Actividad Guiada 1 de Algoritmos de Optimizacion

Nombre: Javier Canales Navarrete

https://colab.research.google.com/drive/1W4pvcf4aY_60-q098cfcQet-wUyb7nDP?usp=sharing

https://github.com/Javicana/03MAIR-Algoritmos-de-Optimizacon-2021

```
#Torres de Hanoi - Divide y venceras
################################################################################


################################################################
def Torres_Hanoi(N, desde, hasta):
  #N - Nº de fichas
  #desde - torre inicial
  #hasta - torre final

  if N == 1 :
    print("Lleva la ficha desde " + str(desde) + " hasta " + str(hasta))

  else:
    Torres_Hanoi(N-1, desde, 6-desde-hasta)
    print("Lleva la ficha desde " + str(desde) + " hasta " + str(hasta))
    Torres_Hanoi(N-1, 6-desde-hasta,  hasta)

Torres_Hanoi(3, 1, 3)
################################################################
```

```
     Lleva la ficha desde 1 hasta 3
     Lleva la ficha desde 1 hasta 2
     Lleva la ficha desde 3 hasta 2
     Lleva la ficha desde 1 hasta 3
     Lleva la ficha desde 2 hasta 1
     Lleva la ficha desde 2 hasta 3
     Lleva la ficha desde 1 hasta 3
```

```
#Cambio de monedas - Técnica voraz
################################################################################
SISTEMA = [12, 5 ,2, 1]
################################################################
def cambio_monedas(CANTIDAD,SISTEMA):
#....
  SOLUCION = [0]*len(SISTEMA)
  ValorAcumulado = 0

  for i,valor in enumerate(SISTEMA):
    monedas = (CANTIDAD-ValorAcumulado)//valor
    SOLUCION[i] = monedas
    ValorAcumulado += monedas*valor

    if CANTIDAD == ValorAcumulado:
      return SOLUCION

  print("No es posible encontrar solucion")
cambio_monedas(15,SISTEMA)


################################################################
```

```
     [1, 0, 1, 1]
```

```
#N Reinas - Vuelta Atrás()
################################################################################

#Verifica que en la solución parcial no hay amenzas entre reinas
```

```
#Verifica que en la solución parcial no hay amenaza entre reinas
#################################################################
def es_prometedora(SOLUCION,etapa):
#################################################################
  #print(SOLUCION)
  #Si la solución tiene dos valores iguales no es valida => Dos reinas en la misma fila
  for i in range(etapa+1):
    #print("El valor " + str(SOLUCION[i]) + " está " +  str(SOLUCION.count(SOLUCION[i])) + " veces")
    if SOLUCION.count(SOLUCION[i]) > 1:
      return False

    #Verifica las diagonales
    for j in range(i+1, etapa +1 ):
      #print("Comprobando diagonal de " + str(i) + " y " + str(j))
      if abs(i-j) == abs(SOLUCION[i]-SOLUCION[j]) : return False
  return True

#Traduce la solución al tablero
#################################################################
def escribe_solucion(S):
#################################################################
  n = len(S)
  for x in range(n):
    print("")
    for i in range(n):
      if S[i] == x+1:
        print(" X " , end="")
      else:
        print(" - ", end="")



#Proceso principal de N-Reinas
#################################################################
def reinas(N, solucion=[],etapa=0):
#################################################################
### ....
  if len(solucion) == 0:         # [0,0,0...]
    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(solucion)
      else:
        reinas(N, solucion, etapa+1)
    else:
      None

  solucion[etapa] = 0

reinas(4,solucion=[],etapa=0)
```

```
[2, 4, 1, 3]
[3, 1, 4, 2]
```

```
escribe_solucion([3, 1, 4, 2])
```

```
 -  X  -  -
 -  -  -  X
 X  -  -  -
 -  -  X  -
```