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

IA

Tarea 4

Inteligencia Artificial

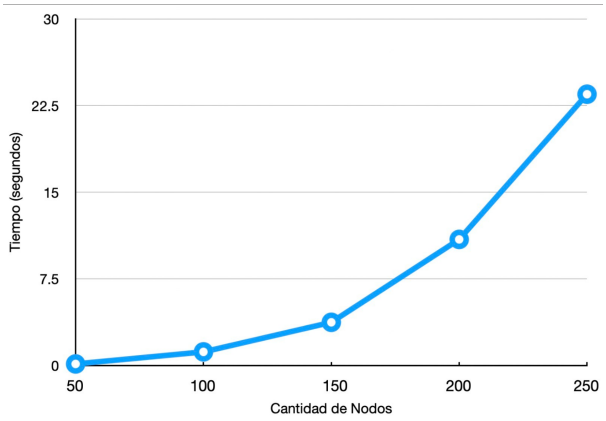
Docente: JUAN PABLO ROSAS BALDAZO

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

San Nicolás de los Garza, a 03 de marzo del 2023

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

```
importar tiempo

inicio = tiempo.actual()

clase Grafo:
    función __init__(self, grafo = {}):
        self.grafo = grafo

    función agregar_arista(self, nodo1, nodo2, peso):
        si nodo1 no está en self.grafo:
            agregar nodo1 como una lista vacía a self.grafo
        agregar nodo2 y peso a la lista de adyacencia del nodo1 en self.grafo

    función dfs(self, inicio, final):
        visitados = conjunto vacío  # Conjunto de nodos visitados
        pila = [(inicio, [], 0)]  # Pila de nodos por visitar, junto con el camino recorrido y el peso total hasta ese momento

        mientras pila no está vacía:
            nodo, camino, peso = desapilar un elemento de pila
            si nodo es igual a final:
                devolver camino concatenado con [nodo] y peso
            agregar nodo a visitados
            para cada vecino y peso_vecino en la lista de adyacencia del nodo en self.grafo:
                si vecino no está en visitados:
                    apilar (vecino, concatenar camino con [nodo], peso + peso_vecino) a pila

        devolver None  # Si no se encuentra una ruta

    función ruta_mas_corta_dfs(self, inicio, final):
        ruta, peso = llamar a la función dfs con inicio y final
        si ruta es None:
            devolver None
        devolver ruta y peso

grafica = {

}

gr = Grafo(grafica)

para cada g en grafica:
    para cada x en grafica:
        si g es igual a x:
            continuar

        path = llamar a la función ruta_mas_corta_dfs con g y x
        mostrar 'El camino mas corto de ', g, ' a ', x, ' es: ', concatenar path[0], 'con costo: ', path[1]

final = tiempo.actual()
mostrar "tiempo de ejecucion", final - inicio
```



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	

> Las conclusiones satisafcen la hipótesis de que según crecía el numero de objetos el tiempo de computo sería mayor

> 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