8 3
1 6 4
7 5 _
f(n): 7, g(n):2 , h(n): 5, M: 4 a la izquierda
2 8 3
1 4 _
7 6 5
f(n): 7, g(n):2 , h(n): 5, M: 6 hacia arriba
2 8 3
1 6 4
7 _ 5
f(n): 7, g(n):3 , h(n): 4, M: 2 hacia abajo
_ 8 3
2 1 4
7 6 5
f(n): 8, g(n):3 , h(n): 5, M: 7 hacia arriba
2 8 3
7 1 4
_ 6 5
---- Nodo Expandido ----
f(n): 6, g(n):2 , h(n): 4, M: 8 hacia abajo
2 _ 3
1 8 4
7 6 5
----Nodos Cerrados---:
f(n): 5, g(n):0 , h(n): 5, M:
2 8 3
1 6 4
7 _ 5
f(n): 4, g(n):1 , h(n): 3, M: 6 hacia abajo
2 8 3
1 _ 4
7 6 5
f(n): 6, g(n):2 , h(n): 4, M: 1 a la derecha
2 8 3
_ 1 4
7 6 5

---- ITERACION 5---:
----Nodos Abiertos---:
f(n): 6, g(n):3 , h(n): 3, M: 1 a la izquierda
2 8 3
1 _ 4
7 6 5
f(n): 6, g(n):3 , h(n): 3, M: 2 a la derecha
_ 2 3
1 8 4
7 6 5
f(n): 6, g(n):3 , h(n): 3, M: 8 hacia arriba
2 8 3
1 _ 4
7 6 5
f(n): 7, g(n):1 , h(n): 6, M: 7 a la derecha
2 8 3
1 6 4
_ 7 5
f(n): 7, g(n):1 , h(n): 6, M: 5 a la izquierda
2 8 3
1 6 4
7 5 _
f(n): 7, g(n):2 , h(n): 5, M: 4 a la izquierda
2 8 3
1 4 _
7 6 5
f(n): 7, g(n):2 , h(n): 5, M: 6 hacia arriba
2 8 3
1 6 4
7 _ 5
f(n): 7, g(n):3 , h(n): 4, M: 2 hacia abajo
_ 8 3
2 1 4
7 6 5
```

```
f(n): 8, g(n):3 , h(n): 5, M: 7 hacia arriba
2 8 3
7 1 4
_ 6 5
f(n): 8, g(n):3 , h(n): 5, M: 3 a la izquierda
2 3 _
1 8 4
7 6 5
---- Nodo Expandido ----
f(n): 6, g(n):3 , h(n): 3, M: 1 a la izquierda
2 8 3
1 _ 4
7 6 5
----Nodos Cerrados---:
f(n): 5, g(n):0 , h(n): 5, M:
2 8 3
1 6 4
7 _ 5
f(n): 4, g(n):1 , h(n): 3, M: 6 hacia abajo
2 8 3
1 _ 4
7 6 5
f(n): 6, g(n):2 , h(n): 4, M: 1 a la derecha
2 8 3
_ 1 4
7 6 5
f(n): 6, g(n):2 , h(n): 4, M: 8 hacia abajo
2 _ 3
1 8 4
7 6 5
---- ITERACION 6---:
----Nodos Abiertos---:
f(n): 6, g(n):3 , h(n): 3, M: 2 a la derecha
_ 2 3
1 8 4
7 6 5
f(n): 6, g(n):3 , h(n): 3, M: 8 hacia arriba
2 8 3
1 _ 4
7 6 5
f(n): 7, g(n):1 , h(n): 6, M: 7 a la derecha
2 8 3
1 6 4
_ 7 5
f(n): 7, g(n):1 , h(n): 6, M: 5 a la izquierda
2 8 3
1 6 4
7 5 _
f(n): 7, g(n):2 , h(n): 5, M: 4 a la izquierda
2 8 3
1 4 _
7 6 5
f(n): 7, g(n):2 , h(n): 5, M: 6 hacia arriba
2 8 3
1 6 4
7 _ 5
f(n): 7, g(n):3 , h(n): 4, M: 2 hacia abajo
_ 8 3
2 1 4
7 6 5
f(n): 8, g(n):3 , h(n): 5, M: 7 hacia arriba
2 8 3
7 1 4
_ 6 5
f(n): 8, g(n):3 , h(n): 5, M: 3 a la izquierda
2 3 _
1 8 4
7 6 5
f(n): 8, g(n):4 , h(n): 4, M: 1 a la derecha
2 8 3
_ 1 4
7 6 5
f(n): 8, g(n):4 , h(n): 4, M: 8 hacia abajo
2 _ 3
1 8 4
7 6 5
f(n): 9, g(n):4 , h(n): 5, M: 4 a la izquierda
2 8 3
1 4 _
7 6 5
f(n): 9, g(n):4 , h(n): 5, M: 6 hacia arriba
2 8 3
1 6 4
7 _ 5
---- Nodo Expandido ----
f(n): 6, g(n):3 , h(n): 3, M: 2 a la derecha
_ 2 3
1 8 4
7 6 5
----Nodos Cerrados---:
```

```
f(n): 5, g(n):0 , h(n): 5, M:
2 8 3
1 6 4
7 _ 5
f(n): 4, g(n):1 , h(n): 3, M: 6 hacia abajo
2 8 3
1 _ 4
7 6 5
f(n): 6, g(n):2 , h(n): 4, M: 1 a la derecha
2 8 3
_ 1 4
7 6 5
f(n): 6, g(n):2 , h(n): 4, M: 8 hacia abajo
2 _ 3
1 8 4
7 6 5
f(n): 6, g(n):3 , h(n): 3, M: 1 a la izquierda
2 8 3
1 _ 4
7 6 5

---- ITERACION 7---:
----Nodos Abiertos---:
f(n): 6, g(n):3 , h(n): 3, M: 8 hacia arriba
2 8 3
1 _ 4
7 6 5
f(n): 6, g(n):4 , h(n): 2, M: 1 hacia arriba
1 2 3
_ 8 4
7 6 5
f(n): 7, g(n):1 , h(n): 6, M: 7 a la derecha
2 8 3
1 6 4
_ 7 5
f(n): 7, g(n):1 , h(n): 6, M: 5 a la izquierda
2 8 3
1 6 4
7 5 _
f(n): 7, g(n):2 , h(n): 5, M: 4 a la izquierda
2 8 3
1 4 _
7 6 5
f(n): 7, g(n):2 , h(n): 5, M: 6 hacia arriba
2 8 3
1 6 4
7 _ 5
f(n): 7, g(n):3 , h(n): 4, M: 2 hacia abajo
_ 8 3
2 1 4
7 6 5
f(n): 8, g(n):3 , h(n): 5, M: 7 hacia arriba
2 8 3
7 1 4
_ 6 5
f(n): 8, g(n):3 , h(n): 5, M: 3 a la izquierda
2 3 _
1 8 4
7 6 5
f(n): 8, g(n):4 , h(n): 4, M: 1 a la derecha
2 8 3
_ 1 4
7 6 5
f(n): 8, g(n):4 , h(n): 4, M: 8 hacia abajo
2 _ 3
1 8 4
7 6 5
f(n): 8, g(n):4 , h(n): 4, M: 2 a la izquierda
2 _ 3
1 8 4
7 6 5
f(n): 9, g(n):4 , h(n): 5, M: 4 a la izquierda
2 8 3
1 4 _
7 6 5
f(n): 9, g(n):4 , h(n): 5, M: 6 hacia arriba
2 8 3
1 6 4
7 _ 5
---- Nodo Expandido ----
f(n): 6, g(n):3 , h(n): 3, M: 8 hacia arriba
2 8 3
1 _ 4
7 6 5
----Nodos Cerrados---:
f(n): 5, g(n):0 , h(n): 5, M:
2 8 3
1 6 4
7 _ 5
f(n): 4, g(n):1 , h(n): 3, M: 6 hacia abajo
2 8 3
```

```
1 _ 4
7 6 5
f(n): 6, g(n):2 , h(n): 4, M: 1 a la derecha
2 8 3
_ 1 4
7 6 5
f(n): 6, g(n):2 , h(n): 4, M: 8 hacia abajo
2 _ 3
1 8 4
7 6 5
f(n): 6, g(n):3 , h(n): 3, M: 1 a la izquierda
2 8 3
1 _ 4
7 6 5
f(n): 6, g(n):3 , h(n): 3, M: 2 a la derecha
_ 2 3
1 8 4
7 6 5

---- ITERACION 8---:
----Nodos Abiertos---:
f(n): 6, g(n):4 , h(n): 2, M: 1 hacia arriba
1 2 3
_ 8 4
7 6 5
f(n): 7, g(n):1 , h(n): 6, M: 7 a la derecha
2 8 3
1 6 4
_ 7 5
f(n): 7, g(n):1 , h(n): 6, M: 5 a la izquierda
2 8 3
1 6 4
7 5 _
f(n): 7, g(n):2 , h(n): 5, M: 4 a la izquierda
2 8 3
1 4 _
7 6 5
f(n): 7, g(n):2 , h(n): 5, M: 6 hacia arriba
2 8 3
1 6 4
7 _ 5
f(n): 7, g(n):3 , h(n): 4, M: 2 hacia abajo
_ 8 3
2 1 4
7 6 5
f(n): 8, g(n):3 , h(n): 5, M: 7 hacia arriba
2 8 3
7 1 4
_ 6 5
f(n): 8, g(n):3 , h(n): 5, M: 3 a la izquierda
2 3 _
1 8 4
7 6 5
f(n): 8, g(n):4 , h(n): 4, M: 1 a la derecha
2 8 3
_ 1 4
7 6 5
f(n): 8, g(n):4 , h(n): 4, M: 8 hacia abajo
2 _ 3
1 8 4
7 6 5
f(n): 8, g(n):4 , h(n): 4, M: 2 a la izquierda
2 _ 3
1 8 4
7 6 5
f(n): 8, g(n):4 , h(n): 4, M: 1 a la derecha
2 8 3
_ 1 4
7 6 5
f(n): 8, g(n):4 , h(n): 4, M: 8 hacia abajo
2 _ 3
1 8 4
7 6 5
f(n): 9, g(n):4 , h(n): 5, M: 4 a la izquierda
2 8 3
1 4 _
7 6 5
f(n): 9, g(n):4 , h(n): 5, M: 6 hacia arriba
2 8 3
1 6 4
7 _ 5
f(n): 9, g(n):4 , h(n): 5, M: 4 a la izquierda
2 8 3
1 4 _
7 6 5
f(n): 9, g(n):4 , h(n): 5, M: 6 hacia arriba
2 8 3
1 6 4
7 _ 5
---- Nodo Expandido ----
f(n): 6, g(n):4 , h(n): 2, M: 1 hacia arriba
```

```
1 2 3
_ 8 4
7 6 5
----Nodos Cerrados---:
f(n): 5, g(n):0 , h(n): 5, M:
2 8 3
1 6 4
7 _ 5
f(n): 4, g(n):1 , h(n): 3, M: 6 hacia abajo
2 8 3
1 _ 4
7 6 5
f(n): 6, g(n):2 , h(n): 4, M: 1 a la derecha
2 8 3
_ 1 4
7 6 5
f(n): 6, g(n):2 , h(n): 4, M: 8 hacia abajo
2 _ 3
1 8 4
7 6 5
f(n): 6, g(n):3 , h(n): 3, M: 1 a la izquierda
2 8 3
1 _ 4
7 6 5
f(n): 6, g(n):3 , h(n): 3, M: 2 a la derecha
_ 2 3
1 8 4
7 6 5
f(n): 6, g(n):3 , h(n): 3, M: 8 hacia arriba
2 8 3
1 _ 4
7 6 5
Resultado
f(n): 5, g(n):5 , h(n): 0, M: 8 a la izquierda
1 2 3
8 _ 4
7 6 5
```

In [ ]:

	Sí	No	A veces
Todos los miembros se han integrado al trabajo del grupo	X		
Todos los miembros participan activamente	X		
Todos los miembros respetan otras ideas aportadas	X		
Todos los miembros participan en la elaboración del informe	X		
Me he preocupado por realizar un trabajo cooperativo con mis compañeros	X		
Señala si consideras que algún aspecto del trabajo en grupo no ha sido adecuado	x		