



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE INGENIERÍA

Materia

Inteligencia Artificial

Proyecto 1

Búsqueda A-estrella

Presenta

López Becerra Ricardo

Navarrete Zamora Aldo Yael

Núñez Hernández Diego

Fecha de entrega

Jueves 7 de octubre del 2022

Semestre

2023-1

¿Qué es el método A*?

El método A* (A-estrella) es un algoritmo de recorrido de gráficos para encontrar rutas. Es muy utilizado ya que es completo y óptimo, lo que significa que siempre va a llegar a una solución y que la solución encontrada es la mejor. El algoritmo fue publicado por primera vez en 1968 por Peter Hart, Nils Nilsson y Bertram Raphael. Este algoritmo puede verse como una extensión del algoritmo de Dijkstra. Su principal optimización es que utiliza una función heurística para tomar la decisión de qué nodo expandir. Sí, la función heurística utilizada es admisible (Nunca sobreestima el el coste de alcanzar el objetivo) se garantiza que la solución va a ser óptima.

En este caso, la función heurística es la distancia en línea recta desde cualquier nodo al destino. Esta distancia siempre será menor que cualquier otra ruta, por lo cumple con ser una función heurística admisible.

La principal diferencia de este algoritmo con Breadth First Search es la toma de la decisión de qué nodo expandir. Esta decisión se realiza tomando el nodo donde la $f(n)$ es la menor entre todas las $f(n)$ de todos los nodos ya visitados pero no expandidos. $f(n)$ se puede obtener de la siguiente manera:

$$f(n) = g(n) + h(n)$$

Donde $g(n)$ es la distancia desde el origen hasta el nodo n y $h(n)$ es la distancia heurística entre el nodo n y el destino.

Algoritmo de A*

1. Determinar la ciudad actual.
2. Sumar el valor recorrido más el valor heurístico de cada ciudad adyacente a la actual.
3. Elegir la ciudad con menor función f de entre todas las ciudades visitadas.
4. Expandir dicha ciudad.
5. Repetir hasta que se tenga que expandir la ciudad destino o hasta que todas las ciudades hayan sido visitadas.

El pseudocódigo de búsqueda A* es representado de la siguiente forma.

```
distancia_f = mapa que describa la función f.  
distancia_g = mapa que describa la distancia hasta ese nodo desde el origen.  
vino_de = mapa que al final del algoritmo describe de qué otro nodo proviene (recuperar camino).
```

```
cola = cola de prioridad inicializada con la distancia 0 y el nodo inicial.
```

```
Ciclo mientras la cola tenga elementos:
```

```
    distancia_f_actual, nodo_actual = primer elemento de la cola de prioridad.
```

```
    Si el nodo actual es el nodo final:
```

```
        Imprime 'se ha llegado al nodo final'
```

```
        Regresa distancia_g, vino_de
```

```

Para cada nodo_vecino, distancia_f y distancia heurística del nodo actual:
    Guarda la suma distancia_g del nodo actual y la distancia entre el nodo vecino y el
actual en una variable temporal.
    Si la variable temporal es menor a la distancia_g del nodo siguiente:
        distancia_g del nodo siguiente será la distancia temporal.
        distancia heurística será la distancia heurística del nodo actual.
        F = G + H
    Indicar de dónde vino el nodo en vino_de
    Insertar en la cola de prioridad a la distancia_f y al siguiente nodo como un par.
Regresa distancia_g, vino_de

```

Manual de Usuario

Requerimientos

- Una computadora.
- Python (versión > 3.7).

Primeramente, el archivo `astar.py` deberá ser descargado desde el repositorio listado en el tríptico. Ejecutar el programa es muy sencillo. Puede hacerse de dos formas:

1. Mediante consola (bash,zsh) o ventana de comandos.
 - Ubicarse en la ruta donde se encuentra el archivo descargado con el comando `cd .`
 - Una vez en la misma ruta del archivo `.py`, se deberá ejecutar el archivo con el comando `python3 <nombre_del_archivo>.py`.
2. Mediante editor de texto.
 - Este archivo será abierto en algún editor de texto o IDE (*Visual Studio Code, Sublime Text, etc.*).
 - Una vez abierto el archivo, se ejecutará mediante una extensión o botón propio del editor de texto.

Una vez ejecutado el programa se desplegará un menú de la siguiente manera

```

*-----*
| METODO A-Estrella, INTELIGENCIA ARTIFICIAL |
*-----*
|  MAPA DE CIUDADES DISPONIBLES PARA VIAJE  |
|                                             |
| -Arad          -Bucharest      -Craiova   |
| -Dobreta       -Eforie         -Fagaras   |
| -Giurgiu        -Hirsova       -Iasi       |
| -Lugoĵ         -Mehadia        -Neamt      |
| -Oradea         -Pitesti       -Sibiu      |
| -Rimnicu Vilcea -Timisoara     -Urziceni   |
| -Vaslui         -Zerind        |
*-----*
      Ingresa el nombre de la ciudad de inicio:

```

El usuario deberá escribir alguna ciudad de las listadas en el mapa de Rumania. Seguido, el programa se ejecutará la búsqueda A* de dicha ciudad hacia Bucharest, la cuál generará la salida paso a paso, así como la trayectoria final y costos totales de la misma.

Suponiendo que la ciudad de inicio sea Arad, la salida es la siguiente.

Ingresa el nombre de la ciudad de inicio: Arad

Paso 0: La ciudad a expandir es Arad

La función $f(n)$ de Sibiu es $f(n) = 140 + 253 = 393$. Se agrega a la cola de prioridad.

La función $f(n)$ de Timisoara es $f(n) = 118 + 329 = 447$. Se agrega a la cola de prioridad.

La función $f(n)$ de Zerind es $f(n) = 75 + 374 = 449$. Se agrega a la cola de prioridad.

Paso 1: La ciudad a expandir es Sibiu

La distancia a Arad pasando por Sibiu: (280) es mayor

a la distancia mínima actual para llegar a Arad:(0)

La función $f(n)$ de Fagaras es $f(n) = 239 + 178 = 417$. Se agrega a la cola de prioridad.

La función $f(n)$ de Oradea es $f(n) = 291 + 380 = 671$. Se agrega a la cola de prioridad.

La función $f(n)$ de Rimnicu es $f(n) = 220 + 193 = 413$. Se agrega a la cola de prioridad.

Paso 2: La ciudad a expandir es Rimnicu

La función $f(n)$ de Craiova es $f(n) = 366 + 160 = 526$. Se agrega a la cola de prioridad.

La función $f(n)$ de Pitesti es $f(n) = 317 + 98 = 415$. Se agrega a la cola de prioridad.

La distancia a Sibiu pasando por Rimnicu: (300) es mayor

a la distancia mínima actual para llegar a Sibiu:(140)

Paso 3: La ciudad a expandir es Pitesti

La función $f(n)$ de Bucharest es $f(n) = 418 + 0 = 418$. Se agrega a la cola de prioridad.

La distancia a Craiova pasando por Pitesti: (455) es mayor

a la distancia mínima actual para llegar a Craiova:(366)

La distancia a Rimnicu pasando por Pitesti: (414) es mayor

a la distancia mínima actual para llegar a Rimnicu:(220)

Paso 4: La ciudad a expandir es Fagaras

La distancia a Bucharest pasando por Fagaras: (450) es mayor

a la distancia mínima actual para llegar a Bucharest:(418)

La distancia a Sibiu pasando por Fagaras: (338) es mayor

a la distancia mínima actual para llegar a Sibiu:(140)

Paso 5: La ciudad a expandir es Bucharest

La ciudad a expandir es la final, se llegó al destino!

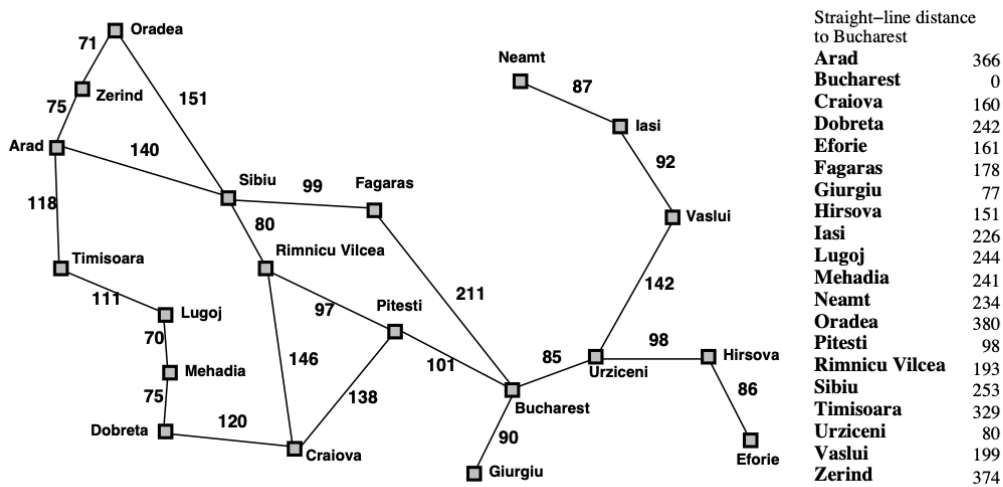
Termina

Costo: 418

Trayectoria: Sibiu(140) -> Rimnicu(220) -> Pitesti(317) -> Bucharest(418)


Ejemplos

Romania with step costs in km



Arad a Bucharest

Paso	Ciudad extendida
0. $0+366=366$ _{Ara}	
1. $140+253=393$ _{Ara-Sib} $118+329=447$ _{Ara-Tim} $75+374=449$ _{Ara-Zer}	Arad
2. 447 _{Ara-Tim} 449 _{Ara-Zer} $239+178=417$ _{Ara-Sib-Fag} $291+380=671$ _{Ara-Sib-Ora} $220+193=413$ _{Ara-Sib-Rim}	Sibiu
3. 447 _{Ara-Tim} 449 _{Ara-Zer} 417 _{Ara-Sib-Fag} 671 _{Ara-Sib-Ora} $366+160=526$ _{Ara-Sib-Rim-Cra} $317+98=415$ _{Ara-Sib-Rim-Pit}	Rimnicu
4. 447 _{Ara-Tim} 449 _{Ara-Zer} 417 _{Ara-Sib-Fag} 671 _{Ara-Sib-Ora} 526 _{Ara-Sib-Rim-Cra} $418+0=418$ _{Ara-Sib-Rim-Pit-Buc} $455+160=615$ _{Ara-Sib-Rim-Pit-Cra}	Pitesti

5. 447 _{Ara-Tim} 449 _{Ara-Zer} 671 _{Ara-Sib-Ora} 526 _{Ara-Sib-Rim-Cra} 418 _{Ara-Sib-Rim-Pit-Buc} 450+0=450 _{Ara-Sib-Fag-Buc} 	Fagaras
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------

Termina

Trayectoria: Arad-Sibiu-Fagaras-Bucharest

Costo: 418 km

Mehadia a Bucharest

Paso	Ciudad extendida
0. 0+241=241 _{Meh}	
1. 75+242=317 _{Meh-Dob} 70+244=314 _{Meh-Lug}	Mehadia
2. 317 _{Meh-Dob} 181+329=510 _{Meh-Lug-Tim}	Lugoj
3. 510 _{Meh-Lug-Tim} 195+160=355 _{Meh-Dob-Cra}	Dobreta
4. 510 _{Meh-Lug-Tim} 333+98=431 _{Meh-Dob-Cra-Pit} 341+193=534 _{Meh-Dob-Cra-Rim}	Craiova
5. 510 _{Meh-Lug-Tim} 534 _{Meh-Dob-Cra-Rim} 434+0=434 _{Meh-Dob-Cra-Pit-Buc} 430+193=623 _{Meh-Dob-Cra-Pit-Rim}	Pitesti

Termina

Trayectoria: Dobreta-Craiova-Pitesti-Bucharest

Costo: 434 km

Referencias

- Herrera C., Abel. Notas de clase Inteligencia Artificial. UNAM. Facultad de Ingeniería [Consulta 4 de octubre del 2022].
- Russell, Stuart J. (2018). Artificial intelligence, a modern approach. Norvig, Peter (4ta ed.). Boston: Pearson.[Consulta 4 de octubre del 2022].