12

# LaTeX, git, python, Numpy, Matplotlib, gdb y profiling

Luis Sánchez Casanueva

**Resumen**—En esta tarea, mediante VMbox y una MV de linux, usaremos spyder como ide para el desarrollo del ejercicio de la práctica, a la par que mantenemos un control de las versiones durante todo el desarrollo mediante un repositorio en gitHub. Todo ello para cumplimentar y realizar el analisis y las restricciones que el enunciado[1] nos plantea, y las indicaciones de la rúbrica [2]

**Index Terms**—UMU, FIUM, Ejercicio, Python, C, Numpy, matplotlib, Latex LATEX.

## 1. INTRODUCTION

EN PRIMER LUGAR explicaremos brevemente los conceptos básicos que rodean a las diferentes tareas que realizaremos durante el desarrollo de esta practica.

### 1.1. Conceptos clave

#### *1.1.1. Python*

Es un lenguaje de programacion de código abierto interpretádo, dinámico y multiplataforma cuya filosofía es la legibilidad de su código, mediante la resolucion dinámica de nombres.

#### *1.1.2. C*

Lenguaje de programación orientado a los sistemas operativos pero actualmente de propósito general. De nivel medio pero apreciado por la eficiencia del código que produce. En nuestro caso lo usaremos para suplir aspectos que python no nos deja configurar.

#### *1.1.3. Numpy*

Es una conocida extensión 'open Source' de python que le agrega soporte para vectores y matrices con funciones matemáticas de alto nivel.

#### *1.1.4. matlibplot*

Es una biblioteca para la generación de graficos a partir de listas o arrays en Python. Proporciona una API 'pylab' que se asemeja a la de 'MATLAB'

#### *1.1.5. git*

Software para el control de versiones, de código libre, y ampliamente usado en la actualidad.

#### *1.1.6. gitHub*

Es una plataforma de desarrollo colaborativo que aloja proyectos utilizando el control de versiones de Git.

- *Muchas gracias al lector y a los profesores de la UM, y mas especificamente a los de la FIUM por su labor en estos tiempos de excepcionalidad*
*E-mail: luis.sanchezc@um.es*
- *Uso solo para evaluación docente, S. Luis adscrito a la UMU*

*Manuscrito realizado el 19/05/2020*

## 2. RESOLUCIÓN DEL PROBLEMA

En primer lugar plantearemos el ejercicio y a continuación iremos detallando como fue el desarrollo del ejercicio.

### 2.1. Enunciado del ejercicio

Imaginemos que, dada una lista de elementos, queremos obtener una nueva versión de la misma sin repeticiones. Python nos ofrece los conjuntos para ello, pero también podríamos conseguir lo mismo usando listas, diccionarios, etc. Queremos evaluar las distintas posibilidades que tenemos en Python para saber cuál es la más eficiente y compararlas con una implementación propia en C de una función de eliminación de duplicados.

Debemos implementar un programa en Python que reciba *tres parámetros*: *un fichero de entrada* con una lista de elementos, uno por linea, *un fichero de salida* donde se guardará esa misma lista de elementos pero sin repeticiones, *y otro fichero de salida* donde se guardará en PDF la gráfica que mostrará el tiempo usado por cada técnica para los distintos tamaños de datos. Esta gráfica se generará con Matplotlib. El orden de salida puede ser distinto al de entrada.

El fichero de entrada debe contener 2000000 numeros y deben ser numeros naturales entre 0 y 99999 (ambos inclusive), posteriormente, hacer un estudio de los tiempos de las técnicas incrementando de 2000 en 2000 el numero de casos analizados hasta llegar a los 200000.

Dos de los metodos deben ser mediante estructuras diferentes en Python y una tercera forma que debemos implementar en C y que usaremos desde Python a través de ctypes

Este último método implementado en C, debe recibir la lista de elementos original y devolver la lista sin duplicados. La memoria dinámica que se necesite reservar se reservará en Python. Los alumnos eligen que código implementar para que sea lo más eficiente posible.

Aparte de la gráfica generada en PDF se deberáincluir un perfilado (profiling) del tiempo de ejecución y el consumo de memoria durante la ejecución.

Se deben controlar posibles errores, mediante la captura de excepciones, mostrando en su lugar un mensaje de error informativo.

## 2.2. Desarrollo del ejercicio

Para la resolución del ejercicio partimos del código de partida proporcionado para el boletín 3 del bloque 4 de prácticas[3].

### 2.2.1. Control de versiones

Lo primero que hice fue inicializar nuestro repositorio git local con el codigo de partida, posteriormente cada subseccion compondrá un hito que considere suficiente para merecer un commit. No subí a la plataforma de github mi repositorio hasta casi el final[3], se puede ver un 'merge branch' en los commits cuando tubo lugar.

### 2.2.2. Desarrollo del generador en Python

En primer lugar, dado que no tenia nociones de Python, empece con la tarea de generar el fichero de entrada con el formato indicado para ir cogiendo habilidades y nociones con un problema sencillo. En primer lugar comprueba que el número de parámetros sea el correcto, es decir: uno, y que puede crear un fichero con el texto que le hemos pasado por parámetro.

A continuación y mediante las extensiones que nos ofrece Numpy, creamos un array aleatorio de numeros reales entre '0' y '1', y lo multiplicamos por el máximo número con el que queremos poblar nuestro array generado, '99999'. Iteramos el array y los escribimos en el fichero, y por ultimo lo cerramos.

### 2.2.3. Esqueleto y primer método

A continuación y usando como guia el código proporcionado en 'bloque4.py' modifique la función 'main()' para que aceptara el numero de parametros especificados en la práctica e hiciera las correspondientes comprobaciones, es decir: Comprobar que el numero de parametros es '3', comprobar que se puede abrir el archivo pasado como primer parámetro y comprobar que ningun número del archivo supere el 99999.

Una vez los parametros eran almacenados correctamente y era capaz de operar con ellos implementé mi primer método nativo de Python para eliminar duplicados, utilicé la funcionalidad de 'set()' para eliminar duplicados de una lista, pero me devolvia un tipo de datos que no me aceptaba 'matplotlib' para representar, fue necesario procesarlos con 'sorted()' para que fuera hasheables (les añade un orden).

Dado que aun no estaba seguro del funcionamiento, aproveche estas entradas para debugger y comprender el funcionamiento del código en Python, realmente aun no entendia que se me pedia dibujar como comparativa y a esta altura aun no se dibujaba nada. El commit se realiza cuando se genera el fichero de salida, con el formato correcto y la solución correcta (duplicados eliminados) con este primer método (no conserva el orden).

### 2.2.4. Segundo método y comprension de la grafica objetivo

Cuando iva a implementar el segundo metodo de eliminación de dúplicados en Python, releyendo el enunciado caí en que me pediais el resultado de comparar los rendimientos de los diferentes métodos para diferentes tamaños de entrada. Modifique ligeramente el main() para guardar las listas de los tiempos de cada metodo, y modifique el flujo para meter el bucle que me permitiría comprar los resultados. También añadi un segundo método mediante 'set()' y 'list()' que es el mas habitual (conserva el orden)

### 2.2.5. Dibujo mediante matplotlib de la comparativa

En esta fase, modifique el funcionamiento de la funcion proporcionada para dibujar la gráfica, para que mostrara los datos de los tiempos de ambos métodos (1 y 2) y su correcta nomenclatura en el gráfico.

### 2.2.6. Funcion en C

Esta fase era en la que más cómodo me encontraba, escribi tanto wrapper.c como wrapper.h, se que no es correcto ni necesaria esa nomenclatura dado que el wrapper (envoltorio) es la clase en Python que ejecuta la librería dinámica, pero en el momento no me di cuenta. El algoritmo usado para eliminar los duplicados es una especie de 'algoritmo de la burbuja' pero solo moviendo al elemento repetido al ultimo lugar y decrementando el size del array.

Posteriormente me daria cuenta que su eficiencia era nula y lo cambiaría por uno un poco mas eficiente ( me deje el grueso del algoritmo comentado y no vi su ineficiencia ).

### 2.2.7. Implementación del wrapper

Esta parte de la tarea era la que más me imponía, conceptualmente no sabía como llevar a cabo la comunicación mediante los ctypes de python a c. El esqueleto y estructura el el mismo que el proporcionado pero adaptando el numero de parametros, el nombre y ubicación de las librerias y los parámetros suministrados.

El commit se realiza funcionando el metodo 3 en el bloque principal y pudiendo medir sus tiempos.

### 2.2.8. Inclusion del 3er método en la grafica

Este commit lleva consigo la inclusión del método 3 en los analisis realizados y cambiando la manera de dibujarlos en un primer intento por dibujar ambas modelos de gráfica originales en el pdf.

### 2.2.9. Impresión al pdf

A continuación modifique los metodos de impresion y creacion de las figuras para poder generar y guardar el archivo, se me olvidó controlar la excepción de no poder crear el archivo, será incluida en un commit posterior.

### 2.2.10. Distribucion del proyecto

En este commit redistribuí los elementos en la estuctura solicitada por los profesores (/doc /src) y modifique las relaciones en codigo para que se encontraran las librerias en sus nuevas localizaciones. La carpeta /doc no se incluyo en el repositorio al estar vacia y no contener ningun archivo aun.

### 2.2.11. Aclaracion

Los dos siguientes commits corresponden al momento en el que caí que era necesario que el repositorio git estuviera accesible online a través de la plataforma de GitHub, elimine un README del repositorio online (primer commit), para que no archivos conflictivos y poder hacer un pull al repositorio remoto, y a continuacion realice un pull origin con el flag '–allow-unrelated-histories'.

#### 2.2.12. README y segundo intento

Volvi a intentar añadir la carpeta '/doc' vacia, e inclui un readme al proyecto.

#### 2.2.13. último commit/pull

Por último, realice este documento, y tras finalizar su edicion realice el último pull, junto con las correcciones de errores que habia ido encontrando al tener que desgranar todo el proceso realizado en los últimos días para la elaboración del mismo.

Entre otros correción de guardas al crear el pdf, descomentar el metodo en C y por tanto darme cuenta de lo terrible de su eficiencia y tratar de corregirlo con la version 2.0 y otros menores

### 2.3. Apendice

A continuación listamos el código de las clases y el diagrama resultante de ejecutar la comparativa entre los tres métodos

#### 2.3.1. base.py

```
import sys
import time
import numpy as np
import matplotlib.pyplot as plt
from wrapper import wrapper
from matplotlib.backends.backend_pdf import PdfPages


# Funcion auxiliar: muestra la figura matplotlib pendiente
# y espera una pulsacion de tecla:
#@profile
def show_plot_and_wait_for_key():

    plt.show()
    input("<Hit Enter To Close>")
    plt.close()


# Funci n auxiliar que muestra en matplotlib los arrays de entrada y de salida:
def plot_values(values_in1, values_in2, values_in3, name,
                        line_else_bars=True, width=0.5):
    f = plt.figure()
    if line_else_bars == True:
        plt.plot(values_in1, color = 'r', label="Method 1")
        plt.plot(values_in2, color = 'g', label="Method 2")
        plt.plot(values_in3, color = 'b', label="Method 3")
    else:
        plt.bar(np.arange(len(values_in1)) - width, values_in1, width=width, color='r',
                label="Method 1")
        plt.bar(np.arange(len(values_in2)), values_in2, width=width, color='g',
                label="Method 2")
        plt.bar(np.arange(len(values_in3)), values_in3, width=width, color='b',
                label="Method 3")

    plt.title('Comparative of time spend between the methods'.format(["bars", "lines"]
                                                                     [line_else_bars]))
    plt.legend()
    plt.xlabel('CasosConsiderados * 2000')
    plt.ylabel('Segundos')
    try:
        f.savefig(str(name + ".pdf"),bbox_inches='tight')
    except:
        print("Error al crear el archivo pdf con los gr ficos")

# C digo main:
def main():
    # Control de argumentos de l nea de comandos:
    if len(sys.argv) != 4:
        print("Uso: {} ficheroEntrada , ficheroSalida , pdfLogFile".format(sys.argv[0]))
        sys.exit(0)

    #reading fEntrada
    try:
        f = open(sys.argv[1])
        lista = f.readlines()
    except:
        print("No se encuentra el archivo de Entrada")
        sys.exit(-1)

    slist = sorted(lista)

    #parsing fEntrada
    try:
        for number in slist:
            N = int(number)
            if not (0 <= N <= 99999):
                raise ValueError()
    except:
        print("All values must be a int value between 0 and 99999")
        sys.exit(-1)

    timeOption1 = []
    timeOption2 = []
    timeOption3 = []

    #Creamos las soluciones
    for i in range(2000,200000,2000):
        #M todo 1
        t0_sol1 = time.time_ns()
        sol1 = sorted(set(slist[0:i]))
        texec_sol1 = (time.time_ns()-t0_sol1)/1.0e9

        #M todo 2
        t0_sol2 = time.time_ns()
        sol2 = list(set(slist[0:i]))
        texec_sol2 = (time.time_ns()-t0_sol2)/1.0e9



        #M todo 3
        listTarget = slist[0:i]
        results = [int(i) for i in listTarget]
        sol3 = np.zeros_like(listTarget)
        t0_sol3 = time.time_ns()
        sol3 = wrapper(results,i)
        texec_sol3 = (time.time_ns()-t0_sol3)/1.0e9



        #Guardamos los datos para esta iteracion
        timeOption1.append(texec_sol1)
        timeOption2.append(texec_sol2)
        timeOption3.append(texec_sol3)

    #Creamos el archivo solucion
    try:
        name = sys.argv[2]
        file = open(name, "w")

    except:
        print("Can't create outputList file")
        sys.exit(-1)

    for num in sol1:

        file.write(str(num))


    # Guardamos los graficos en el pdf de salida marcado

    print("Comenzamos el plot de las comparativas")

    plot_values(timeOption1, timeOption2, timeOption3, sys.argv[3])

    show_plot_and_wait_for_key()



if __name__ == '__main__':
    main()
```

#### 2.3.2. generador.py

```
import numpy as np
import sys

def main():
    # Control de argumentos de l nea de comandos:
    if len(sys.argv) != 2:
        print("Uso: {} nombreFichero".format(sys.argv[0]))
        sys.exit(0)

    try:
        name = sys.argv[1]
        file = open(name, "w")

    except:
        print("Can't create file")
        sys.exit(-1)


    #Generamos el array de valores
    SIZE = 200000  # Tama o del array.
    arr = np.random.rand(SIZE) * 99999

    #Lo guardamos en file
    for x in arr:
        numero = int(x)
        file.write(str(numero))
        file.write("\n")

    file.close


if __name__ == '__main__':
    main()
```

#### 2.3.3. wrapper.py

```
import ctypes, os
import numpy as np

# Wrapper python para llamar a la funci n implementada en C.
def wrapper(vin, size):
    # Objeto correspondiente a la funci n dentro de la biblioteca.
    funcwrapper = LIBWRAPPER.wrapper

    # Prototipo de la funci n: dos arrays a floats, la longitud de
    # los arrays y el escalar de multiplicaci n. Observa que ctypes no
    # define punteros a datos que no sean c_char, c_wchar y c_void, por
    # lo que hay que crearlos con POINTER.
    funcwrapper.argtypes = [ctypes.POINTER(ctypes.c_float),
                            ctypes.POINTER(ctypes.c_float),
                            ctypes.c_int]

    # Valor devuelto por la funci n (se puede eliminar, pues es el
    # comportamiento por defecto).
```

```
    funcwrapper.restype = ctypes.c_int

    # Puesto que ctypes espera que los dos primeros par metros sean
    # instancias de punteros a c_float (es decir, instancias LP_c_float),
    # seg n el prototipo de la funci n, no podemos usar directamente ni
    # vin ni vout (como copia de vin), ya que esos elementos son de tipo
    # List y no instancias de LP_c_float. Lo que hacemos es definir salida
    # como un array de valores c_float, con tantos elementos como tiene vin.
    # Observa que hacemos lo mismo con el primer par metro que se le pasa
    # a la funci n, si bien, en este caso, los valores deben ser los del
    # vector de entrada.

    salida=(ctypes.c_float * size)()
    entrada =(ctypes.c_float * size)(*vin)
    # Llamada a la funci n de la biblioteca compartida.
    funcwrapper(entrada, salida, size)

    # Vamos a devolver un vector. Para eso, hacemos una copia del vector
    # de entrada. Podr amos devolver directamente    salida    , pero ser a
    # un objeto de ctypes tal y como lo hemos definido, con lo que una
    # simple orden   print (salida)   no nos mostrar a su contenido sino
    # su tipo. Queda m s "elegante" devolver algo como la entrada.
    vout=np.resize(vin.copy(), len(salida))

    # Copiamos a dicho vector de salida el resultado de la funci n.
    for i in range(len(vout)):
      vout[i]=salida[i]

    # Devolvemos el vector de salida.
    return vout

if __name__ == "wrapper":
    # Cargamos la biblioteca compartida en ctypes.
    LIBWRAPPER = ctypes.CDLL (os.path.abspath(os.path.join(
                                                os.path.dirname(__file__),
                                                "../C/libwrapper.so.1"
                                                )))
```

### 2.3.4. wrapper.c v1.0 (DESCARTADA)

```
int wrapper( float *vin, float *vout,  int size)
{
   for(int i = 0; i<size-1; i++)
    {
        for(int j = i+1; j< size; j++)
        {
            if(vin[i] == vin[j])
            {
                for(int k = j; k<size-1; k++)
                {
                    float aux;
                    aux = vin[k];
                    vin[k] = vin[k+1];
                    vin[k+1] = aux;
                }
                size--;
                j--;
            }
        }
    }

    for(int i=0;i<size;i++)
    {
        vout[i] = vin[i];
    }
    realloc(vout, size * sizeof(*vout));
    return 0;
}
```

### 2.3.5. wrapper.c v2.0

```
int wrapper( float *vin, float *vout,  int size)
{

    int array[size];


    for(int i = 0; i<size; i++)
    {
        array[(int)vin[i]] = 1;
    }

    int newsize=0;
    for(int i=0;i<size; i++)
    {
       if(array[i]==1){
           vout[newsize] = i;
           newsize++;
        }
    }
    realloc(vout, size * sizeof(*vout));
    return 0;
}
```

### 2.3.6. wrapper.h

```
int wrapper(float *vin,    // Vector de entrada
            float *vout,   // Vector de salida
            int size);     // Longitud de los vectores
```

### 2.3.7. Makefile

```
all: libwrapper.so.1.0.1 libwrapper.so.1

libwrapper.so.1.0.1: wrapper.c wrapper.h
        gcc -Wall -c -fPIC -g wrapper.c
        gcc -shared -Wl,-soname,libwrapper.so.1 -o libwrapper.so.1.0.1
                                                        wrapper.o

libwrapper.so.1: libwrapper.so.1.0.1
        ln -s libwrapper.so.1.0.1 libwrapper.so.1

clean:
        rm -fr libwrapper.so.1.0.1 libwrapper.so.1 wrapper.o *~
```

## 3. CONCLUSIÓN

Como siempre el camino para la realización de esta práctica ha estado plagado de piedras, probé primero a usar EVA para su realización, pero el sistema debe ester saturado, era impensable realizar la practica con unos tiempos de respuesta tan lentos. A continuación probe con el subsitema de linux integrado en w10 en mi portatil, pero fui incapaz de usar la interfaz gráfica de SPYDER puesto que el subsistema no conoce a los displays. Mi instalación de ubuntu dual con windows mediante GRUb no funciona, se me apaga cada vez que intento iniciarlo.

Por suerte, fui capaz de realizar la práctica en su totalidad creando una VM para ello.

Respecto al contenido de la práctica en si mismo me a parecido exquisito, nunca habia trabajado con Python ni creado librerias dinamicas en un lenguaje diferente al del programa que estaba creando, me ha aportado un primer contacto con este lenguaje y las maravillosas posibilidades que ofrece, es un lenguaje que tengo pendiente dada mi afinidad por las IAs y el machine learning. LaTeX por su parte ya ha demostrado a lo largo del curso su utilidad y la facilidad que nos ofrece para generar documentos de calidad facilmente. No creo que mi función implementada en C haya cumplido en terminos de rendimiento, me habría gustado disponer de más tiempo para poder optimizarla más dado que ambas soluciones que planteé eran pésimas. Las funcionalidades de Git/GitHub en cambio han quedado en un segundo plano en mi caso, realizandose de manera metódica para cumplir un requisito pero dado que el proyecto lo he realizado de manera individual no hemos podido disfrutar de las ventajas que ofrece. En mi dia a dia yo suelo usar los servicios de google como repositorio, guarda historial de versiones para cada archivo con comentarios autor y fechas, la unica funcionalidad que no incorpora es la de generar ramas diferentes para el desarrollo del proyecto.

## REFERENCIAS


[1] https://aulavirtual.um.es/access/content/attachment/1918_G_2019_N_N/Tareas/9b880415-d8d5-4fd5-b503-a8359726218c/PracticaEntregable-Bloques3y4.pdf *UMU TEII*. Entregable bloques 3-4 Curso 19/20.

[2] https://aulavirtual.um.es/access/content/attachment/1918_G_2019_N_N/Anuncios/3dbcbb7a-fbc5-4cbc-ba5a-399d11da60ab/RubricaTEII-B3-B4.pdf *Docker Docs*. Rubrica entregables 3-4 Curso 19/20

[3] https://github.com/Rakjas/b3-4TEII *Docker Docs*. URL repositorio GitHub del proyecto



**Luis Sánchez Casanueva** Luis Sánchez Casanueva, nacido el 3 de Julio de 1992 en Madrid, España, termino bachiller bilingüe en el 2010. Curso un grado en arquitectura entre 2010-2015 en Cartagena, abandonando en sus ultimos años de carrera por encotrarse incomodo con los contenidos docentes/profesionales que estaba adquiriendo y en busca de una carrera mas practica y logica que le realizara intelectualmente. Actualmente cursa 3º/4º Ingenieria informatica en la UMU, y dedica su

tiempo al estudio de las Redes Neuronales y Sistemas Inteligentes combinadas con algunas de las ramas de investigación mas prometedoras como la Edición genética mediante proteinas (CRISPER, viricos..) y la Cuántica.