

Actividad Guiada 1 de Algoritmos de Optimizacion

Nombre: Mayra Pullupaxi

https://colab.research.google.com/drive/1qEXfCHaa8G_NYJvCiH4KGpO26hHUGPU?usp=sharing

https://github.com/MayAlejita/03MIAR_Algoritmos_de_Optimizacion

```
import math
float("inf")

inf

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

#####
def Torres_Hanoi(N, desde, hasta):
    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
#####

#####
def cambio_monedas(CANTIDAD, SISTEMA):
    SOLUCION = [0]*len(SISTEMA)
    valor_acumulado = 0

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

```

return SOLUCION

print("No es posible encontrar solucion")

SISTEMA = [25,10,5,1]
CANTIDAD = 26
cambio_monedas(CANTIDAD, SISTEMA)
#####

[1, 0, 0, 1]

#N Reinas - Vuelta Atrás()
#####

#Verifica que en la solución parcial no hay amenazas entre reinas
#####
def es_prometedora(SOLUCION,etapa):
#####
    # print(SOLUCION)
    #Si la solución tiene dos valores iguales no es valida => Dos reinas en la misma
    for i in range(etapa+1):
        #print("El valor " + str(SOLUCION[i]) + " está " + str(SOLUCION.count(SOLUCION
        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:
        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([2, 4, 1, 3])
```

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

✓ 0 s se ejecutó 15:53

