ESCUELA NACIONAL DE ESTUDIOS SUPERIORES
UNIDAD MORELIA

Universidad Nacional Autónoma de México

Escuela Nacional de Estudios Superiores
Unidad Morelia

Tecnologías para la Información en Ciencias

Inteligencia Artificial

Búsquedas no informadas

Situación problema

Este trabajo se realizó con el apoyo del Programa UNAM-DGAPA-PAPIME dentro del Proyecto para la innovación y mejoramiento para la enseñanza, con clave PE110324, Diseño didáctico multi-seriado para potenciar habilidades de modelación matemática y reflexión crítica en el aprendizaje de la Inteligencia Artificial.

Presentación

El presente diseño didáctico lleva por título “Enredos” y es una situación-problema que admite el uso de representaciones materiales, geométricas, de lenguajes formales, aritméticas y algebraicas. Dentro de la asignatura de Inteligencia Artificial, corresponde al contenido temático de *representación y búsqueda de soluciones*; en él, permite motivar la consideración y valoración de opciones para representar un espacio de estados, además de proveer un ejemplo concreto útil para ilustrar estrategias de búsquedas no informadas.

Su elaboración incluye un documento y un recurso interactivo. El documento tiene, a su vez, dos secciones. La primera es un manual para estudiantes que incorpora un documento de presentación de la situación, una hoja de trabajo y un manual donde el cuaderno interactivo se ha editado para impresión; la segunda es un manual para docentes con una guía para el uso de material concreto, orientaciones didácticas para su implementación y una edición del cuaderno interactivo que permite anticipar las respuestas que dará el alumnado.

El diseño se desarrolló en un proceso metodológico iterativo de investigación basada en diseño, que permitió observar que el diseño incide positivamente en el aprendizaje de habilidades de representación, motivando a que el estudiantado identifique criterios para validar sus propuestas a partir de (1) la conveniencia para facilitar la programación asociada al procesamiento computacional, (2) la eficiencia al analizar en términos de tiempo el algoritmo de búsqueda, (3) la pertinencia para resolver la situación contextual presentada inicialmente y (4) la complejidad para interpretar las soluciones que se encuentran utilizando algoritmos de inteligencia artificial.

ESCUELA NACIONAL DE ESTUDIOS SUPERIORES
UNIDAD MORELIA

Universidad Nacional Autónoma de México

Escuela Nacional de Estudios Superiores
Unidad Morelia

Tecnologías para la Información en Ciencias

Inteligencia Artificial

Búsquedas no informadas

Manual para estudiantes

Este trabajo se realizó con el apoyo del Programa UNAM-DGAPA-PAPIME dentro del Proyecto para la innovación y mejoramiento para la enseñanza, con clave PE110324, Diseño didáctico multi-seriado para potenciar habilidades de modelación matemática y reflexión crítica en el aprendizaje de la Inteligencia Artificial.

Enredos

Enredos es un juego que requiere de un tablero cuadrado y dos cuerdas de distintos colores, que se sujetan en las esquinas como se muestra en la figura 1.

Figura 1: Posición inicial de los enredos.

Los movimientos que podemos realizar en este juego, son dos: *torcer* y *rotar*.

- *Torcer*: Nombremos A, B, C y D a los extremos de las cuerdas en los vértices del cuadrado, comenzando por la esquina superior izquierda y en dirección contraria a las manecillas del reloj. Al aplicar esta operación, los segmentos de cuerda de los lugares C y D se intercambiarán de sitio, siendo que la cuerda del lugar C pasa por debajo de la que que se ubicaba en D.

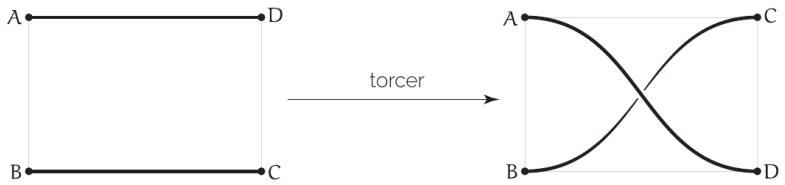

Figura 2: Se muestra el resultado de torcer a partir del estado inicial.

- *Rotar*: Con esta operación se gira el tablero en dirección de las manecillas del reloj.

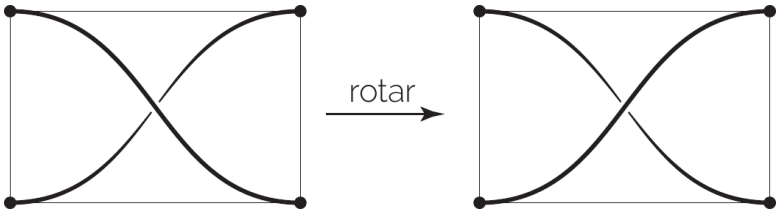

Figura 3: Se muestra el resultado de torcer a partir del estado anterior. Observa que la cuerda que está por encima ha cambiado de lugar.

Los movimientos pueden realizarse en cualquier orden. La aplicación sucesiva de varias de ellas resultará en un *enredo*.

Dado un enredo particular, ¿es posible regresar a la posición inicial utilizando estos dos movimientos? Es importante insistir en que estos dos pasos son las únicas operaciones válidas.

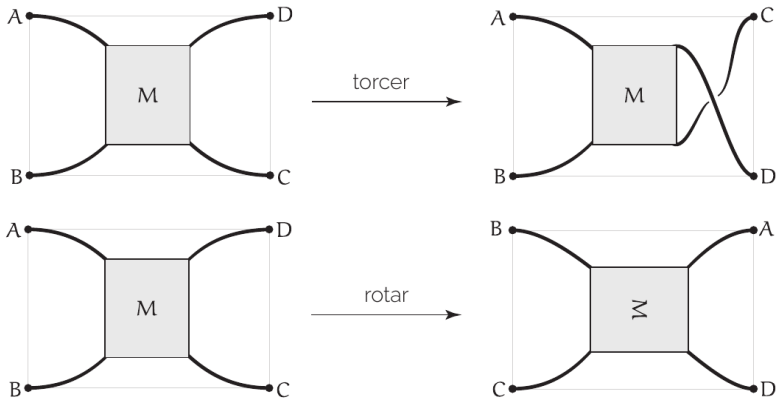

Figura 4: Dado un enredo particular M, se ilustran los dos únicos pasos que es posible realizar con él.

1. Realiza la secuencia de pasos *torcer*, *torcer*, *torcer*, *rotar* y *torcer*. Luego, desenrédala.
2. Realiza la secuencia de pasos *torcer*, *torcer*, *torcer*, *rotar*, *torcer*, *torcer*, *torcer*, *torcer*, *torcer*, *rotar*, *torcer* y *torcer*. Luego, desenrédala.
3. Representa formas organizadas de buscar la solución en la hoja de trabajo adjunta.
4. Desarrolla un algoritmo de búsqueda para encontrar la secuencia que desenreda cualquier enredo que te presenten.

Hoja de trabajo

Nombre:

¿Cómo podemos hacer una representación que nos ayude a explorar ordenadamente, utilizando dibujos de los enredos?

¿Cómo podemos hacer una representación que nos ayude a explorar ordenadamente, utilizando secuencias de caracteres?

¿Cómo podemos hacer una representación que nos ayude a explorar ordenadamente, aprovechando la forma aritmética de los enredos?

¿Cómo valorarías cada una de las representaciones anteriores de acuerdo a los siguientes criterios?

- **Que permita considerar todas las operaciones posibles y sus resultados, partiendo desde un estado particular.**
- **Que ayude a explorar caminos ordenadamente, sin perder la secuencia.**
- **Que facilite identificar cuando llegamos a un paso que ya habíamos encontrado antes, para reducir nuestra búsqueda.**
- **Que permita registrar los puntos en donde hemos tomado decisiones, para poder retomarlos y considerar caminos alternativos.**

Cuaderno de Jupyter

En esta actividad buscamos un algoritmo que permita desenmarañar un enredo creado con cuerdas después de repetir dos operaciones en reiteradas ocasiones. Para ello se «aritmetizan» los enredos; esto es, a cada enredo se la asigna un valor que depende de la secuencia de pasos que se llevó a cabo y, para poderlo desenredar, se debe llegar a aquel que tiene asignado el 0.

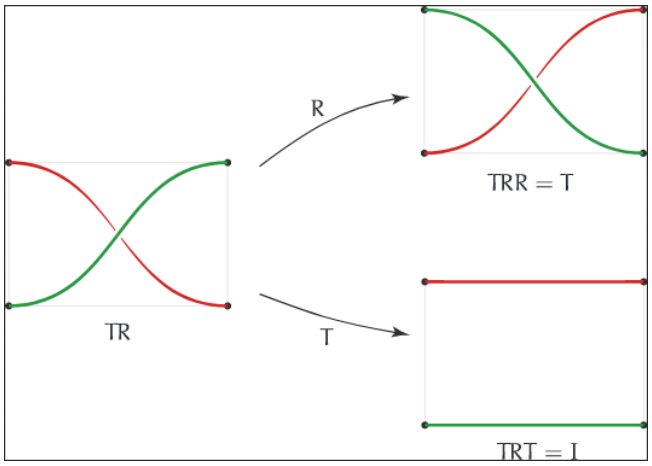

Figura 1: A partir de un enredo particular, se muestran los efectos de torcer o rotar.

Representación

```
import copy

from collections import deque
```

A continuación, definiremos la clase **Tangle** para representar los enredos. A cada enredo se le asocia un valor y la secuencia de pasos que llevaron a él, escrita como una cadena de caracteres *R* y *T* para indicar las operaciones Rotar y Torcer, respectivamente.

```
class Tangle:

    def __init__(self,sequence:str,value:int):
        self.sequence = sequence
        self.value    = value


    def __eq__(self,other):
        return (self.sequence == other.sequence)


    def __copy__(self):
        return Tangle(self.sequence, self.value)


    def __deepcopy__(self,memo):
        sequence = copy.deepcopy(self.sequence,memo)
        value    = copy.deepcopy(self.value   ,memo)
        return Tangle(sequence,value)


    def __str__(self):
        return "El enredo es \"{}\" y se le asocia el valor {}.".format(self.sequence,
    self.value)
```

De acuerdo con la definición de **eq**, ¿cuándo dos enredos son iguales?

¡Coloca aquí tus respuestas!

```
x = Tangle("",0)
print(x)
```

```
>>> El enredo es "" y se le asocia el valor 0.
```

¿Qué enredo representa **x**?

¡Coloca aquí tus respuestas!

Torcer y rotar

A continuación definiremos la operación Torcer.

```
# Función que dado un 'enredo' aplicará a este la operación 'Torcer', resultando en:
#   - Agregar un caracter 'T' a la secuencia que describe al enredo.
#   - Modificar el valor asociado al enredo.

def twist(tangle: Tangle):

    tangle.sequence += 'T'

    if( tangle.value != float('-inf') ):
        tangle.value += 1
```

Recordemos que algún enredo podría tener un valor que no sea racional. De acuerdo al código ¿qué sucede en este caso?

¡Coloca aquí tus respuestas!

En seguida, mostraremos un ejemplo. Comenzaremos con el enredo ***x*** que representa el estado inicial y realizaremos una operación de Torcer.

```
x = Tangle("",0)
print(x)
twist(x)
print(x)
```

```
>>> El enredo es "" y se le asocia el valor 0.
>>> El enredo es "T" y se le asocia el valor 1.
```

A continuación, definiremos la operación Rotar.

```
# Función que dado un 'enredo' aplicará a este la operación 'Rotar', resultando en:
#   - Agregar un caracter 'R' a la secuencia que describe al enredo.
#   - Modificar el valor asociado al enredo.

def rotate(tangle: Tangle):

    tangle.sequence += 'R'

    # Caso 1:
    if( tangle.value == 0 ):
        tangle.value = float('-inf')
    # Caso 2:
    elif( tangle.value == float('-inf') ):
        tangle.value = 0
    # Caso 3:
    else:
        tangle.value = -(1/tangle.value)
```

De acuerdo con el código, ¿cómo se atiende el caso donde hay un enredo que no es racional? ¿Podría existir alguna excepción que este código no esté considerando?

¡Coloca aquí tus respuestas!

En seguida, mostraremos otro ejemplo. Comenzaremos del estado inicial y realizaremos una operación de Rotar.

```
x = Tangle("",0)
print(x)
rotate(x)
print(x)
```

```
>>> El enredo es "" y se le asocia el valor 0.
>>> El enredo es "R" y se le asocia el valor -inf.
```

Ejercicio 1

Diseña y programa una función que dada una secuencia de caracteres *T* y *R* sea capaz de generar y devolver un enredo al que se le han aplicado estas operaciones.

Nota: *T* representa la operación torcer y *R* representa la operación rotar.

```
def processor(sequence: str):

    # Espacio para realizar Ejercicio 1.
```

```
print(processor("TRTTT"))
```

¡Coloca aquí tus respuestas!

Ejercicio 2

Una vez que hayas resuelto el Ejercicio 1, usa tu función para generar el enredo descrito por la secuencia *"TTTTRTTR"*. Posteriormente aplica las operaciones necesarias para que este enredo sea "desenredado".

Nota: Las operaciones utilizadas deberán estar en la siguiente celda, se debe ver en pantalla cuál es la descripción del enredo *"TTTTRTTR"* y las operaciones aplicadas.

```
# Espacio para realizar Ejercicio 2

# w representa enredo TTTTRTTR

# Imprimir en pantalla cómo se describe w

# Operaciones aplicadas

# Imprimir en pantalla cómo se describe w
```

¡Coloca aquí tus respuestas!

Búsqueda

Nodos en el árbol de búsqueda

```python
class Node:

    def __init__(self,tangle:Tangle,level:int=0):
        self.tangle = tangle
        self.level  = level


    def __eq__(self, other):
        return self.tangle.value == other.tangle.value


    def __copy__(self):
        return Node(self.tangle,self.level)


    def __deepcopy__(self,memo):
        tangle = copy.deepcopy(self.tangle,memo)
        level  = copy.deepcopy(self.level ,memo)
        return Node(tangle,level)


    def __str__(self):
        return f"El nodo tiene las siguientes características: \n \t\t Enredo: {self.
    tangle.sequence} \n \t\t Valor : {self.tangle.value} \n \t\t Nivel : {self.level}"
```

De acuerdo con su definición, ¿qué significa que **"__eq__"** resulte verdadero para dos nodos?

¡Coloca aquí tus respuestas!

Teniendo en cuenta la pregunta anterior, ¿cuál es la distinción entre los enredos físicos, los enredos con la definición de clase *Tangle* y los nodos que se acaban de definir?

¡Coloca aquí tus respuestas!

```python
# Enredo de la clase Tagle sin secuencia de operaciones
x_original = Tangle("",0)

# Enredo de la clase Tagle rotado 4 veces
x_rotated  = copy.deepcopy(x_original)
```

```
6
7 rotate(x_rotated)
8 rotate(x_rotated)
9 rotate(x_rotated)
10 rotate(x_rotated)
11
12 # Resultado de la comparación
13 print( x_original == x_rotated )
```

```
1 node_original = Node(x_original)
2
3 node_rotated  = Node(x_rotated)
4
5 # Resultado de la comparación
6 print( node_original == node_rotated )
```

El ejemplo anterior, ¿concuerda con lo que observaste en tu última respuesta o la modifica de alguna manera?

¡Coloca aquí tus respuestas!

Ejercicio 3

A continuación, usa la función diseñada en el Ejercicio 1 para construir el enredo "TTTTTTTTRTTTTRTTRTR" que nos servirá de inicio para empezar a desenredar automáticamente.

```
1 initial_tangle = # Espacio para el ejercicio 3
```

```
1 initial_node = Node(initial_tangle)
2 print(initial_node)
```

¡Coloca aquí tus respuestas!

¿Cómo podemos generar los nodos hijo dentro del árbol de búsqueda para este problema?

¡Coloca aquí tus respuestas!

Aplicar una u otra de las operaciones a un nodo genera dos hijos posibles. Afirmamos que, al repetir ese proceso, se construye un árbol, puesto que un mismo nodo nunca aparece dos veces.

De hecho, un mismo enredo no aparece dos veces. ¿Por qué se puede afirmar esto último?

¡Coloca aquí tus respuestas!

```
# Función que aplica una operación a un nodo del árbol de búsqueda, regresando un nuevo
    nodo con la operación aplicada.
#   - Genera el nuevo nodo que en un futuro será el modificado.
#   - Aplica la operación al 'Tangle' del nuevo nodo.
#   - Aumenta un nivel en el árbol de búsqueda para el nuevo nodo.
#   - Regresa en nodo modificado.

def apply_operation(node,operation):

    modified_node = copy.deepcopy(node)

    if( operation == "twist" ):
        twist( modified_node.tangle )
    elif( operation == "rotate" ):
        rotate( modified_node.tangle )

    modified_node.level += 1

    return modified_node
```

En las siguientes tres celdas se presenta una operación aplicada al nodo que generamos hace algunas celdas, este es el resultado:

```
print(initial_node)
```

```
aux_node = apply_operation(initial_node,"rotate")
```

```
print(aux_node)
```

Cuando se aplica una operación sobre un nodo, ¿qué cambios hay entre el nodo inicial y el que se genera?

¡Coloca aquí tus respuestas!

Automatización

En las siguientes líneas proponemos un código para automatizar la búsqueda.

```
# Función que automatiza la búsqueda de una secuencia que desenrede automaticamente,
#     usando búsqueda en anchura y realizando los siguientes pasos:
#
#   - Generar la frontera de búsqueda con el nodo inicial
#   - Mientras exista un nodo en la frontera de búsqueda:
#         * Tomar el primer nodo de la frontera y quitarlo de la misma.
#         * Verificar si cumple con las características del nodo objetivo.
#               > Si cumple las características, presentar la secuencia de operaciones para "desenredar"
#               > Si no cumple, generar a sus nodos hijo.
#         * Para cada nodo hijo verificar si ya lo hemos explorado,
#           y si no lo hemos explorado agregarlo a la frontera de búsqueda.

def make_search(root:Node):

    # Construcción de la frontera de búsqueda
    frontier = deque()
    frontier.append(root)

    seen_before_nodes = set([root.tangle.value])


    # Búsqueda
    while( frontier ):

        # Obtener la información del nodo a explorar y retirarlo de la frontera de búsqueda
        node = frontier[0]
        frontier.popleft()

        # Nodo objetivo, ?`lo encontramos?
        if( node.tangle.value == 0.0 ):
            print("Listo, la respuesta es: {}".format(node.tangle.sequence[ len(node.tangle.sequence) - node.level : ]) )
            break

        # Generar los nodos hijo, en caso de que el nodo no sea el deseado.
        children_nodes    = []
        allowed_operators = ["twist","rotate"]

        for operator in allowed_operators:
            new_node = apply_operation(node,operator)
            children_nodes.append( new_node )

        # Verificar si hemos explorado los nodos hijo
        for child_node,last_operation in zip(children_nodes,allowed_operators):

```

```
45              if( child_node.tangle.value not in seen_before_nodes ):
46                  frontier.append(child_node)
47                  seen_before_nodes.add(child_node.tangle.value)
```

¿Qué utilidad tiene el conjunto **seen_before_nodes**?

¡Coloca aquí tus respuestas!

Recapitulando

```
1 initial_tangle = processor("TTTTRTTR")
```

```
1 initial_node = Node(initial_tangle)
2 print(initial_node)
```

```
1 make_search(initial_node)
```

A partir del análisis del código y haciendor referencia a él, responde las siguientes preguntas:

1. ¿Se encuentra siempre la solución?
2. ¿La solución que se encuentra es óptima? ¿Por qué?

¡Coloca aquí tus respuestas!

Las computadoras no entienden los decimales

Supongamos que tenemos una secuencia de operaciones en la cual hemos aplicado 10^{400} veces la operación torcer y al final la operación rotar. Responde:

1. ¿Cuál será el valor asociado a este enredo?
2. ¿Que valor almacenará la computadora?
3. Dado el código anterior, ¿qué pasará en el algoritmo? Considera el error de representación numérica.

¡Coloca aquí tus respuestas!

```
# ?Esto te da alguna sugerencia? Juega con los valores del exponente

k = (1)/(10**400)
print(k)
print( k == 0.0 )
```

Representar con fracciones el valor asociado a cada enredo resulta conveniente para atender la situación descrita anteriormente. Modifica los códigos de este cuaderno interactivo, para utilizar esta representación y usa tu código para encontrar la solución para desenredar los nudos de la tarea 1.

Para enviar la respuesta a esta actividad, elabora un nuevo cuaderno interactivo solamente con los códigos necesarios para ejecutar tu propuesta.

Sugerencia: Ten cuidado con las operaciones y comparaciones.

Algunas preguntas que te ayudarán en esta tarea son:

1. ¿Qué significa en este caso torcer y rotar?
2. ¿Cambia la forma de verificar si un nodo ya ha sido visto antes?

¡Coloca aquí tus respuestas!