Universidad Autónoma de Nuevo León
Facultad de Ciencias Físico Matemáticas
San Nicolás de los Garza

Algoritmos Informados

Tarea 5

Inteligencia Artificial

Docente: JUAN PABLO ROSAS BALDAZO

Matrícula	Nombre
1947480	FELIX YAHVEH ALANIS CANDELARIA
1963196	JUAN CARLOS DIAZ GONZALES
1962111	JOHANN JOSEPH VELÁZQUEZ ANTONIO

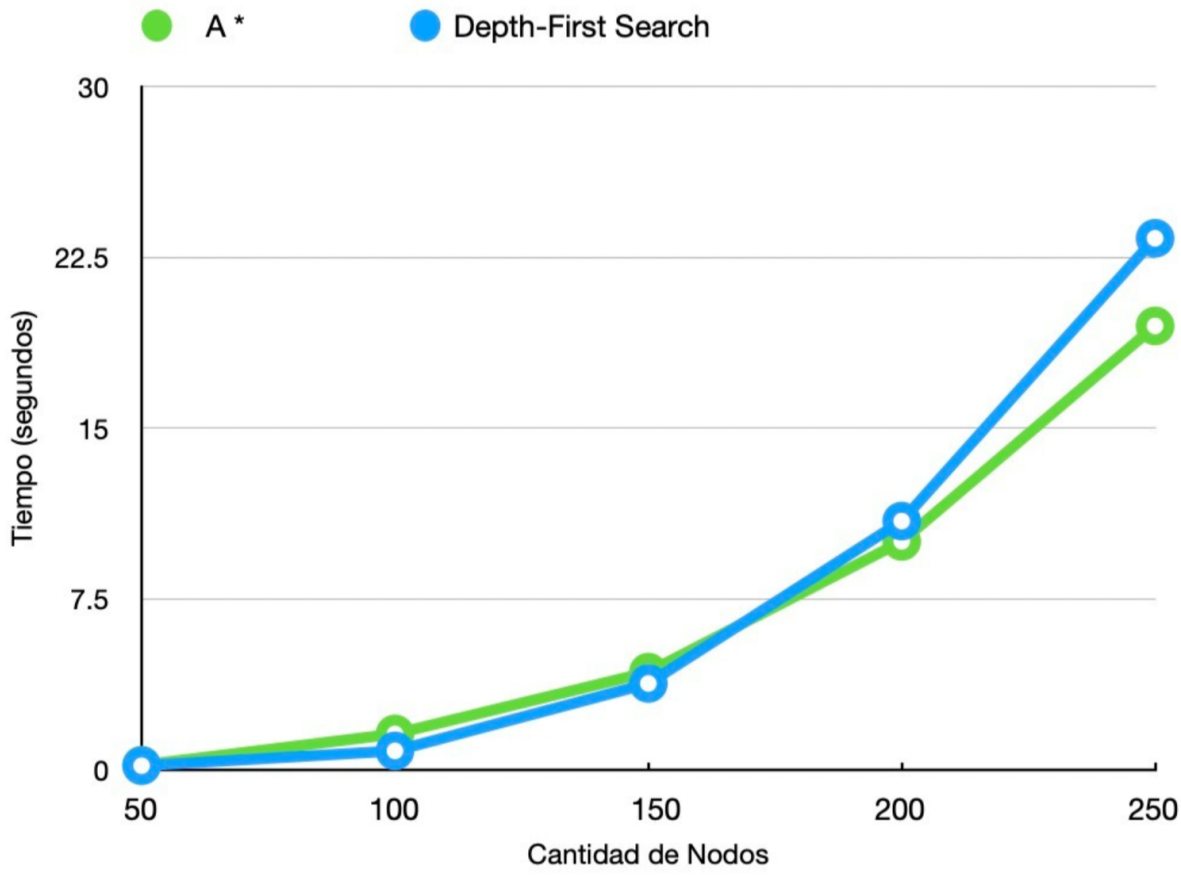
San Nicolás de los Garza, a 10 de marzo del 2023

Algoritmo A*

GITHUB: https://github.com/Johann-28/IA

```
Algoritmo a_estrella

    Incluir "grafo" como gr
    Incluir "math"
    Incluir "heapq"

    Funcion distancia(nodo_a, nodo_b)
        x1 = longitud(nodo_a)
        y1 = longitud(nodo_a)
        x2 = longitud(nodo_b)
        y2 = longitud(nodo_b)
        Retornar raiz_cuadrada((x1 - x2) ^ 2 + (y1 - y2) ^ 2)
    FinFuncion

    Funcion a_estrella(nodo_inicial, nodo_final)
        abiertos = [(0, nodo_inicial)]
        cerrados = conjunto()
        padres = diccionario()
        costos = diccionario()
        ruta = diccionario()

        costos[nodo_inicial] = 0
        ruta[nodo_inicial] = [nodo_inicial]

        Mientras que abiertos no sea vacío Hacer
            costo_actual, nodo_actual = heapq.eliminar_menor(abiertos)

            Si nodo_actual = nodo_final Entonces
                Retornar ruta[nodo_actual], costos[nodo_actual]
            FinSi

            cerrados.insertar(nodo_actual)

            Para cada nodo_vecino, peso en gr.graph[nodo_actual] Hacer
                Si nodo_vecino está en cerrados Entonces
                    Continuar
                FinSi

                nuevo_costo = costos[nodo_actual] + peso
                Si nodo_vecino no está en costos o nuevo_costo < costos[nodo_vecino] Enton
ces
                    costos[nodo_vecino] = nuevo_costo
                    heuristica = distancia(gr.graph[nodo_vecino], gr.graph[nodo_final])
                    heapq.insertar(abiertos, (nuevo_costo + heuristica, nodo_vecino))
                    padres[nodo_vecino] = nodo_actual
```

```
                    ruta[nodo_vecino] = concatenar(ruta[nodo_actual], [nodo_vecino])
                FinSi
            FinPara
        FinMientras

        Retornar nulo
    FinFuncion

    grafica = {}

    grafo = gr.Grafo(grafica)

    Para cada g en grafica Hacer
        Para cada x en grafica Hacer
            Si g = x Entonces
                Continuar
            FinSi

            camino = a_estrella(g, x)
            Escribir("El camino mas corto de ", g, " a ", x, " es: ", camino[0], " con cos
to: ", camino[1])
        FinPara
    FinPara

FinAlgoritmo
```



Grafo	Tiempo	Promedio
1(50 nodos)	0.176553964614868	
1(50 nodos)	0.177275896072388	0.191284577051799
1(50 nodos)	0.22002387046814	
2(100 nodos)	1.90430498123169	
2(100 nodos)	1.4430091381073	1.56369670232137
2(100 nodos)	1.34377598762512	
3(150 nodos)	4.20839715003967	
3(150 nodos)	4.41426205635071	4.27712941169739
3(150 nodos)	4.20872902870178	
4(200 nodos)	9.98338603973389	
4(200 nodos)	10.1339530944824	10.0201309521993
4(200 nodos)	9.94305372238159	
5(250 nodos)	20.4532761573792	
5(250 nodos)	18.952672958374	19.488408724467
5(250 nodos)	19.0592770576477	

Grafo	Tiempo	Promedio
1(50 nodos)	0.231358051300049	
1(50 nodos)	0.137994050979614	0.137994050979614
1(50 nodos)	0.128237009048462	
2(100 nodos)	0.165863037109375	
2(100 nodos)	1.1821460723877	1.1821460723877
2(100 nodos)	1.10285687446594	
3(150 nodos)	3.77639102935791	
3(150 nodos)	3.74148511886597	3.74148511886597
3(150 nodos)	3.81621909141541	
4(200 nodos)	10.9260671138763	
4(200 nodos)	10.9164490699768	10.9164490699768
4(200 nodos)	10.8981599807739	
5(250 nodos)	23.0855779647827	
5(250 nodos)	23.4820199012756	23.4820199012756
5(250 nodos)	23.4465022087097	

A*	Depth-First Search

En resumen, al aumentar el número de nodos en el grafo, se observa un comportamiento exponencial en ambos algoritmos de búsqueda. Sin embargo, al comparar su desempeño en el mismo problema, se encontró que el algoritmo A* fue ligeramente más óptimo que el algoritmo de búsqueda en profundidad. Esto se debe a la naturaleza del algoritmo A*, el cual busca siempre la mejor alternativa y se dirige hacia ella, lo que lo hace más eficiente en términos de tiempo y espacio. Estas comparaciones nos permiten entender cómo se comportan los algoritmos en diferentes situaciones y nos ayudan a seleccionar la mejor opción para resolver un problema específico.

Se realizaron 3 iteraciones en cada configuración de grafo, y se graficó el promedio utilizado.
Todas las pruebas fueron echas bajo el mismo ambiente del IDE Visual Studio Code.
Las características del equipo en que se corrió fueron las siguientes:
SO: Windows 11 Home Single Language x64
Procesador: AMD Ryzen 5 5600H with Radeon Graphics
3.30 GHz
RAM: 16,0 GB