Carlos Somoza Martínez:
https://colab.research.google.com/drive/1TT-dD3Oir4xvly1Kw9UTTcooUYLOhX2J?usp=sharing

Actividad guiada

Divide y vencerás

- Comprobar que es un caso sencillo
- Ser capaces de recombinar subproblemas para dar solución al problema general
- Problemas disjuntos

```
#Torres Hanoi

def torres_hanoi(N,desde,hasta):
  pivote=6
  if(N==1):
    print("Leva la ficha desde "+str(desde)+" hasta "+str(hasta))

  else:
    torres_hanoi(N-1,desde,pivote-desde-hasta)
    print("Lleva la ficha desde " +str(desde)+" hasta "+str(hasta))
    torres_hanoi(N-1,pivote-desde-hasta,hasta)
torres_hanoi(3,1,3)
```

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

Técnica voraz

- Elige en cada etapa la solución óptima rechazando todas las demás
- Valida la factabilidad
- Criterio que compruebe que hemops finalizado

```
#Cambio de monedas
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 = ValorAcumulado + monedas*valor

    if CANTIDAD == ValorAcumulado:
      return SOLUCION

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

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

Backtracking

Nos facilita la posibilidad de hacer **podas**

```
#Problema 4 reinas

#Verifica que en la solución parcial no hay amenzas 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]
```