A yellow and blue logo with a cross and a lamp

AI-generated content may be incorrect.

INGENIERÍA EN INTELIGENCIA ARTIFICIAL

**Documentación del Proyecto: Búsqueda de Rutas Interestatales con Prolog y Python**

**Docente: Vega Soliz Luis Ariel**

**Estudiantes: José Alfredo Zambrana Cruz**

Santa Cruz - Bolivia

Junio, 20

**1. Enunciado**

El presente proyecto consiste en la implementación de un sistema de búsqueda de rutas entre estados de Estados Unidos utilizando lógica declarativa (Prolog) para la gestión del grafo y algoritmos de recorrido, en conjunto con Python para la visualización gráfica e interfaz de usuario. El objetivo principal es permitir al usuario encontrar todas las rutas posibles y la mejor ruta (de menor distancia) entre un estado origen y un estado destino.

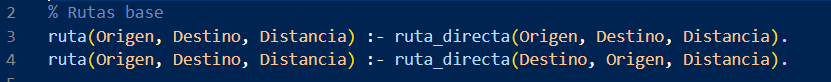
**Desarrollo**

**2. Estructura del Proyecto**

El proyecto se organiza en los siguientes archivos:

* mapa.pl: Define el grafo de estados y distancias, y contiene la lógica para encontrar rutas.
* logica.py: Conecta Python con Prolog utilizando pyswip.
* interfaz.py: Implementa la interfaz gráfica con tkinter y muestra los resultados.
* grafo\_config.py: Contiene las coordenadas gráficas de cada nodo (estado).

**2.1 Lógica en Prolog (mapa.pl)**

- Se definieron rutas bidireccionales usando el predicado **ruta**.  


- Además, se agregaron los siguientes predicados:

* **camino**: Genera una ruta válida entre dos estados sin repetir nodos. El predicado camino usa de una regla auxiliar **camino**\_ que se encarga de implementar la lógica de recursión y ver los casos para encontrar la ruta de un punto A a uno punto B .

A computer screen shot of text

AI-generated content may be incorrect.

* + camino\_ tiene 2 casos:

1.- La ruta de una ciudad A a una ciudad B es directa.

A computer screen shot of text

AI-generated content may be incorrect.

A computer screen shot of text

AI-generated content may be incorrect.2.- La ruta necesita de una o mas ciudades intermedias para llegar de una ciudad A a una ciudad B.

* **A blue screen with white text

  AI-generated content may be incorrect.mejor\_camino:** Encuentra la ruta de menor costo utilizando findall y sort.
* **ruta\_directa:** Predicados que serán nuestros hechos (datos).

Ejemplo: ruta\_directa(ciduadA, ciudadB, costo).

A screen shot of a computer program

AI-generated content may be incorrect.

**2.2 Conexión con Python (logica.py)**

Este módulo usa pyswip para realizar consultas a Prolog desde Python y extraer los resultados de:

* Todas las rutas posibles (camino)
* La mejor ruta (mejor\_camino)

A computer screen with text on it

AI-generated content may be incorrect.

**2.3 Interfaz Gráfica (interfaz.py)**

Se empleó tkinter para representar visualmente los estados como nodos y las conexiones como aristas. Las funcionalidades incluyen:

* Búsqueda de todas las rutas posibles (color azul)
* Búsqueda de la mejor ruta (color rojo)
* Visualización de los resultados en un TextBox

El módulo completo se puede encontrar en el GitHub (Anexos).

**2.4. Posicionamiento de Nodos (grafo\_config.py)**

Se asignaron coordenadas a cada estado para su correcta representación en el canvas de la interfaz.

El módulo completo se puede encontrar en el GitHub (Anexos).

**3. Logica del Proyecto (comunicación entre módulos)**

A screenshot of a computer program

AI-generated content may be incorrect.A continuación, se presenta la logica del proyecto en el entorno de desarrollo Visual Studio Code.

El proyecto está compuesto por los siguientes archivos:

* **mapa.pl**  
  Archivo en Prolog que contiene los hechos y reglas sobre las conexiones entre estados. Permite calcular todas las rutas posibles y la ruta más corta entre dos estados utilizando lógica declarativa.
* **logica.py**  
  Módulo encargado de la comunicación con mapa.pl utilizando una interfaz entre Python y Prolog (mediante pyswip). Ofrece funciones como obtener\_mejor\_ruta y obtener\_todas\_las\_rutas, que devuelven tanto la ruta como su costo total.
* **grafo\_config.py**  
  Define las posiciones en coordenadas (x, y) de cada estado dentro del canvas. Esta información es utilizada para dibujar el grafo de conexiones entre los nodos (estados) de forma visual.
* **interfaz.py**  
  Es el núcleo visual de la aplicación. Implementa una interfaz gráfica utilizando la librería tkinter de Python. Este módulo:
  + Dibuja el grafo en un Canvas, mostrando nodos (estados) y conexiones (rutas).
  + Permite al usuario ingresar un origen y un destino.
  + Integra botones para buscar todas las rutas o la mejor ruta, consultando a logica.py.
  + Ilumina gráficamente el recorrido encontrado en el canvas, diferenciando con colores (azul para todas las rutas, rojo para la mejor).
  + Muestra los resultados y el costo en un área de texto.

Esta estructura modular permite separar claramente la interfaz, la lógica, la configuración visual del grafo y la base de conocimientos lógica, facilitando el mantenimiento y escalabilidad del sistema.

**4. Funcionamiento del Sistema**

1. El usuario ingresa el estado de origen y destino.
2. Al presionar "Buscar todas las rutas", se listan todas las rutas válidas y se iluminan en azul.
3. Al presionar "Buscar mejor ruta", se muestra la ruta de menor costo en rojo.
4. La interfaz es responsiva y permite explorar diferentes combinaciones de estados.

A screenshot of a computer

AI-generated content may be incorrect.

**5. Conclusiones**

Este proyecto integra de forma exitosa la lógica declarativa de Prolog con la visualización y experiencia de usuario que ofrece Python. Es una demostración efectiva de la resolución de problemas de grafos aplicados al transporte o logística.

**6. Tecnologías empleadas:**

* SWI-Prolog
* Python 3.11+
* Tkinter
* Pyswip
* VS Code

**7. Anexos**

**enlace del repositorio git:** https://github.com/AlfredoZC/prologProject