Branch: master ▾ 03MAIR-Algoritmos-de-optimizacion / AG1 / AG1_Isabel_Vazquez.ipynb

Find file Copy path

404isabel Creado con Colaboratory 712a85b a minute ago

1 contributor

242 lines (242 sloc) | 6.71 KB

Raw Blame History

AG1 - Actividad Guiada 1

Isabel Vázquez Trigás

https://github.com/404isabel/03MAIR-Algoritmos-de-optimizacion/tree/master/AG1

In [0]:
```
from time import time
def calcular_tiempo(f):

    def wrapper(*args, **kwargs):
        inicio = time()
        resultado = f(*args, **kwargs)
        tiempo = time() - inicio
        print("Tiempo de ejecución para algoritmo: "+str(tiempo))
        return resultado

    return wrapper

@calcular_tiempo
def QS(A):
  return quick_sort(A)
```

In [20]:
```
#quick_sort

A = [9187, 244, 4054, 9222, 8373, 4993, 5265, 5470, 4519, 7182, 2035, 3506, 4337, 7580, 2554, 2
824, 8357, 4447, 7379]


def quick_sort(A):
  if len(A)==1:
    return A
  elif len(A)==2:
    return [min(A),max(A)]

  izq=[]
  der=[]
  #Meter todo esto de abajo en un else
```

```
  #mejorable la elección del pivote (máx-min)/2. Igual calculando de una manera eficiente la me
diana. (La mediana recorre la lista completa, y eso incrementa el valor de la complejidad)

  pivote = (A[0]+ A[1]+ A[2])/3

  #Mirar si puedo hacerlo con un while, para eliminar iteraciones
  for i in A:
    if i<pivote:
      izq.append(i)
    else:
      der.append(i)

  return quick_sort(izq)+quick_sort(der)

quick_sort(A)
print(QS(A))
```

```
Tiempo de ejecución para algoritmo: 2.7179718017578125e-05
[244, 2035, 2554, 2824, 3506, 4054, 4337, 4447, 4519, 4993, 5265, 5470, 7182, 7379, 7580, 8357,
8373, 9187, 9222]
```

In [48]:

```
@calcular_tiempo
def cambio_monedas(cantidad,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



sistema=[25,10,5,1]
#Devuelve el número de monedas de cada posición
cambio_monedas(77,sistema)
```

```
cambio_monedas(77,sistema)
```

```
[25, 10, 5, 1]
Tiempo de ejecución para algoritmo: 0.003053903579711914
```

Out[48]: [3, 0, 0, 2]

In [0]:

```
N=4

solucion = [0 for i in range(N)]

etapa=0

def es_prometedora(solucion,etapa):
  for i in range(etapa+1):
    if solucion.count(solucion[i])>1:
      return False

    #Verificar las diagonales
    for j in range(i+1,etapa+1):
      if abs(i-j) == abs(solucion[i]-solucion[j]):
        return False
  return True

def escribe(S):
  n=len(S)
  for x in range(n):
    print("")
    for i


def reinas(N, solucion, etapa):

  if es_prometedora(solucion,etapa):
    if etapa == N-1:
      print(solucion)
      escribe(solucion)
    else:
      #es prometedora
      reinas(N,solucion, etapa+1)
  else:
```

```
    #no es prometedora
    None
```

In [0]: