Trabajo final FTI: Programa que simule una máquina de Turing

* Lenguaje: Python
  + Versión: 3.9.5
* Objetivo: Simular una maquina de Turing mediante un programa por computadora
  + Requerimientos
    - Funcionales
      * Permitir antes del inicio del programa cargar un programa (grafo) y una cinta.
    - No funcionales
      * Que el programa permita ejecución paso a paso
        + Que el programa muestre una salida luego de cada cambio
      * Que el programa permita una ejecución saltándose todos los pasos
  + Estrategia
    - Pensar la maquina de Turing como maquina de estados en cuanto a la parte del programa.
    - Pensar en la cinta como un vector/lista de caracteres.
    - Pensar en la maquina de Turing como un objeto usando OOP
      * El cabezal es un objeto, que depende de la cinta y del programa(grafo)
  + Maquina de Turing:
    - ¿Que es?
      * “Es un modelo matemático de computación que define una máquina abstracta, la cual manipula símbolos en una cinta de acuerdo a un conjunto de reglas” ([fuente](https://en.wikipedia.org/wiki/Turing_machine))
    - Cuales son sus componentes
      * Cabezal (¿Dónde estoy parado en la cinta?)
      * Cinta (memoria/datos)
      * Instrucciones (grafo)
    - ¿Cómo implementamos sus componentes?
      * Cabezal:
        + Estructura de datos

Carácter

* + - * + Es simplemente un apuntador(puntero) a un lugar de la cinta en un momento dado
        + En el momento inicial apunta a la primera posición de la cinta
        + Se mueve un lugar a la izquierda o un lugar a la derecha a la vez, no mas

¿Qué pasa si la cinta se termina cuando queremos ir a la izquierda desde el principio?

La cinta es infinita, no se debe terminar para ninguno de los dos extremos

¿Cómo decidimos que acción tomar luego de movernos a la izquierda o a la derecha?

Tenemos que consultar a las instrucciones con la pregunta ¿Qué hacemos ahora que estamos parados en este símbolo?

¿Qué pasa si el cabezal no para de moverse y posiblemente entra en un bucle infinito?

El usuario es el responsable de detener la maquina. Ella misma no puede darse cuenta de si esta en un bucle o no.

* + - * + Se puede pensar como una variable dentro de la maquina de Turing que represente al símbolo actual y que tenga la posición en la cinta a la que se esta mirando (la cual va a tener que ser modificada si se cambia la longitud de la lista desde el principio).
      * Cinta
        + Estructura de datos: Lista/vector/lista enlazada
        + Se puede pensar como un objeto con varios métodos:

Siguiente(): avanza un lugar en la lista

Anterior(): retrocede un lugar en la lista

Tanto siguiente como anterior se encargan (detrás de escena) de modificar el tamaño y contenido de la lista añadiendo espacios vacíos si decidimos ir hacia los extremos de la misma.

Actual(): retorna el símbolo actual

Escribir(simbolo): escribe el símbolo en el lugar actual

Borrar(simbolo): escribe el símbolo vacío en el lugar actual

Es un caso especifico de Escribir.

EjecutarInstrucciones(lista de instrucciones) : ejecuta una serie de operaciones elementales tales como anterior, actual, escribir, borrar.

* + - * Instrucciones
        + Estructura de datos: Grafo
        + Son la parte mas complicada de la maquina de Turing

Contiene un grafo

Tiene un estado interno, el cual cambia en base al recorrido que vayamos haciendo

Tiene que tener la capacidad de a partir de un símbolo recibido decirnos que hacer en un formato que podamos entender, el cual va a ser una lista de las siguientes instrucciones elementales:

R : right (moverse un lugar a la derecha)

Rsimbolo

Moverse a la derecha hasta que encontremos el simbolo

L : left (moverse un lugar a la izquierda)

Lsimbolo

Moverse a la izquierda hasta que encontremos el simbolo

Ssímbolo (escribir símbolo en el lugar actual)

H : halt (termina la ejecución del programa por error)

END : (aceptar y terminar la ejecución del programa)

Para esto anterior se piensa un método que se llama recibirInstrucciones(símbolo): que recibe un símbolo y nos retorna una lista de instrucciones.

Tiene que tener la capacidad de cargarle un grafo

Se piensa un método cargarGrafo(grafo) que simplemente asigna el grafo interno a un grafo, ahora el tema es definir como tiene que ser este grafo, y para eso vamos a pensar una clase personalizada de grafo para la maquina de Turing.

* + - * Grafo
        + Atributos

Lista de vértices

Nodo inicial (este es el puntero a nodo que usamos para recorrer el grafo y el cual nos marca el estado en el que estamos)

* + - * + accionesSiguientes(simbolo):

retorna la lista de acciones elementales a partir de un símbolo dado, se llama inicialmente sin símbolo para iniciar la maquina de turing.

* + - * Nodo
        + atributos

Indicador que nos dice si hay trayectorias que pueden ser tomadas sin ninguna condición ni símbolo necesario, estas se recorren primero

* + - * + Siguiente(símbolo):

Si tiene una arista con este símbolo entonces el puntero a nodo actual cambia al nodo siguiente y nos retorna la lista de instrucciones elementales

* + - * Trayectoria
        + [condición de salida(puede ser un símbolo, ninguno, o el complemento del simbolo) , Nodo apuntado, acción elemental resultante]
  + ¿Cómo cargar el programa/grafo?
    - Primero hay que definir un alfabeto
    - Se puede guardar archivos con formato csv
* Objeto maquina de Turing:
  + Tiene que poder dar un paso adelante en el tiempo
    - Un desafio interesante sería hacerla que se mueva en ambas direcciones temporales, pudiendo rebobinar tantos pasos como se quiera, una alternativa es tener otro programa que haga lo opuesto, no se me ocurre como generar un programa que haga lo mismo pero al revés de manera computarizada
    - Tiene que poder dar una cantidad arbitraria de pasos adelante
  + Tiene que tener instrucciones; grafo de programa
  + Tiene que tener un cabezal y una cinta

