Branch: master ▾ 03MAIR---Algoritmos-de-optimizacion / AG1 / AG1_Víctor_Cebrián.ipynb

Find file Copy path

vcebrian Creado con Colaboratory 2c083d7 a minute ago

1 contributor

257 lines (257 sloc) | 7.26 KB

Raw Blame History

AG - Actividad guiada 1

Víctor Cebrián

URL Git: https://github.com/vcebrian/03MAIR---Algoritmos-de-optimizacion/tree/master/AG1

In [0]:
```
#Decorador para calcular tiempos de ejecución
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 [67]:
```
#Algoritmo con la técnica divide y venceras - 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)]

  pivote = (min(A) + max(A))/2
  Izq=[]
  Der=[]

  for i in A:
    if i<pivote:
      Izq.append(i)
```

```
      Izq.append(i)
    else:
      Der.append(i)

  return quick_sort(Izq) + quick_sort(Der)

quick_sort(A)

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

print(QS(A))
```

```
Tiempo de ejecución para algoritmo: 4.220008850097656e-05
[244, 2035, 2554, 2824, 3506, 4054, 4337, 4447, 4519, 4993, 5265, 5470, 7182, 7379, 7580, 8357,
8373, 9187, 9222]
```

In [38]:
```
#Algoritmo voraz - Calculo cambio monedas
SISTEMA = [25,10,5,1]

@calcular_tiempo
def cambio_monedas(C, SISTEMA):
  SOLUCION = [0 for i in range(len(SISTEMA))]
  VALOR_ACUMULADO = 0

  for i in range(len(SISTEMA)):
    monedas = int((C-VALOR_ACUMULADO)/SISTEMA[i])
    SOLUCION[i] = monedas
    VALOR_ACUMULADO += monedas*SISTEMA[i]
    if C==VALOR_ACUMULADO:
      return SOLUCION

cambio_monedas(99, SISTEMA)
```

```
Tiempo de ejecución para algoritmo: 8.58306884765625e-06
```

Out[38]:
```
[3, 2, 0, 4]
```

In [68]:
```
#Algoritmo con la técnica vuelta atrás - Reinas
```

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

    #verifica 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 in range(n):
      if S[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

N=4
```

```
SOLUCION = [0 for i in range(N)]

reinas(N, SOLUCION, 0)
```

```
La solución es:
[2, 4, 1, 3]

 .  .  X  .
 X  .  .  .
 .  .  .  X
 .  X  .  .

La solución es:
[3, 1, 4, 2]

 .  X  .  .
 .  .  .  X
 X  .  .  .
 .  .  X  .
```