Algoritmos_Adrián_Pérez_Portero_AG1

February 20, 2024

1 Actividad Guiada 1 de Algoritmos de Optimización

Adrián Pérez Portero

https://colab.research.google.com/drive/1y-qe7pE_w6A3U2CtoeJngX6rqGTcRb8a?usp=sharing

https://github.com/adrianperezp/Alg_Opt/tree/master/Actividades%20Guiadas/AG1

[ ]:
```python
# Torres de Hanoi

def torres_hanoy(N, desde, hasta):
  if N == 1:
    print("Llevar desde " + str(desde) + " hasta " + str(hasta))
  else:
    torres_hanoy(N-1, desde, 6 - desde - hasta)
    print("Llevar desde " + str(desde) + " hasta " + str(hasta))
    torres_hanoy(N-1, 6 - desde - hasta, hasta)

torres_hanoy(4, 1, 3)
```

```
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
```

[ ]:
```python
# Cambio de monedas

def cambio_monedas(CANTIDAD, SISTEMA):
```

```
  print("SISTEMA:")
  print(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]
cambio_monedas(27, SISTEMA)
```

```
SISTEMA:
[25, 10, 5, 1]
```

[ ]:
```
[1, 0, 0, 2]
```

[ ]:
```
# Proceso principal de N-Reinas
def reinas(N, solucion = [], etapa = 0):
    #N         - Tamaño del tablero
    # solucion - Solucion parcial
    # etapa    - nº de reinas colocadas en la solución parcial

    # Inicializa la solución: una lista con ceros
    if len(solucion) == 0:
        solucion = [0 for i in range(N)]

    # Recorremos todas las reinas
    for i in range(1, N + 1):
        solucion[etapa] = i

        #print(solucion)
        if es_prometedora(solucion, etapa):
            if etapa == N - 1:
                print("\n\nLa solución es:")
                print(solucion)
                escribe_solucion(solucion)
            else:
                #print("Es prometedora\n################")
                reinas(N, solucion, etapa + 1)
        else:
          #print("NO PROMETEDORA\n################")
```

```
            None

        solucion[etapa] = 0

def es_prometedora(SOLUCION, etapa):
    #print(SOLUCION)
    # Si la solución tiene dos valores iguales no es válida => 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
```

[ ]:
```
# Viaje por el río - Programación dinámica
#########################################################################

TARIFAS = [
    [0, 5, 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]
]

# 999 se puede sustituir por float ("inf")

#########################################################################
def Precios(TARIFAS):
    # Total de Nodos
    N = len(TARIFAS[0])

    # Inicialización de la tabla de precios
    PRECIOS = [[9999] * N for i in [9999] * N]
    RUTA = [[""] * N for i in [""] * N]

    for i in range(N - 1):
        for j in range(i + 1, N):
            MIN = TARIFAS[i][j]
            RUTA[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])
                    RUTA[i][j] = k
                PRECIOS[i][j] = MIN
    return PRECIOS, RUTA

def calcular_ruta(RUTA, desde, hasta):
    if desde == hasta:
        #print("Ir a: " + str(desde))
        return desde
    else:
        return str(calcular_ruta(RUTA, desde, RUTA[desde][hasta])) +␣
 ↪str(RUTA[desde][hasta])

RUTA = [
    ['', 0, 0, 0, 1, 2, 5],
    ['', '', 1, 1, 1, 3, 4],
    ['', '', '', 2, 3, 2, 5],
    ['', '', '', '', 3, 3, 3],
    ['', '', '', '', '', 4, 4],
    ['', '', '', '', '', '', 5],
    ['', '', '', '', '', '', '']
]
print("\nLa ruta es:")
calcular_ruta(RUTA, 0, 6)
```

```
La ruta es:
```

[ ]: '0025'