M.I.A.

TAREA ALGORITMO A*

José Manuel Alonso Tirado

ÍNDICE

ENUNCIADO

Aplicar el algoritmo A* para hallar la distancia mínima entre las ciudades J y F

.

Distancias en Avión a F desde las diferentes ciudades:

y la distancia por avión a F son:

	A	B	C	D	E	F	G	H	I	J	K
F	38	97	59	62	46		111	112	139	158	87

Mostrar el árbol resultante y detallar en cada paso el nodo seleccionado, así como los valores de g(n), h(n) y f(n) de los nodos generados

Distancias por las carreteras entre las diferentes ciudades:

	A	B	C	D	E	F	G	H	I	J	K
A			63	75		44					
B			61	77			64				
C				29							
D					38			56			34
E						58					59
F											
G								27		55	
H									35		44
I										40	
J											

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

Menor peso Inicial

f (G) = 55+111=166

Camino =55Km

J

Camino =40 Km

f (I) = 40+139=179

G->J Bucle

Camino = 27Km

G

I

H->I Bucle

Camino = 35+40

Camino = 64Km

f (H) = 40+35+112=187

Camino hacia G, al volver hacemos mas KM

H

H

B

Camino = 35+40+56

D->H Bucle

f (H) = 55+27+112=194

f (B) = 55+64+97=216

f (D) 40+35+56+62=193

Camino hacia K, al volver hacemos mas KM

D

K

f (K) 40+35+44+87=206

Aquí tiene Mayor peso que la inicial hacia I de 176, volvemos al nodo I

Nodo Cerrado

A

B

C

Camino = 35+40+56+38

E

f (E) 40+35+56+38+46=215

f (A) 40+35+56+75+38=244

f (B) 40+35+56+77+97=305

f (C) 40+35+56+29+59=219

Camino =

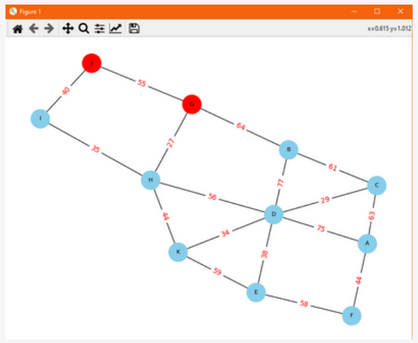
35+40+56+38+58=227

F

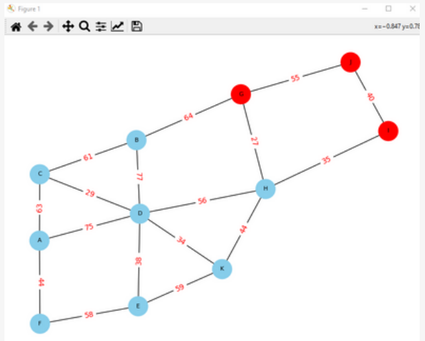
1º PROGRAMA EN PARA PINTAR LOS NODOS

```python
def astar(G, origen, destino):
    abiertos = {origen}
    cerrados = set()
    camino = []

    while abiertos:
        nodo_actual = min(abiertos, key=lambda nodo: G.nodes[nodo]['weight'] + heuristica(nodo, destino))
        abiertos.remove(nodo_actual)
        cerrados.add(nodo_actual)

        for nodo_adyacente, peso in G[nodo_actual].items():
            if nodo_adyacente in cerrados:
                continue

            distancia_estimada = G.nodes[nodo_actual]['weight'] + peso['weight']

            # Imprime los pasos
            print(f"{nodo_actual} -> {nodo_adyacente} ({peso['weight']})")

            if distancia_estimada < G.nodes[nodo_adyacente]['weight']:
                G.nodes[nodo_adyacente]['weight'] = distancia_estimada
                abiertos.add(nodo_adyacente)

                # Mover la condición del destino aquí
                if nodo_adyacente == destino:
                    break

        camino.append(nodo_actual)

    # Visualización del grafo
    plt.figure(figsize=(8, 6))
```

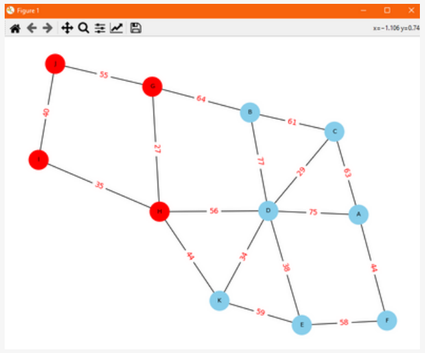
1º Iteración



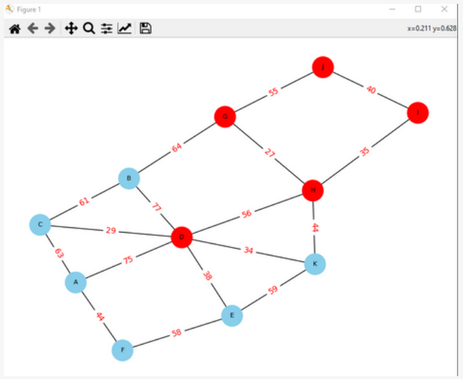
2ª Iter.



3ª Iter.

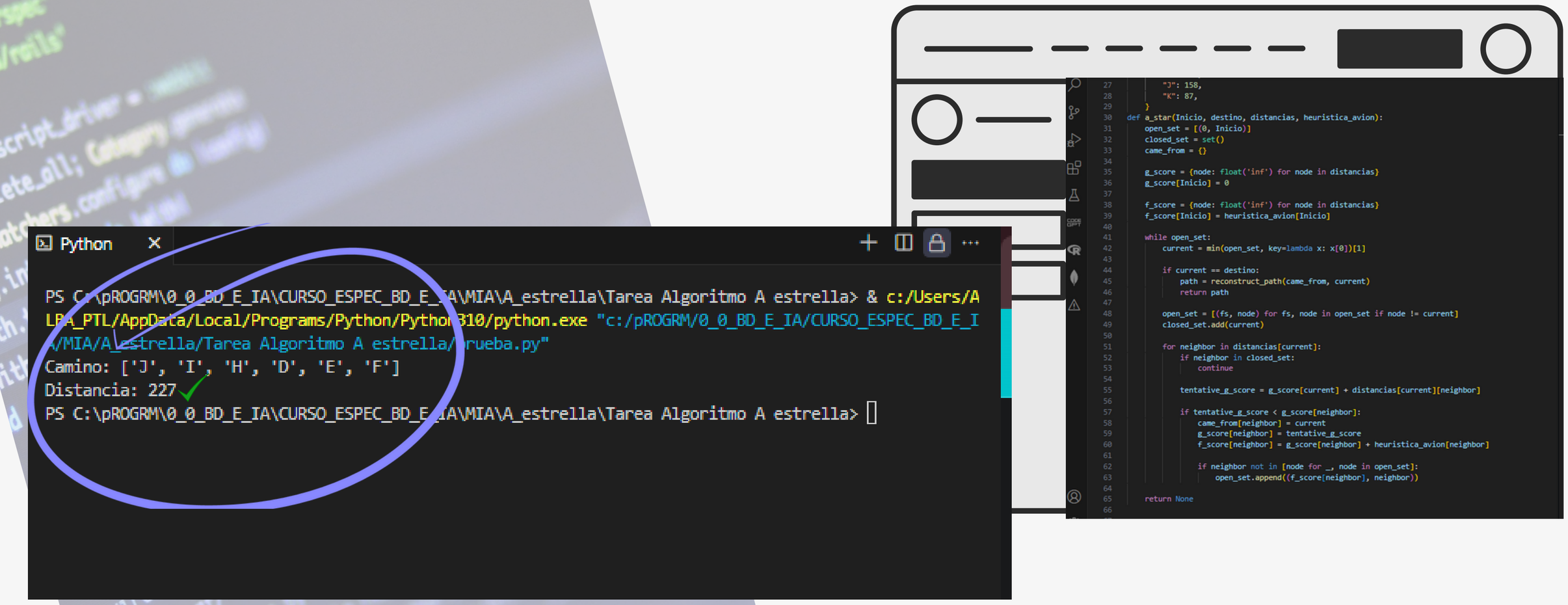


Toma el camino correcto



2 PROGRAMA EN

PARA CALCULAR EL CAMINO OPTIMO



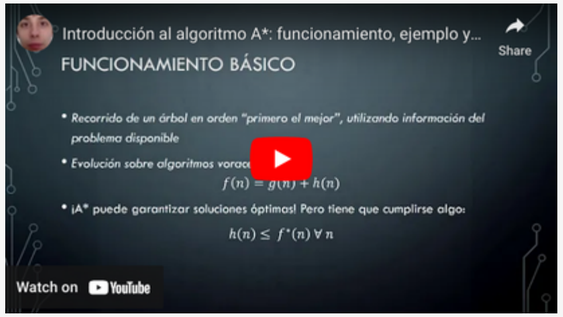
BIBLIOGRAFIA

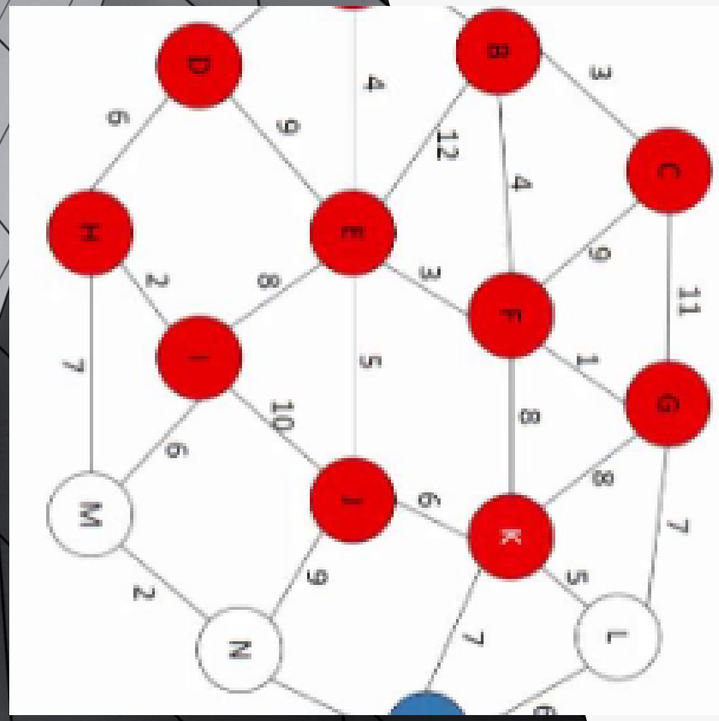
MATERIAL

Apuntes de Clase

SITIOS WEB´S

A* Wikipwedia





CORREO

jmat@hotmail.es