



 mcollada / 03MAIR-Algoritmos-de-optimizacion

 Watch


0

 Star

0

 Fork

0

 Code

 Issues 0

 Pull requests 0

 Projects 0

 Insights

Join GitHub today

GitHub is home to over 31 million developers working together to host and review code, manage projects, and build software together.

Sign up

Dismiss

Branch: master ▾

03MAIR-Algoritmos-de-optimizacion / AG1 / AG1\_MiguelAngelSotoCollada.ipynb

Find file

Copy path

 mcollada Creado con Colaboratory

3b12efc 3 minutes ago

1 contributor

240 lines (240 sloc) | 6.6 KB



Raw

Blame

History



AG1 - Actividad Guiada 1 Miguel Angel Soto Collada

<https://github.com/mcollada/03MAIR-Algoritmos-de-optimizacion/tree/master/AG1>

In [99]: *#quick\_sort*

```
A= [9187,244,4054,9222,8373,4993,5265,5470,4519,7182,2035,3506,4337,7500,2554,2824,8357,4447,7379]
```

```
def quick_sort (A):  
    if len(A)==1:  
        return A  
    if len(A)==2:  
        return [min(A), max(A)]  
  
    IZQ=[]  
    DER=[]  
  
    pivote=(A[0]+A[1]+A[2])/3  
  
    for i in A:  
        if i<pivote :  
            IZQ.append(i)  
        else:  
            DER.append(i)  
  
    return quick_sort (IZQ) + quick_sort (DER)  
  
print ("Quick_Sort: Ordenar de menor a mayor")  
print(quick_sort (A))
```

```
Quick_Sort: Ordenar de menor a mayor  
[244, 2035, 2554, 2824, 3506, 4054, 4337, 4447, 4519, 4993, 5265, 5470, 7182, 7379, 7500, 8357,  
8373, 9187, 9222]
```

In [100]: *#Algoritmo Voraz - Cambio de Moneda*

```
def ObtenerCambio (cantidad) :
    sistema = [25, 10, 5, 1]
    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 cantidad==valor_acumulado:
        return solucion
```

ObtenerCambio (77)

Out[100]: [3, 0, 0, 2]

```
In [115]: #Algoritmo de vuelta atrás
#Problema a resolver Reinas ajedrez
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 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):
    print('llega')
    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:
                #print("Es prometedora\n#####")
                reinas(n,solucion,etapa+1)
        else:
            #print("No prometedora\n#####")
            None

    solucion[etapa]=0

reinas(n,solucion,etapa)

```

La solución es:  
[2, 4, 1, 3]  
llega

```

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

```

La solución es:  
[2, 4, 1, 3]

[3, 1, 4, 2]  
llega

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

