Branch: master ▾ 03MAIR-Algoritmos-de-Optimizacion-2019 / AG1 / AG1 - Luis Fauré Navarro.ipynb

Find file Copy path

lfauren Creado mediante Colaboratory 3f62e5d 8 minutes ago

1 contributor

252 lines (252 sloc) | 7.13 KB

Raw Blame History

AG1 - Actividad 1 Luis Fauré Navarro https://github.com/lfauren/03MAIR-Algoritmos-de-Optimizacion-2019/tree/master/AG1

In [0]:
```
from time import time
#Función para calcular el tiempo de ejecución
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
```

In [59]:
```
# QuickSort
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
  if len(A) == 2:
    return [min(A),max(A)]
  pivote = (A[0] + A[1] +A[2])/3
  IZQ = []
  DER = []
  for i in A:
    if i<=pivote:
      IZQ.append(i)
    else:
      DER.append(i)

  return quick_sort(IZQ)+quick_sort(DER)
```

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


print(QS(A))
```

```
Tiempo de ejecución para algoritmo: 4.982948303222656e-05
[244, 2035, 2554, 2824, 3506, 4054, 4337, 4447, 4519, 4993, 5265, 5470, 7182, 7379, 7580, 8357,
8373, 9187, 9222]
```

In [68]:
```
SISTEMA=[25,10,5,1]
@calcular_tiempo
def cambio_monedas(CANTIDAD,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

cambio_monedas (234, SISTEMA)
```

```
Tiempo de ejecución para algoritmo: 9.298324584960938e-06
```

Out[68]:
```
[9, 0, 1, 4]
```

In [80]:
```
# 4 reinas

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
    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 in range(n):
      if solucion[i] == x+1:
        print(" X ", end="")
      else:
        print(" - ", end="")

def reinas(N,solucion,etapa):
  for i in range(1,N+1):
    solucion[etapa] = i
    if es_prometedora(solucion,etapa):
      if etapa == N-1:
        print("\n\nLa solución es:")
        print(solucion)
        escribe(solucion)
      else:
        reinas(N,solucion,etapa+1)
    else:
      None

    solucion[etapa] = 0

@calcular_tiempo
def rein(N, solucion, etapa):
  return reinas(N, solucion, etapa)

rein(N, solucion, etapa)
```

```
La solución es:
[2, 4, 1, 3]
```

```
[2, 4, 1, 3]

 -  -  X  -
 X  -  -  -
 -  -  -  X
 -  X  -  -

La solución es:
[3, 1, 4, 2]

 -  X  -  -
 -  -  -  X
 X  -  -  -
 -  -  X  - Tiempo de ejecución para algoritmo: 0.007338285446166992
```