Especificaciones del proyecto de Inteligencia Artificial Básico 2017-2

Propósito:

Poner en práctica los conocimientos adquiridos durante el curso de *Inteligencia Artificial Básico 2017-2*

Descripción del problema:

El sistema de transporte colectivo **Metro** de la **CDMX** es un modo de transporte eficiente y rápido… si sabes navegar bien en él. Por este propósito, has decidido aplicar tus conocimientos adquiridos en el curso de *Inteligencia Artificial Básico 2017-2*. Durante este curso aprendiste sobre cómo plantear un problema, así como algoritmos para solucionar estos posibles problemas.

Con este motivo, quieres desarrollar un programa capaz de decirte cuál es la ruta más óptima para ir de una estación a otra. Para modelar el problema, utilizas las estaciones con intersecciones como posibles estados; esto significa que sólo te importan las estaciones con intersecciones:

De las anteriores estaciones entonces, sólo te importan:

- El Rosario
- Instituto del Petróleo
- Deportivo 18 de Marzo
- Martin Carrera
- La Raza

Pues son las únicas con intersecciones.

Como sólo te importa minimizar la cantidad de estaciones tomadas para llegar a un destino, únicamente tomas en cuenta el número de estaciones que hay entre una de las **estaciones con intersecciones** a **otra estación con intersección**.

Por ejemplo:

De la estación '**El Rosario**' a la estación '**Instituto del Petróleo**' hay 5 estaciones:

1. Tezozómoc
2. Azcapotzalco
3. Ferrería
4. Norte 45
5. Vallejo

¡Suerte y que te diviertas!

Evaluación:

Primera Parte: Planteamiento del Problema

8 puntos:

Modelar el mundo del problema.

Tu modelo debe cumplir con la descripción del problema planteado en la página 1 de este documento. De no cumplir con estas especificaciones **no es válido y automáticamente erróneo**.

Para modelar el problema puedes sólamente tomar en cuenta las siguientes estaciones para simplificarte la implementación:

> *El Rosario, Instituto del Petróleo, Deportivo 18 de Marzo, Martin Carrera, La Raza, Consulado, Guerrero, Garibaldi, Morelos, Tacuba, Hidalgo, Bellas Artes, Tacubaya, Balderas, Pino Suárez, Candelaria, San Lázaro, Mixcoac, Centro Médico, Chabacano, Jamaica, Zapata, Ermita, Santa Anita*

Si hay alguna estación con intersecciones conectada con estas que se encuentre en medio de dos estaciones listadas en la caja de arriba, puedes ignorarla.

> En la imágen de arriba se puede apreciar que ***Salto del Agua*** se encuentra entre las estaciones ***Balderas*** *(Listada en la caja anterior)* y ***Pino Suárez*** *(Listada en la caja anterior)*.
>
> Como la estación ***Salto del Agua*** no se encuentra enumerada en la caja superior, puedes ignorar el camino y sólo considerar la ruta ***Balderas → Pino Suárez***.

Indicaciones adicionales y recomendaciones:

- Puedes guiarte en los grafos de muestra incluidos en el archivo ***system.py*** para darte una idea sobre cómo puedes modelar el problema.

- Recuerda que puedes adaptar al sistema según tu implementación del modelado del problema. Utiliza el parámetro '***suc_fn***' cuando declares el método constructor de la clase '***Problem'*** para especificar tu función de sucesores (También puedes utilizar la que he programado por defecto)

Ej:

```
def mi_funcion(estado):
        .
        .
        .

problema = Problem(suc_fn = mi_funcion, start = Coyoacán, goal = Hidalgo)
```

- Para especificar tu propio grafo tienes que utilizar el parámetro *'**space**'*. Puedes ver un ejemplo de su uso en las primeras lineas del método ***main( )***.

Ej:

```
problem = Problem(start = 'A', goal='F', space='g3', heur=heuristica)
solv = Solver()
```

- Utiliza el parámetro '***goal_fn***' cuando declares el método constructor de la clase '***Problem'*** para especificar tu función de sucesores (También puedes utilizar la que he programado por defecto)

- Para ayudarte en tu implementación del modelo del mundo, he programado 3 distintos grafos y además, en las primeras líneas del método ***main()*** he programado también cómo se utiliza la clase ***Solver***, que implementa cada uno de los algoritmos de *búsqueda no informada e informada*.

- También he programado mi solución al problema descrito en la página 1 de este documento. La puedes utilizar para guiarte en tu modelado. **Cualquier indicación de que efectuaste Ingeniería Inversa para obtener mi solución será motivo de cancelación del proyecto. Esto aplica también para las implementaciones de BFS, UCS y A*. Se utilizaran herramientas externas para descartar trampas.**

Segunda Parte: Algoritmia

NOTA IMPORTANTE:
Si no pudiste resolver la primera parte del proyecto, ¡no te preocupes! Todavia puedes pasar.
Implementa los algoritmos en el fichero ***system.py***.
Crea tus instancias de **Problem** de la siguiente manera:

problema = **Problem**(start = *'A'*, goal = *'F'*, space = *'g3'*, heur = *heuristica*)

Y solamente utiliza una instancia de la clase **Solver** para probar tus implementaciones de los algoritmos.

2 puntos:

Implementar BFS.

Para resolver tu problema de búsqueda, debes implementar el algoritmo de *Búsqueda por Amplitud Primero* ***(Breadth First Search)***. Si no pudiste implementar el problema de búsqueda, no te preocupes: sólo basta con programar **BFS** en la clase ***Solver***.

Indicaciones adicionales y recomendaciones:

- Puedes utilizar como base el algoritmo **DFS** pre-programado dentro del archivo ***system.py*** para guiarte en el uso del sistema (Cómo están representados los nodos, etc.)
- Si no puedes implementar el algoritmo por falta de conocimientos de Python 2.7, puedes escribir el pseudocódigo solamente. Este tendrá un valor de **1 punto** únicamente.
- Puedes guiarte en los algoritmos programados en el ***moodle***.

2 puntos:

Implementar UCS.

Para resolver tu problema de búsqueda, debes implementar el algoritmo de *Búsqueda de Costo Uniforme* ***(Uniform Cost Search)***. Si no pudiste implementar el problema de búsqueda, no te preocupes: sólo basta con programar **UCS** en la clase ***Solver***.

Indicaciones adicionales y recomendaciones:

- Puedes utilizar como base el algoritmo **DFS** pre-programado dentro del archivo ***system.py*** para guiarte en el uso del sistema (Cómo están representados los nodos, etc.)
- Si no puedes implementar el algoritmo por falta de conocimientos de Python 2.7, puedes escribir el pseudocódigo solamente. Este tendrá un valor de **1 punto** únicamente.
- Puedes guiarte en los algoritmos programados en el ***moodle***.

4 puntos:

Implementar A*.

Para resolver tu problema de búsqueda, debes implementar el algoritmo A* ***(A star)***. Si no pudiste implementar el problema de búsqueda, no te preocupes: sólo basta con programar **A*** en la clase ***Solver***.

Indicaciones adicionales y recomendaciones:

- Puedes utilizar como base el algoritmo **DFS** pre-programado dentro del archivo ***system.py*** para guiarte en el uso del sistema (Cómo están representados los nodos, etc.)

- Si no puedes implementar el algoritmo por falta de conocimientos de Python 2.7, puedes escribir el pseudocódigo solamente. Este tendrá un valor de **1 punto** únicamente.

- Puedes guiarte en los algoritmos programados en el ***moodle***.

- **Recuerda que para A* es necesario programar tu heurística. Por defecto el sistema utiliza la heurística que he programado.** Para utilizar tu heurística, simplemente en el sistema utilizas el parámetro ***heur*** dentro del método constructor de la clase ***Problem***

Puntaje

Para aprobar el curso y obtener tu constancia es necesario juntar **8 puntos**.

Objeto a evaluar	**Puntaje**
Modelar el mundo del problema	_____ *puntos obtenidos de 8*
Implementar BFS	_____ *puntos obtenidos de 2*
Implementar UCS	_____ *puntos obtenidos de 2*
*Implementar A**	_____ *puntos obtenidos de 4*

Observaciones: