	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Martínez Quintana Marco Antonio

Asignatura: Estructuras de Datos y Algoritmos I

Grupo: 17

No de Práctica(s): 11

Integrante(s): De León Arias Emiliano

No. de Equipo de cómputo empleado: 37

No. de Lista o Brigada: 13

Semestre: 2020-2

Fecha de entrega:14 abril 2020

Observaciones: CALIFICACIÓN: _____

Objetivo

El objetivo de esta guía es implementar, al menos, dos enfoques de diseño (estrategias) de algoritmos y analizar las implicaciones de cada uno de ellos.

Introducción

Como ya hemos observado en las anteriores prácticas, Python nos permite realizar varias operaciones mediante funciones propias de este lenguaje de programación. En esta práctica no solo seguiremos conociendo mas sobre estas funciones, si no también encontraremos diversas estrategias que nos ayudara a realizar algoritmos para poder resolver algunos problemas.

Una de las estrategias es la fuerza bruta, la cual consiste en encontrar todas las soluciones posibles para un problema en particular. De esa estrategia se deriva una conocida como Algoritmos ávidos, los cuales toman decisiones en un orden especifico.

Existen otras estrategias como la programación dinámica o incremental o la famosa divide y vencerás.

Por último seguiremos analizando como graficar con este lenguaje de programación y las funciones que esta incluye.

Desarrollo

Código:

```
#Fuerza Bruta
from string import ascii_letters,digits
from itertools import product
caracteres=ascii_letters+digits
def buscador(con):
    archivo=open("combinaciones.txt","w")

    if 3<=len(con)<=4:
        for i in range(3,5):
            for comb in product(caracteres,repeat=i):
                prueba="".join(comb)
                archivo.write(prueba+ "\n")
                if prueba==con:
                    print("Tu contraseña es {}".format(prueba))
                    archivo.close()
                    break
    else:
        print("Ingresa una contraseña que contenga de 3 a 4 caracteres")

from time import time
t0=time()
con="Hol4"
buscador(con)
print("Tiempos de ejecucion {}".format(round(time()-t0,6)))


#ALgoritmos avidos

def cambio(cantidad,denominaciones):
    resultado=[]
    while(cantidad>0):
        if(cantidad>=denominaciones[0]):
            num=cantidad//denominaciones[0]
            cantidad=cantidad-(num*denominaciones[0])
            resultado.append([denominaciones[0],num])
        denominaciones=denominaciones[1:]
    return resultado

print(cambio(1000,[500,200,100,50,20,5,1]))
print(cambio(500,[500,200,100,50,20,5,1]))
```

```
print(cambio(300,[50,20,5,1]))
print(cambio(200,[5]))
print(cambio(98,[50,20,5,1]))


#Bottom up

def fibonacci_iterativo_v1(numero):
    f1=0
    f2=1
    tmp=0
    for i in range(1,numero-1):
        tmp=f1+f2
        f1=f2
        f2=tmp
    return f2
fibonacci_iterativo_v1(6)

def fibonacci_iterativo_v2(numero):
    f1=0
    f2=1
    for i in range(1,numero-1):
        f1,f2=f2,f1+f2
    return f2
fibonacci_iterativo_v2(6)

def fibonacci_bottom_up(numero):
    f_parciales=[0,1,1]
    while len(f_parciales)<numero:
        f_parciales.append(f_parciales[-1]+f_parciales[-2])
        print(f_parciales)
    return f_parciales[numero-1]
fibonacci_bottom_up(5)

#Top down

memoria={1:0,2:1,3:1}
def fibonacci_top_down(numero):
    if numero in memoria:
        return memoria[numero]
    f=fibonacci_iterativo_v2(numero-1)+fibonacci_iterativo_v2(numero-2)
    memoria[numero]=f
    return memoria[numero]
```

```
fibonacci_top_down(12)
fibonacci_top_down(8)

import pickle
archivo=open("memoria.p","wb")
pickle.dump(memoria,archivo)
archivo.close()
archivo=open("memoria.p","rb")
memoria_de_archivo=pickle.load(archivo)
archivo.close()

#Incremental

def insertionSort(n_lista):
    for index in range(1,len(n_lista)):
        actual=n_lista[index]
        posicion=index
        print("Valor a ordenar= {} ".format(actual))
        while posicion>0 and n_lista[posicion-1]>actual:
            n_lista[posicion]=n_lista[posicion-1]
            posicion=posicion-1
        n_lista[posicion]=actual
        print(n_lista)
        print()
    return n_lista
lista=[21,10,0,11,9,24,20,14,1]
print("lista desordenada {}".format(lista))
insertionSort(lista)
print("lista ordenada {}".format(lista))

#Divide y venceras

def quicksort(lista):
    quicksort_aux(lista,0,len(lista)-1)


def quicksort_aux(lista,inicio,fin):
    if inicio<fin:

        pivote=particion(lista,inicio,fin)

        quicksort_aux(lista,inicio,pivote-1)
        quicksort_aux(lista,pivote+1,fin)
```

```
def particion(lista,inicio,fin):
    pivote=lista[inicio]
    print("Valor del pivote {}".format(pivote))
    izquierda=inicio+1
    derecha=fin
    print("Indice izquierdo {}".format(izquierda))
    print("Indice derecho {}".format(derecha))

    bandera=False
    while not bandera:
        while izquierda<=derecha and lista[izquierda]<=pivote:
            izquierda=izquierda+1
        while lista[derecha]>=pivote and derecha>=izquierda:
            derecha=derecha-1
        if derecha<izquierda:
            bandera=True
        else:
            temp=lista[izquierda]
            lista[izquierda]=lista[derecha]
            lista[derecha]=temp
    print(lista)

    temp=lista[inicio]
    lista[inicio]=lista[derecha]
    lista[derecha]=temp
    return derecha

lista=[21,10,0,11,9,24,20,14,1]
print("Lista desordenada {}".format(lista))
quicksort(lista)
print("Lista ordenada {}".format(lista))
```

```
%pylab inline
import matpotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

from insertionSort import insertionSort_time
from quicksort import quicksort_time
```

```
datos=[ii*100 for ii in range(1,21)]
tiempo_is=[]
tiempo_qs=[]

for ii in datos:
    lista_is=random.sample(range(0,10000000),ii)
    lista_qs=lista_is.copy()
    t0=time()
    insertionSort_time(lista_is)
    tiempo_is.append(round(time()-t0,6))
    t0=time()
    quicksort_time(lista_qs)
    tiempo_qs.append(round(time()-t0,6)

print("Tiempos parciales de ejecucion en INSERT SORT {} [s]
\n".format(tiempo_is))
print("Tiempos parciales de ejecucion en QUICK SORT {} [s]
\n".format(tiempo_qs))

print("Tiempos total de ejecucion en insert sort {} [s]".format(sum(tiempo_is)))
print("Tiempos parciales de ejecucion en quick sort {} [s]".format(sum(tiempo_is)))

fig,ax=subplots()
ax.plot(datos,tiempo_is,label="insert sort",marker="*",color="r")
ax.plot(datos,tiempo_qs,label="insert sort",marker="o",color="b")
ax.set_xlabel("Datos")
ax.set_ylabel("Tiempo")
ax.grid(True)
ax.legend(loc=2);

plt.tittle("Tiempo de ejecucion [s](insert vs.quick)")
plt.show()


import matpotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

times=0

def insertionSort_graph(n_lista):
    global times
    for index in range(1,len(n_lista)):
        times +=1
```

```
        actual=n_lista[index]
        posicion=index
        while posicion>0 and n_lista[posicion-1]>actual:
            times +=1
            n_lista[posicion]=n_lista[posicion-1]
            posicion=posicion-1
        n_lista[posicion]=actual
    return n_lista

TAM=101
eje_x=list(range(1,TAM,1))
eje_y=[]
lista_variable=[]

for num in eje_x:
    lista_variable=random.sample(range(0,1000),num)
    times=0
    lista_variable=insertionSort_graph(lista_variable)
    eje_y.append(times)

fig,ax=plt.subplots(facecolor="w",edgecolor="k")
ax.plot(eje_x,eje_y,marker="o",color="b",linestyle="None")
ax.set_xlabel("x")
ax.set_ylabel("y")
ax.grid(True)
ax.legend(["Insertion sort"])
plt.tittle("Insertion sort")
plt.show
```

Python 3.8.2 Shell

File Edit Shell Debug Options Window Help

```
Tu contraseña es Hol4
Tiempos de ejecucion 17.84136
[[500, 2]]
[[500, 1]]
[[50, 6]]
[[5, 40]]
[[50, 1], [20, 2], [5, 1], [1, 3]]
[0, 1, 1, 2]
[0, 1, 1, 2, 3]
lista desordenada [21, 10, 0, 11, 9, 24, 20, 14, 1]
Valor a ordenar= 10
[10, 21, 0, 11, 9, 24, 20, 14, 1]

Valor a ordenar= 0
[0, 10, 21, 11, 9, 24, 20, 14, 1]

Valor a ordenar= 11
[0, 10, 11, 21, 9, 24, 20, 14, 1]

Valor a ordenar= 9
[0, 9, 10, 11, 21, 24, 20, 14, 1]

Valor a ordenar= 24
[0, 9, 10, 11, 21, 24, 20, 14, 1]

Valor a ordenar= 20
[0, 9, 10, 11, 20, 21, 24, 14, 1]

Valor a ordenar= 14
[0, 9, 10, 11, 14, 20, 21, 24, 1]

Valor a ordenar= 1
[0, 1, 9, 10, 11, 14, 20, 21, 24]

lista ordenada [0, 1, 9, 10, 11, 14, 20, 21, 24]
Lista desordenada [21, 10, 0, 11, 9, 24, 20, 14, 1]
Valor del pivote 21
Indice izquierdo 1
Indice derecho 8
[21, 10, 0, 11, 9, 1, 20, 14, 24]
Valor del pivote 14
```

```
Valor a ordenar= 9
[0, 9, 10, 11, 21, 24, 20, 14, 1]

Valor a ordenar= 24
[0, 9, 10, 11, 21, 24, 20, 14, 1]

Valor a ordenar= 20
[0, 9, 10, 11, 20, 21, 24, 14, 1]

Valor a ordenar= 14
[0, 9, 10, 11, 14, 20, 21, 24, 1]

Valor a ordenar= 1
[0, 1, 9, 10, 11, 14, 20, 21, 24]

lista ordenada [0, 1, 9, 10, 11, 14, 20, 21, 24]
Lista desordenada [21, 10, 0, 11, 9, 24, 20, 14, 1]
Valor del pivote 21
Indice izquierdo 1
Indice derecho 8
[21, 10, 0, 11, 9, 1, 20, 14, 24]
Valor del pivote 14
Indice izquierdo 1
Indice derecho 6
[14, 10, 0, 11, 9, 1, 20, 21, 24]
Valor del pivote 1
Indice izquierdo 1
Indice derecho 4
[1, 0, 10, 11, 9, 14, 20, 21, 24]
Valor del pivote 10
Indice izquierdo 3
Indice derecho 4
[0, 1, 10, 9, 11, 14, 20, 21, 24]
Lista ordenada [0, 1, 9, 10, 11, 14, 20, 21, 24]
>>>
```

Conclusiones

Python es un lenguaje de programación en el cual podemos aplicar un gran numero de estrategias para resolver problemas, creando algoritmos dinámicos y utilizando las funciones que tiene este lenguaje. Es por ello que conocer para que sirve cada función es esencial para poder realizar un código adecuado y correcto

Bibliografía

Laboratorios A y B, Practica 11 Estrategias para la construcción de algoritmos, consultado el 23 de abril
2020, de file: http://lcp02.fi-b.unam.mx/