



Join GitHub today

Dismiss

GitHub is home to over 40 million developers working together to host and review code, manage projects, and build software together.

[Sign up](#)

Branch: master ▾

[03MAIR---Algoritmos-de-Optimizacion---2019](#) / [AG1](#) /[Find file](#)[Copy path](#)[WilliamSteveRodriguezVillamizar_AG1.ipynb](#)

wisrovi Creado con Colaboratory

ba14a93 36 seconds ago

[1 contributor](#)

719 lines (719 sloc) | 24.2 KB



Raw

Blame

History



AG1- Actividad Guiada 1

Nombre y Apellidos: William Steve Rodriguez Villamizar

Colab: https://colab.research.google.com/github/wisrovi/03MAIR---Algoritmos-de-Optimizacion--2019/blob/master/AG1/WilliamSteveRodriguezVillamizar_AG1.ipynb#scrollTo=cmrw2Nx-wnNu

Url: https://github.com/wisrovi/03MAIR---Algoritmos-de-Optimizacion--2019/blob/master/AG1/WilliamSteveRodriguezVillamizar_AG1.ipynb

```
In [0]: #Decorador de Ricardo Lebron
from functools import wraps
from time import time

def calcular_tiempo(f):
    @wraps(f)
    def cronometro(*args, **kwargs):
        t_inicial = time() #tomo la hora antes de ejecutar la funcion
        salida = f(*args, **kwargs)
        t_final = time() #tomo la hora despues de ejecutar la funcion
        print('Tiempo transcurrido (en segundos): {}'.format(t_final - t_inicial))
        return salida
    return cronometro
```

```
In [0]: import random
```

Torres de Hanoi (por recursividad):

```
In [3]: def torres_hanoi(N, desde=1, hasta=3):
        if N==1:
            print("Llevar desde " + str(desde) + " hasta " + str(hasta))
        else:
            torres_hanoi(N-1, desde, 6 - desde - hasta)
            print("Llevar desde " + str(desde) + " hasta " + str(hasta))
```

```

    print("Llevar desde " + str(desde) + " hasta " + str(hasta))
    torres_hanoi(N-1, 6 - desde - hasta, hasta)

@calcular_tiempo
def verMovimientos_JugarTorresHanoi(numero_discos):
    torres_hanoi(4)
    verMovimientos_JugarTorresHanoi(4)

```

```

Llevar desde 1 hasta 2
Llevar desde 1 hasta 3
Llevar desde 2 hasta 3
Llevar desde 1 hasta 2
Llevar desde 3 hasta 1
Llevar desde 3 hasta 2
Llevar desde 1 hasta 2
Llevar desde 1 hasta 3
Llevar desde 2 hasta 3
Llevar desde 2 hasta 1
Llevar desde 3 hasta 1
Llevar desde 2 hasta 3
Llevar desde 1 hasta 2
Llevar desde 1 hasta 3
Llevar desde 2 hasta 3
Tiempo transcurrido (en segundos): 0.0017056465148925781

```

Fibonacci (por recursividad):

```

In [4]: def Fibonacci(final, valor = 0, anterior = 0, listado = []):
        if valor == 0:
            listado.append(0)
            listado.append(1)
            Fibonacci(final, 1, 0, listado)
        else:
            if (valor + anterior) <= final:
                listado.append(valor + anterior)
                Fibonacci(final, valor + anterior, valor, listado)
            else:
                print(listado)

@calcular_tiempo
def VerMovimientosFibonacci(numeroFinalFibonacci):

```

```
def verNumerosFibonacci(numeroFinalFibonacci) :  
    Fibonacci(1000)
```

```
VerNumerosFibonacci(1000)
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987]
```

```
Tiempo transcurrido (en segundos): 0.00012993812561035156
```

Inicio de la actividad guiada:

```
In [0]: def quick_sort(A):  
        if len(A)<=1:  
            return A  
  
        if len(A)==2:  
            return [min(A), max (A)]  
  
        pivote = (A[0] + A[1] + A[2])/3  
        IZQ = []  
        DER = []  
  
        for i in A:  
            if i <= pivote:  
                IZQ.append(i)  
            else:  
                DER.append(i)  
        return quick_sort(IZQ) + quick_sort(DER)  
  
@calcular_tiempo  
def QS(A):  
    return quick_sort(A)
```

```
In [7]: A = [9187, 244, 4054, 9222, 8373, 4993, 5265, 5470, 4519, 7182, 2035, 3506, 4337, 7580, 25  
54, 2824, 8357, 4447, 7379]  
A = list(map(lambda x: random.randrange(1,10000), range(1,600)))  
  
print(QS(A))
```

```
Tiempo transcurrido (en segundos): 0.0013816356658935547
```

```
[29, 55, 62, 78, 79, 120, 122, 133, 138, 154, 167, 176, 194, 200, 206, 215, 245, 277, 285,
```

293, 327, 348, 369, 386, 387, 389, 393, 440, 481, 497, 546, 549, 564, 580, 597, 601, 603,
610, 612, 622, 640, 684, 689, 703, 703, 705, 711, 717, 741, 751, 764, 770, 771, 779, 797,
800, 809, 819, 823, 851, 855, 878, 887, 897, 913, 918, 923, 937, 944, 978, 994, 1020, 104
3, 1066, 1069, 1080, 1086, 1097, 1150, 1154, 1168, 1171, 1174, 1188, 1194, 1198, 1218, 122
3, 1227, 1298, 1343, 1357, 1390, 1427, 1432, 1438, 1444, 1463, 1469, 1520, 1585, 1595, 159
5, 1623, 1655, 1672, 1706, 1723, 1788, 1838, 1844, 1853, 1857, 1862, 1877, 1893, 1901, 190
8, 1914, 1934, 1979, 1984, 1990, 2003, 2059, 2063, 2063, 2070, 2080, 2090, 2181, 2184, 219
4, 2199, 2202, 2231, 2243, 2254, 2256, 2267, 2286, 2298, 2311, 2323, 2329, 2380, 2387, 241
7, 2438, 2444, 2459, 2504, 2582, 2603, 2623, 2624, 2625, 2636, 2668, 2675, 2678, 2697, 269
8, 2708, 2739, 2744, 2806, 2831, 2836, 2838, 2839, 2841, 2849, 2873, 2876, 2880, 2938, 297
3, 2987, 2988, 2991, 2991, 3067, 3071, 3104, 3118, 3137, 3155, 3163, 3255, 3259, 3261, 327
5, 3278, 3282, 3296, 3300, 3341, 3379, 3382, 3388, 3399, 3431, 3470, 3478, 3496, 3505, 352
6, 3526, 3557, 3629, 3633, 3649, 3659, 3689, 3756, 3779, 3815, 3823, 3845, 3861, 3867, 387
9, 3909, 3913, 3939, 3945, 3990, 3991, 3994, 4018, 4038, 4045, 4045, 4055, 4072, 4082, 408
4, 4087, 4089, 4109, 4115, 4128, 4153, 4154, 4169, 4203, 4219, 4231, 4238, 4244, 4268, 430
3, 4315, 4316, 4318, 4324, 4326, 4329, 4345, 4357, 4359, 4386, 4398, 4402, 4426, 4427, 443
4, 4436, 4459, 4466, 4468, 4539, 4555, 4555, 4560, 4614, 4621, 4622, 4625, 4689, 4706, 471
5, 4737, 4737, 4759, 4773, 4791, 4802, 4826, 4830, 4835, 4867, 4871, 4910, 4936, 4991, 500
8, 5020, 5037, 5037, 5110, 5114, 5173, 5193, 5196, 5227, 5241, 5254, 5266, 5286, 5317, 534
4, 5350, 5408, 5424, 5425, 5442, 5448, 5469, 5470, 5478, 5491, 5493, 5511, 5514, 5517, 555
8, 5593, 5597, 5612, 5623, 5648, 5676, 5676, 5720, 5761, 5775, 5803, 5816, 5840, 5856, 586
6, 5919, 5932, 5953, 5979, 5988, 5993, 5999, 6013, 6021, 6031, 6069, 6092, 6099, 6101, 611
8, 6132, 6134, 6143, 6146, 6148, 6163, 6173, 6178, 6200, 6207, 6249, 6250, 6274, 6278, 629
1, 6305, 6306, 6357, 6401, 6414, 6415, 6421, 6447, 6454, 6474, 6478, 6481, 6493, 6507, 653
3, 6548, 6549, 6550, 6565, 6572, 6574, 6587, 6589, 6597, 6622, 6650, 6651, 6658, 6662, 668
0, 6700, 6717, 6739, 6746, 6768, 6790, 6808, 6808, 6829, 6830, 6833, 6838, 6845, 6852, 690
4, 6922, 6924, 6969, 6977, 7010, 7034, 7050, 7071, 7076, 7095, 7099, 7126, 7139, 7141, 715
3, 7197, 7204, 7207, 7224, 7235, 7235, 7240, 7293, 7335, 7355, 7370, 7391, 7391, 7406, 741
7, 7419, 7422, 7426, 7440, 7452, 7460, 7466, 7468, 7486, 7512, 7518, 7553, 7565, 7567, 757
2, 7586, 7611, 7645, 7658, 7667, 7671, 7674, 7678, 7719, 7732, 7734, 7768, 7770, 7785, 780
3, 7844, 7848, 7852, 7861, 7882, 7884, 7905, 7915, 7922, 7947, 7952, 7964, 7985, 7997, 802
4, 8035, 8085, 8099, 8134, 8161, 8185, 8200, 8231, 8236, 8237, 8297, 8347, 8375, 8378, 839
0, 8418, 8421, 8432, 8447, 8469, 8477, 8490, 8505, 8520, 8529, 8560, 8593, 8627, 8638, 864
2, 8645, 8658, 8672, 8681, 8715, 8727, 8744, 8761, 8773, 8774, 8780, 8836, 8863, 8869, 891
5, 8915, 8948, 8957, 9000, 9008, 9033, 9047, 9061, 9067, 9098, 9102, 9104, 9106, 9112, 913
9, 9157, 9173, 9178, 9192, 9209, 9213, 9236, 9280, 9284, 9286, 9308, 9334, 9355, 9371, 937
7, 9407, 9409, 9448, 9451, 9472, 9483, 9514, 9623, 9633, 9643, 9650, 9664, 9665, 9665, 967
3, 9691, 9717, 9757, 9764, 9769, 9786, 9807, 9815, 9836, 9853, 9872, 9892, 9934, 9963, 997
0, 9987]

Cambio Monedas:

```
In [0]: @calcular_tiempo
def cambio_monedas(CANTIDAD, SISTEMA):
    SOLUCION = [0 for i in range(len(SISTEMA))]
    VALOR_ACUMULADO = 0

    for i in range(len(SISTEMA)):
        monedas = int((CANTIDAD - VALOR_ACUMULADO) / SISTEMA[i])
        SOLUCION[i] = monedas
        VALOR_ACUMULADO += monedas * SISTEMA[i]
        if VALOR_ACUMULADO == CANTIDAD:
            return SOLUCION

    return SOLUCION
```

```
In [9]: SISTEMA = [25, 10, 5, 1]
CANTIDAD = 234
solucion = cambio_monedas(CANTIDAD, SISTEMA)

print("Sistema: ", end="")
print(SISTEMA)

print("Solucion: ", end="")
print(solucion, end="")
print(" para una cantidad de: ", CANTIDAD)
```

Tiempo transcurrido (en segundos): 1.0967254638671875e-05

Sistema: [25, 10, 5, 1]

Solucion: [9, 0, 1, 4] para una cantidad de: 234

Reinas:

```
In [0]: #Algoritmo de vuelta atras para el problema de N reinas
def es_prometedora(solucion, etapa):
    for i in range(etapa + 1):
        if solucion.count(solucion[i]) > 1:
            return False

    #verificar si son individuales
```

```

#vertical si son individuales
for j in range(i+1, etapa + 1):
    if abs(i-j) == abs(solucion[i]-solucion[j]):
        return False
return True

def reinas(N, solucion = [], etapa=0):
    if len(solucion)==0:
        solucion = [0 for i in range(N)]

    for i in range(1, N+1):
        solucion[etapa] = i

        if es_prometedora(solucion, etapa):
            if etapa == N-1:
                print("\n\nLa solucion es: ", end="")
                print(solucion)

            else:
                reinas(N, solucion, etapa+1)

        else:
            None

    solucion[etapa] = 0

@calcular_tiempo
def BuscarPosicionReinasEnTablero(NumeroReinas):
    reinas(NumeroReinas)

```

In [11]: BuscarPosicionReinasEnTablero(8)

La solucion es: [1, 5, 8, 6, 3, 7, 2, 4]

La solucion es: [1, 6, 8, 3, 7, 4, 2, 5]

La solucion es: [1, 7, 4, 6, 8, 2, 5, 3]

La solucion es: [1, 7, 5, 8, 2, 4, 6, 3]

La solucion es: [2, 4, 6, 8, 3, 1, 7, 5]

La solucion es: [2, 5, 7, 1, 3, 8, 6, 4]

La solucion es: [2, 5, 7, 4, 1, 8, 6, 3]

La solucion es: [2, 6, 1, 7, 4, 8, 3, 5]

La solucion es: [2, 6, 8, 3, 1, 4, 7, 5]

La solucion es: [2, 7, 3, 6, 8, 5, 1, 4]

La solucion es: [2, 7, 5, 8, 1, 4, 6, 3]

La solucion es: [2, 8, 6, 1, 3, 5, 7, 4]

La solucion es: [3, 1, 7, 5, 8, 2, 4, 6]

La solucion es: [3, 5, 2, 8, 1, 7, 4, 6]

La solucion es: [3, 5, 2, 8, 6, 4, 7, 1]

La solucion es: [3, 5, 7, 1, 4, 2, 8, 6]

La solucion es: [3, 5, 8, 4, 1, 7, 2, 6]

La solucion es: [3, 6, 2, 5, 8, 1, 7, 4]

La solucion es: [3, 6, 2, 7, 1, 4, 8, 5]

La solucion es: [3, 6, 2, 7, 5, 1, 8, 4]

La solucion es: [3, 6, 4, 1, 8, 5, 7, 2]

La solucion es: [3, 6, 4, 2, 8, 5, 7, 1]

La solucion es: [3, 6, 8, 1, 4, 7, 5, 2]

La solucion es: [3, 6, 8, 1, 5, 7, 2, 4]

La solucion es: [3, 6, 8, 2, 4, 1, 7, 5]

La solucion es: [3, 7, 2, 8, 5, 1, 4, 6]

La solucion es: [3, 7, 2, 8, 6, 4, 1, 5]

La solucion es: [3, 8, 4, 7, 1, 6, 2, 5]

La solucion es: [4, 1, 5, 8, 2, 7, 3, 6]

La solucion es: [4, 1, 5, 8, 6, 3, 7, 2]

La solucion es: [4, 2, 5, 8, 6, 1, 3, 7]

La solucion es: [4, 2, 7, 3, 6, 8, 1, 5]

La solucion es: [4, 2, 7, 3, 6, 8, 5, 1]

La solucion es: [4, 2, 7, 5, 1, 8, 6, 3]

La solucion es: [4, 2, 8, 5, 7, 1, 3, 6]

La solucion es: [4, 2, 8, 6, 1, 3, 5, 7]

La solucion es: [4, 6, 1, 5, 2, 8, 3, 7]

La solucion es: [4, 6, 8, 2, 7, 1, 3, 5]

La solucion es: [4, 6, 8, 3, 1, 7, 5, 2]

La solucion es: [4, 7, 1, 8, 5, 2, 6, 3]

La solucion es: [4, 7, 3, 8, 2, 5, 1, 6]

La solucion es: [4, 7, 5, 2, 6, 1, 3, 8]

La solucion es: [4, 7, 5, 3, 1, 6, 8, 2]

La solucion es: [4, 8, 1, 3, 6, 2, 7, 5]

La solucion es: [4, 8, 1, 5, 7, 2, 6, 3]

La solucion es: [4, 8, 5, 3, 1, 7, 2, 6]

La solucion es: [5, 1, 4, 6, 8, 2, 7, 3]

La solucion es: [5, 1, 8, 4, 2, 7, 3, 6]

La solucion es: [5, 1, 8, 6, 3, 7, 2, 4]

La solucion es: [5, 2, 4, 6, 8, 3, 1, 7]

La solucion es: [5, 2, 4, 7, 3, 8, 6, 1]

La solucion es: [5, 2, 6, 1, 7, 4, 8, 3]

La solucion es: [5, 2, 8, 1, 4, 7, 3, 6]

La solucion es: [5, 3, 1, 6, 8, 2, 4, 7]

La solucion es: [5, 3, 1, 7, 2, 8, 6, 4]

La solucion es: [5, 3, 8, 4, 7, 1, 6, 2]

La solucion es: [5, 7, 1, 3, 8, 6, 4, 2]

La solucion es: [5, 7, 1, 4, 2, 8, 6, 3]

La solucion es: [5, 7, 2, 4, 8, 1, 3, 6]

La solucion es: [5, 7, 2, 6, 3, 1, 4, 8]

La solucion es: [5, 7, 2, 6, 3, 1, 8, 4]

La solucion es: [5, 7, 4, 1, 3, 8, 6, 2]

La solucion es: [5, 8, 4, 1, 3, 6, 2, 7]

La solucion es: [5, 8, 4, 1, 7, 2, 6, 3]

La solucion es: [6, 1, 5, 2, 8, 3, 7, 4]

La solucion es: [6, 2, 7, 1, 3, 5, 8, 4]

La solucion es: [6, 2, 7, 1, 4, 8, 5, 3]

La solucion es: [6, 3, 1, 7, 5, 8, 2, 4]

La solucion es: [6, 3, 1, 8, 4, 2, 7, 5]

La solucion es: [6, 3, 1, 8, 5, 2, 4, 7]

La solucion es: [6, 3, 5, 7, 1, 4, 2, 8]

La solucion es: [6, 3, 5, 8, 1, 4, 2, 7]

La solucion es: [6, 3, 7, 2, 4, 8, 1, 5]

La solucion es: [6, 3, 7, 2, 8, 5, 1, 4]

La solucion es: [6, 3, 7, 4, 1, 8, 2, 5]

La solucion es: [6, 4, 1, 5, 8, 2, 7, 3]

La solucion es: [6, 4, 2, 8, 5, 7, 1, 3]

La solucion es: [6, 4, 7, 1, 3, 5, 2, 8]

La solucion es: [6, 4, 7, 1, 8, 2, 5, 3]

La solucion es: [6, 8, 2, 4, 1, 7, 5, 3]

La solucion es: [7, 1, 3, 8, 6, 4, 2, 5]

La solucion es: [7, 2, 4, 1, 8, 5, 3, 6]

La solucion es: [7, 2, 6, 3, 1, 4, 8, 5]

La solucion es: [7, 3, 1, 6, 8, 5, 2, 4]

La solucion es: [7, 3, 8, 2, 5, 1, 6, 4]

La solucion es: [7, 4, 2, 5, 8, 1, 3, 6]

La solucion es: [7, 4, 2, 8, 6, 1, 3, 5]

La solucion es: [7, 5, 3, 1, 6, 8, 2, 4]

La solucion es: [8, 2, 4, 1, 7, 5, 3, 6]

La solucion es: [8, 2, 5, 3, 1, 7, 4, 6]

La solucion es: [8, 3, 1, 6, 2, 5, 7, 4]

© 2019 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Help](#)

[Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)