



Why GitHub? ▾ Enterprise Explore ▾ Marketplace Pricing ▾

Search



Sign in

Sign up

 mcollada / 03MAIR-Algoritmos-de-optimizacion

 Watch

0

 Star

0

 Fork

0

 Code

 Issues 0

 Pull requests 0

 Projects 0

 Insights

Join GitHub today

GitHub is home to over 31 million developers working together to host and review code, manage projects, and build software together.

Sign up

Dismiss

Branch: master ▾

03MAIR-Algoritmos-de-optimizacion / AG2 / Miguel_Angel_Soto_Collada_AG2.ipynb

Find file

Copy path

 mcollada Add files via upload

2932201 43 seconds ago

1 contributor

355 lines (354 sloc) | 10.8 KB



Raw

Blame

History



https://github.com/mcollada/03MAIR-Algoritmos-de-optimizacion/blob/master/AG2/Miguel_Angel_Soto_Collada_AG2.ipynb

```
In [1]: import pdb
import random
import math
from time import time

N=3000
Lista_2D=[ (random.randrange(1,N*10),random.randrange(1,N*10)) for _ in range(N)]
```

```
In [2]: #Función para calcular el tiempo de ejecución
def calcular_tiempo(f):
    def wrapper(*args,**kwargs):
        inicio = time()
        resultado = f(*args,**kwargs)
        tiempo = time() - inicio
        print("\r\nTiempo de ejecución para algoritmo: "+"{0:.25f}".format(tiempo))
        return resultado
    return wrapper

def distancia(A,B):
    if type(A) is int or type(A) is float:
        return abs(B-A)
    else:
        # distancia euclidea
        return math.sqrt(sum([(A[i]-B[i])**2 for i in range(len(A))]))

distancia((1,3),(2,5))
```

Out[2]: 2.23606797749979

```
In [3]: #Fuerza_Bruta
def Distancia_Fuerza_Bruta(L):
    # Inicializamos el valor con un valor muy alto
    mejor_distancia=1000000000
```

```

mejor_distancia = 10000000

A,B=(),()
for i in range (len(L)):
    for j in range (i+1,len(L)):
        d = distancia(L[i],L[j])
        if d < mejor_distancia:
            A,B=L[i],L[j]
            mejor_distancia = d
    return [A,B]

Distancia_Fuerza_Bruta(Lista_2D)

```

Out[3]: [(21591, 9918), (21598, 9912)]

```

In [4]: #Divide y venceras
def Distancia_Divide_Y_Venceras(L):
    # Si hay pocos elementos usamos fuerza bruta
    if len(L)<10:
        return Distancia_Fuerza_Bruta(L)

    LISTA_IZQ=sorted(L,key=lambda x: x[0])[:len(L)//2]
    LISTA_DER=sorted(L,key=lambda x: x[0])[len(L)//2:]

    PUNTOS_LISTA_IZQ = Distancia_Divide_Y_Venceras(LISTA_IZQ)
    PUNTOS_LISTA_DER = Distancia_Divide_Y_Venceras(LISTA_DER)

    # Mejora ordenar solo una vez la lista
    # ver que puntos estan tan cerca del limite como el mínimo de las
    # distancias de los pares de puntos elegidos por separados en cada uno de las listas
    return Distancia_Fuerza_Bruta(PUNTOS_LISTA_IZQ + PUNTOS_LISTA_DER)

#####
# Mejoras:
#####
def Distancia_Divide_Y_Venceras_Solo_Ordena_Una_Vez(L,bListaOrdenada):
    # Si hay pocos elementos usamos fuerza bruta
    if len(L)<10:
        return Distancia_Fuerza_Bruta(L)

```

```

    if (not bListaOrdenada):
        L=sorted(L,key=lambda x: x[0])
        bListaOrdenada=True

    LISTA_IZQ=L[:len(L)//2]
    LISTA_DER=L[len(L)//2:]

    PUNTOS_LISTA_IZQ = Distancia_Divide_Y_Venceras_Solo_Ordena_Una_Vez(LISTA_IZQ, bListaOrdenada)
a)
    PUNTOS_LISTA_DER = Distancia_Divide_Y_Venceras_Solo_Ordena_Una_Vez(LISTA_DER, bListaOrdenada)
a)

    # Mejora ordenar solo una vez la lista
    # ver que puntos estan tan cerca del limite como el mínimo de las
    # distancias de los pares de puntos elegidos por separados en cada uno de las listas
    return Distancia_Fuerza_Bruta(PUNTOS_LISTA_IZQ + PUNTOS_LISTA_DER)

@calcular_tiempo
def LANZA(L):
    L=sorted(L,key=lambda x: x[0])
    return Distancia_Divide_Y_Venceras(L)
@calcular_tiempo
def LANZA_MEJORA(L):
    L=sorted(L,key=lambda x: x[0])
    return Distancia_Divide_Y_Venceras_Solo_Ordena_Una_Vez(L,False)

SOL = LANZA(Lista_2D)
print(SOL)

SOL_MEJORADA = LANZA_MEJORA(Lista_2D)
print(SOL_MEJORADA)

```

Tiempo de ejecución para algoritmo: 0.0438804626464843750000000
 [(2, 7408), (2, 7408)]

Tiempo de ejecución para algoritmo: 0.0329108238220214843750000
 [(2, 7408), (2, 7408)]

In [5]: # Paseo_Por_El_Rio (Ejemplo para empresas de distribución)

```

TARIFAS = [
[0,5,4,3,999,999,999],
[999,0,999,2,3,999,11],
[999,999, 0,1,999,4,10],
[999,999,999, 0,5,6,9],
[999,999, 999,999,0,999,4],
[999,999, 999,999,999,0,3],
[999,999,999,999,999,999,0]
]

def Precios(TARIFAS):
    N=len(TARIFAS[0])

    PRECIOS=[ [9999]*N for i in range(N) ]
    RUTAS=[ [""]*N for i in range(N) ]

    for i in range(N-1):
        for j in range(i+1, N):
            MIN = TARIFAS[i][j]
            RUTAS[i][j]=i

            for k in range(i,j):
                if PRECIOS[i][k] + TARIFAS[k][j] < MIN:
                    MIN = min( MIN , PRECIOS[i][k] + TARIFAS[k][j])
                    RUTAS[i][j]=k
            PRECIOS[i][j]=MIN
    return PRECIOS, RUTAS

def calcular_ruta(RUTA, desde, hasta):
    if desde == hasta:
        return desde
    else:
        return str(calcular_ruta(RUTA, desde, RUTA[desde][hasta])) + ',' + str(RUTA[desde][hasta])

PRECIOS, RUTAS=Precios(TARIFAS)

```

```

PRECIOS, RUTAS=PRECIOS(TARIFAS)
print(PRECIOS)
print()
print(RUTAS)

```

```

print("\nLa ruta es:")
print(calcular_ruta(RUTAS, 0,6))

```

```

[[9999, 5, 4, 3, 8, 8, 11], [9999, 9999, 999, 2, 3, 8, 7], [9999, 9999, 9999, 1, 6, 4, 7], [999
9, 9999, 9999, 9999, 5, 6, 9], [9999, 9999, 9999, 9999, 9999, 999, 4], [9999, 9999, 9999, 9999,
9999, 9999, 3], [9999, 9999, 9999, 9999, 9999, 9999, 9999]]

```

```

[['', 0, 0, 0, 1, 2, 5], ['', '', 1, 1, 1, 3, 4], ['', '', '', 2, 3, 2, 5], ['', '', '', '', 3,
3, 3], ['', '', '', '', '', 4, 4], ['', '', '', '', '', 5], ['', '', '', '', '', '']]

```

La ruta es:
0,0,2,5

```

In [6]: #####
# Mejora:
#####
# Aporto como mejora función que retorna el precio de la ruta
def Obtener_Precio(PRECIOS, desde, hasta):
    return PRECIOS[desde][hasta]

print("\nEl precio de la ruta es:")
print(str(Obtener_Precio(PRECIOS, 0,6)))
print()

```

El precio de la ruta es:
11

```

In [7]: #####
# Mejora:
#####
# Aporto como mejora función que visualiza una Lista en formato matriz
import pandas as pd
def Visualizar_Matriz_Lista(titulo,LISTA):
    # Asignar nombre a columnas desde 0 hasta la longitud de la lista
    columnas=""

```

```

for i in range(len(LISTA)):
    columnas=columnas+str(i)

# Convierto la Lista pasada como parametro en un pandas dataframe
df_LISTA=pd.DataFrame(LISTA,columns=list(columnas))

# Visualizar el Pandas DataFrame
print(titulo)
print ()
print(df_LISTA)

Visualizar_Matriz_Lista ('Matriz de coste de ir de un punto a otro:',PRECIOS)
print ()
Visualizar_Matriz_Lista ('Matriz de rutas posibles desde un punto:',RUTAS)

```

Matriz de coste de ir de un punto a otro:

	0	1	2	3	4	5	6
0	9999	5	4	3	8	8	11
1	9999	9999	999	2	3	8	7
2	9999	9999	9999	1	6	4	7
3	9999	9999	9999	9999	5	6	9
4	9999	9999	9999	9999	9999	999	4
5	9999	9999	9999	9999	9999	9999	3
6	9999	9999	9999	9999	9999	9999	9999

Matriz de rutas posibles desde un punto:

	0	1	2	3	4	5	6
0	0	0	0	1	2	5	
1		1	1	1	3	4	
2			2	3	2	5	
3				3	3	3	
4					4	4	
5						5	
6							

