AG1- Actividad Guiada 1

Nombre y Apellidos: William Steve Rodriguez Villamizar

Url: https://colab.research.google.com/github/wisrovi/03MAIR-Algoritmos-de-Optimizacion/blob/main/AG1/ActividadGuiada1.ipynb

Torres Hanoi con divide y venceras y recursividad

```
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, list({1,2,3} - {desde, hasta})[0])
    print("Llevar desde " + str(desde) + " hasta " + str(hasta))
    torres_hanoi(N-1, list({1,2,3} - {desde, hasta})[0], hasta)
```

```
torres_hanoi(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
```

Devolucion cambio monedas

```
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


SISTEMA = [25, 10, 5, 1]
CANTIDAD = 37
solucion = cambio_monedas(CANTIDAD, SISTEMA)

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

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

```
Sistema: [25, 10, 5, 1]
Solucion: [1, 1, 0, 2] para una cantidad de:  37
```

Problema N reinas

```
#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
    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)

        for i in solucion:
          for j in range(len(solucion)):
```

```
            if j+1 == i:
              print("R", end=" ")
            else:
              print("-", end=" ")
          print()
      else:
        reinas(N, solucion, etapa+1)
    else:
      None

    solucion[etapa] = 0
```

```
reinas(8)
```

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


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


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


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

```
- - - R - - - -
- - - - - R - -
- - R - - - - -


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


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

Puntos con menor distancia

Lista random

```
import random
def GenerarLista(dimension=1, size = 10000, maximo=10000):
  if dimension == 1:
    return list(random.randrange(1,maximo) for x in range(1,size + 1))
  if dimension == 2:
    return list((random.randrange(1,maximo),random.randrange(1,maximo)) for x in range(1
  if dimension == 3:
    return list((random.randrange(1,maximo),random.randrange(1,maximo),random.randrange(

```

Distancia dos puntos (1, 2 y 3 dimensiones)

```
import math

def distancia_dos_puntos(punto1:tuple, punto2:tuple):
  # http://campusvirtual.cua.uam.mx/pdfs/paea/18o/tm/tema5_cont_c.pdf
  # https://www.varsitytutors.com/hotmath/hotmath_help/spanish/topics/distance-formula-i

  if len(punto1) ==  len(punto2):
    if len(punto1) == 1:
      return abs(punto2[0] - punto1[0])
    elif len(punto1) == 2:
      return abs(math.sqrt( pow(punto2[0]-punto1[0],2) + pow(punto2[1]-punto1[1],2) ))
    elif len(punto1) == 3:
```

```
        return abs(math.sqrt(  pow(punto2[0]-punto1[0],2)  +  pow(punto2[1]-punto1[1],2)
      else:
        return None
    else:
      return None
```

Buscar menor distancia

```
def BuscarMenorDistancia(lista:list):
  if len(lista) >= 3:
    distancia_menor = 10e100
    puntos_distancia_menor = [0,0]

    for i in range(len(lista)):
      for j in range(i+1, len(lista)):
        A, B = lista[i], lista[j]
        if not isinstance(lista[0], tuple):
          A, B = (A, ), (B,  )

        AB = distancia_dos_puntos(A, B)

        if AB < distancia_menor:
          distancia_menor, puntos_distancia_menor = AB, [lista[i], lista[j]]

    return puntos_distancia_menor
```

Test

```
listado = GenerarLista(dimension=1, size=10, maximo=5000)

print(listado)

BuscarMenorDistancia(listado)
```

```
[1844, 4666, 2767, 2512, 282, 1662, 4943, 4524, 1133, 4811]
[4943, 4811]
```