Algoritmos - Actividad Guiada 1

Nombre: Esmarlin Julissa Moreno Nivar

GitHub: https://github.com/julissrock/03MIAR-Algoritmos-de-Optimizacion/blob/main/Algoritmos_AG1.ipynb

Torres de Hanoi con Divide y vencerás

In [9]:

```
def Torres_Hanoi(N, desde, hasta):
  if N ==1 :
    print("Lleva la ficha " ,desde , " hasta ", hasta )

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


Torres_Hanoi(3, 1 , 3)
```

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

In [10]:

```
#Sucesión_de_Fibonacci
#https://es.wikipedia.org/wiki/Sucesi%C3%B3n_de_Fibonacci
#Calculo del termino n-simo de la suscesión de Fibonacci
def Fibonacci(N:int):
  if N < 2:
    return 1
  else:
    return Fibonacci(N-1)+Fibonacci(N-2)

Fibonacci(5)
```

Out[10]:

```
8
```

Devolución de cambio por técnica voraz

In [11]:

```
def cambio_monedas(N, SM):
  SOLUCION = [0]*len(SM)   #SOLUCION = [0,0,0,0,..]
  ValorAcumulado = 0

  for i,valor in enumerate(SM):
    monedas =  (N-ValorAcumulado)//valor
    SOLUCION[i] = monedas
    ValorAcumulado = ValorAcumulado + monedas*valor
    if ValorAcumulado == N:
      return SOLUCION


cambio_monedas(15,[25,10,5,1])
```

Out[11]:
```
[0, 1, 1, 0]
```

N-Reinas por técnica de vueta atrás

In [12]:
```
def escribe(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="")


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(SOLUCIO
    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



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)
        #escribe(solucion)
        print()
      else:
        reinas(N, solucion, etapa+1)
    else:
      None

    solucion[etapa] = 0

reinas(8)
```

```
[1, 5, 8, 6, 3, 7, 2, 4]

[1, 6, 8, 3, 7, 4, 2, 5]

[1, 7, 4, 6, 8, 2, 5, 3]

[1, 7, 5, 8, 2, 4, 6, 3]

[2, 4, 6, 8, 3, 1, 7, 5]

[2, 5, 7, 1, 3, 8, 6, 4]

[2, 5, 7, 4, 1, 8, 6, 3]

[2, 6, 1, 7, 4, 8, 3, 5]

[2, 6, 8, 3, 1, 4, 7, 5]

[2, 7, 3, 6, 8, 5, 1, 4]

[2, 7, 5, 8, 1, 4, 6, 3]

[2, 8, 6, 1, 3, 5, 7, 4]

[3, 1, 7, 5, 8, 2, 4, 6]

[3, 5, 2, 8, 1, 7, 4, 6]

[3, 5, 2, 8, 6, 4, 7, 1]

[3, 5, 7, 1, 4, 2, 8, 6]

[3, 5, 8, 4, 1, 7, 2, 6]

[3, 6, 2, 5, 8, 1, 7, 4]

[3, 6, 2, 7, 1, 4, 8, 5]

[3, 6, 2, 7, 5, 1, 8, 4]

[3, 6, 4, 1, 8, 5, 7, 2]

[3, 6, 4, 2, 8, 5, 7, 1]

[3, 6, 8, 1, 4, 7, 5, 2]

[3, 6, 8, 1, 5, 7, 2, 4]

[3, 6, 8, 2, 4, 1, 7, 5]

[3, 7, 2, 8, 5, 1, 4, 6]

[3, 7, 2, 8, 6, 4, 1, 5]

[3, 8, 4, 7, 1, 6, 2, 5]

[4, 1, 5, 8, 2, 7, 3, 6]

[4, 1, 5, 8, 6, 3, 7, 2]

[4, 2, 5, 8, 6, 1, 3, 7]

[4, 2, 7, 3, 6, 8, 1, 5]
```

```
[4, 2, 7, 3, 6, 8, 5, 1]

[4, 2, 7, 5, 1, 8, 6, 3]

[4, 2, 8, 5, 7, 1, 3, 6]

[4, 2, 8, 6, 1, 3, 5, 7]

[4, 6, 1, 5, 2, 8, 3, 7]

[4, 6, 8, 2, 7, 1, 3, 5]

[4, 6, 8, 3, 1, 7, 5, 2]

[4, 7, 1, 8, 5, 2, 6, 3]

[4, 7, 3, 8, 2, 5, 1, 6]

[4, 7, 5, 2, 6, 1, 3, 8]

[4, 7, 5, 3, 1, 6, 8, 2]

[4, 8, 1, 3, 6, 2, 7, 5]

[4, 8, 1, 5, 7, 2, 6, 3]

[4, 8, 5, 3, 1, 7, 2, 6]

[5, 1, 4, 6, 8, 2, 7, 3]

[5, 1, 8, 4, 2, 7, 3, 6]

[5, 1, 8, 6, 3, 7, 2, 4]

[5, 2, 4, 6, 8, 3, 1, 7]

[5, 2, 4, 7, 3, 8, 6, 1]

[5, 2, 6, 1, 7, 4, 8, 3]

[5, 2, 8, 1, 4, 7, 3, 6]

[5, 3, 1, 6, 8, 2, 4, 7]

[5, 3, 1, 7, 2, 8, 6, 4]

[5, 3, 8, 4, 7, 1, 6, 2]

[5, 7, 1, 3, 8, 6, 4, 2]

[5, 7, 1, 4, 2, 8, 6, 3]

[5, 7, 2, 4, 8, 1, 3, 6]

[5, 7, 2, 6, 3, 1, 4, 8]

[5, 7, 2, 6, 3, 1, 8, 4]

[5, 7, 4, 1, 3, 8, 6, 2]

[5, 8, 4, 1, 3, 6, 2, 7]

[5, 8, 4, 1, 7, 2, 6, 3]
```

```
[6, 1, 5, 2, 8, 3, 7, 4]

[6, 2, 7, 1, 3, 5, 8, 4]

[6, 2, 7, 1, 4, 8, 5, 3]

[6, 3, 1, 7, 5, 8, 2, 4]

[6, 3, 1, 8, 4, 2, 7, 5]

[6, 3, 1, 8, 5, 2, 4, 7]

[6, 3, 5, 7, 1, 4, 2, 8]

[6, 3, 5, 8, 1, 4, 2, 7]

[6, 3, 7, 2, 4, 8, 1, 5]

[6, 3, 7, 2, 8, 5, 1, 4]

[6, 3, 7, 4, 1, 8, 2, 5]

[6, 4, 1, 5, 8, 2, 7, 3]

[6, 4, 2, 8, 5, 7, 1, 3]

[6, 4, 7, 1, 3, 5, 2, 8]

[6, 4, 7, 1, 8, 2, 5, 3]

[6, 8, 2, 4, 1, 7, 5, 3]

[7, 1, 3, 8, 6, 4, 2, 5]

[7, 2, 4, 1, 8, 5, 3, 6]

[7, 2, 6, 3, 1, 4, 8, 5]

[7, 3, 1, 6, 8, 5, 2, 4]

[7, 3, 8, 2, 5, 1, 6, 4]

[7, 4, 2, 5, 8, 1, 3, 6]

[7, 4, 2, 8, 6, 1, 3, 5]

[7, 5, 3, 1, 6, 8, 2, 4]

[8, 2, 4, 1, 7, 5, 3, 6]

[8, 2, 5, 3, 1, 7, 4, 6]

[8, 3, 1, 6, 2, 5, 7, 4]

[8, 4, 1, 3, 6, 2, 7, 5]
```