



**UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO**

**FACULTAD DE INGENIERÍA**

**Inteligencia artificial**

**Grupo 1**

**Proyecto 1: método de búsqueda Uniforme usando Python**

**Profesor: Abel Camacho**

**INTEGRANTES**

* Arreguin Portillo Diana Laura
* Calzada Martínez Jonathan Omar
* González Nova Rafael Antonio

**Objetivo:** Poner en práctica la teoría vista en clase buscando el desarrollo de el método de búsqueda uniforme con Python.

**Introducción:**

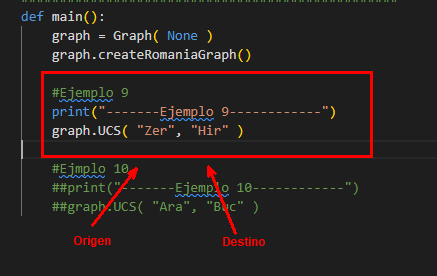
En [ciencia de la computación](https://es.wikipedia.org/wiki/Ciencia_de_la_computaci%C3%B3n), la búsqueda de costo uniforme (BCU) es un [algoritmo](https://es.wikipedia.org/wiki/Algoritmo) de [búsqueda no informada](https://es.wikipedia.org/wiki/B%C3%BAsquedas_no_informadas) utilizado para recorrer sobre [grafos](https://es.wikipedia.org/wiki/Grafo) el camino de costo mínimo entre un [nodo](https://es.wikipedia.org/wiki/V%C3%A9rtice_(teor%C3%ADa_de_grafos)) raíz y un nodo destino. La búsqueda comienza por el nodo raíz y continúa visitando el siguiente nodo que tiene menor costo total desde la raíz. Los nodos son visitados de esta manera hasta que el nodo destino es alcanzado.

En la cola, cada nodo se asocia con su costo total desde la raíz, donde se les da mayor prioridad a los caminos de costo mínimo. El nodo en la cabeza de la cola es expandido, adicionando sus nodos vecinos con el costo total desde la raíz hasta el nodo respectivo. La búsqueda de costo uniforme es completa y óptima si el costo de cada paso excede algún límite eps positivo.[1](https://es.wikipedia.org/wiki/B%C3%BAsqueda_de_costo_uniforme#cite_note-1)​ El tiempo para el caso peor y la complejidad espacial es *O(b1 + C\*/ε)*, donde C\* es el costo de la solución óptima y b es el [factor de ramificación](https://es.wikipedia.org/wiki/Factor_de_ramificaci%C3%B3n). Cuando todos los costos entre los nodos son iguales, esto se convierte en *O(bd + 1)*.[2](https://es.wikipedia.org/wiki/B%C3%BAsqueda_de_costo_uniforme#cite_note-2)​

**Desarrollo:**

**Manual:** Se probara el programa en Python 3.8.3, en el editor de texto Visual Studio Code de la siguiente manera:

Para poder señar la ruta de inicio y de final de sebe mandar a la función de la siguiente manera:



Luego procedemos a ejecutar el programa con el nombre BCU.py



\*\*\*\*\*MEJOR RUTA\*\*\*\*\*

INICIO: Zer

FIN: Hir

Name: Zer

Neighbors: Ora Ara

Distance: 71 75

Color: GREY

Distance: 0

Predecessor: without predecessor

Place: 0

==============

Name: Ara

Neighbors: Zer Tim Sib

Distance: 75 118 140

Color: GREY

Distance: 75

Predecessor: Zer

Place: 0

==============

Name: Sib

Neighbors: Ora Ara Rim Fag

Distance: 151 140 80 99

Color: GREY

Distance: 215

Predecessor: Ara

Place: 0

==============

Name: Rim

Neighbors: Cra Sib Pit

Distance: 146 80 97

Color: GREY

Distance: 295

Predecessor: Sib

Place: 0

==============

Name: Pit

Neighbors: Cra Rim Buc

Distance: 138 97 101

Color: GREY

Distance: 392

Predecessor: Rim

Place: 0

==============

Name: Buc

Neighbors: Fag Pit Giu Urz

Distance: 211 101 90 85

Color: GREY

Distance: 493

Predecessor: Pit

Place: 0

==============

Neighbors: Buc Hir Vas

Distance: 85 98 142

Color: GREY

Distance: 578

Predecessor: Buc

Place: 0

==============

Name: Hir

Neighbors: Urz Efo

Distance: 98 86

Color: GREY

Distance: 676

Predecessor: Urz

Place: 0

Podemos ver que al final hay un costo de 676, y pasa por zer-ara-sib-rim-pit-buc-urz-hir

**Conclusiones:**

**Se puede comprobar la efectividad del método de Costo Uniforme para la búsqueda de rutas optimas**

**Bibliografía:**

* Teoría proporcionada por el profesor en clase
* **Artificial Intelligence: A Modern Approach, 4th US ed**