Laboratorio de Informática I – 2547100
Departamento de Ingeniería Electrónica y de Telecomunicaciones
Facultad de Ingeniería
Universidad de Antioquia

Práctica 5: Eficiencia de programas numéricos iterativos

1. Objetivos

- Poner en práctica los conceptos básicos de programación en Python
- Resolver problemas numéricos con algoritmos simples basados en iteración
- Comparar la eficiencia de diferentes algoritmos que resuelven un mismo problema

2. Marco teórico

La técnica algorítmica llamada **búsqueda lineal** utiliza la *enumeración exhaustiva* para encontrar un dato que se busca en una lista. Cada elemento de la lista es evaluado hasta encontrar el dato buscado. Esta técnica se considera de fuerza bruta pues no aprovecha ninguna característica de la posible organización de los datos de la lista para hacer una búsqueda más rápida. La **búsqueda binaria**, por el contrario, nos permite acortar enormemente los pasos necesarios para encontrar el dato buscado, en el caso en que los datos de la lista estén ordenados. En cada paso, la búsqueda binaria nos permite reducir a la mitad el espacio de búsqueda.

Estas dos técnicas también pueden ser aplicadas al problema de encontrar las raíces de un polinomio. Cuando tenemos un intervalo definido en un polinomio, podemos utilizar la enumeración **exhaustiva** para encontrar sus raíces, evaluando la función para cada valor candidato de la raíz. Además podemos aplicar la técnica de búsqueda binaria para encontrar más rápido las raíces de un polinomio. A esta técnica se le conoce como el **método de bisección**.

3. Tareas a realizar

Esta práctica consiste en la implementación de dos programas para buscar datos en una lista y dos programas más para encontrar las raíces de un polinomio. Cada pareja de programas cumple la misma función pero utiliza algoritmos con diferentes niveles de eficiencia.

Búsqueda en una lista

Una de las acciones algorítmicas más comunes es la de buscar una secuencia de elementos para determinar si un dato dado se encuentra o no. En los ejercicios 3.1 y 3.2, la secuencia de datos se almacena en una variable tipo lista del lenguaje Python. El tipo de dato lista puede albergar combinaciones de todos los tipos de datos, es no-escalar y además es mutable, lo que lo hace muy flexible.

3.1. Implemente el algoritmo de búsqueda lineal, completando el archivo busqueda_lineal_template.py y guiándose por las indicaciones que aparecen en los comentarios. Ayuda: recuerde el uso de la instrucción break.

3.2. Implemente el algoritmo de búsqueda **binaria**, completando el archivo `busqueda_binaria_template.py` y guiándose por las indicaciones que aparecen en los comentarios. Asuma que la lista de datos está ordenada.

Modifique en ambas implementaciones la lista de búsqueda tal que esta sea:

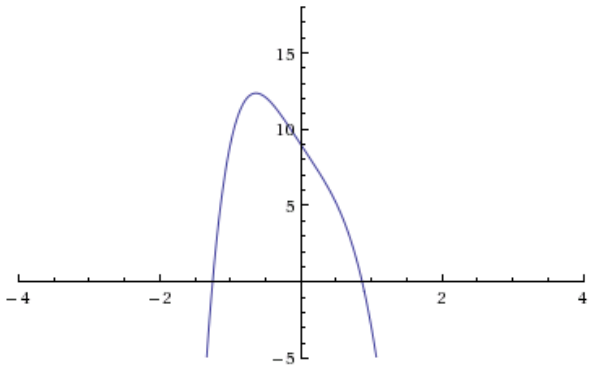
```
L = [-50, -45, -23, -21, -14, -9, -2, 0, 1, 3, 5, 16, 17, 24, 29, 30, 40, 52, 53, 92]
```

Así mismo modifique las claves (valores a buscar) para cada algoritmo y llene la siguiente tabla para **observar la eficiencia de cada alternativa**. Obtenga el número de iteraciones del algoritmo a partir de ejecutar los programas para cada valor que se indica.

Clave	Número de iteraciones	
	Lineal	Binaria
-45		
-21		
0		
92		
100		

Raíces de un polinomio

Considere la función $f(x) = x^5 - 6x^4 - 7x + 9$ cuya gráfica se muestra a continuación:



Se requiere hallar el valor aproximado de la raíz de esta función en el intervalo $[-2, -1]$ empleando una tolerancia $\epsilon = 0.1\%$.

3.3. Implemente el algoritmo de búsqueda de raíces para la función anterior usando el método **exhaustivo**. Para ello complete el archivo `raices_exhaustivo_template.py` guiándose por los comentarios en el código fuente y usando $\Delta x = 0.00001$.

3.4. Implemente el algoritmo de búsqueda de raíces para la función anterior usando el método de **bisección**. Para ello complete el archivo `raices_biseccion_template.py` guiándose por los comentarios en el código fuente.

3.5. ***(Opcional)*** Implemente el algoritmo de búsqueda de raíces para la función anterior usando el método de **Newthon-Raphson**.

Analice la eficiencia de los algoritmos observando la cantidad de iteraciones que cada uno de los dos programas necesita para encontrar la solución.

Reporte de resultados

3.6. Integre los cuatro algoritmos en un solo programa que al final imprima en pantalla lo siguiente:

```
-------------------------------------
| Algoritmo | Iteraciones promedio |
-------------------------------------
| Lineal    |          PL          |
| Binaria   |          PB          |
-------------------------------------
```

```
 -------------------------------
| Algoritmo       | Iteraciones |
 -------------------------------
| Exhaustivo      |     IE      |
| Bisección       |     IB      |
| Newthon-Raphson |     INR     | (Opcional)
 -------------------------------
```

Donde PL y PB deben ser los promedios de la cantidad de iteraciones que cada algoritmo realiza para los valores de prueba mostrados en la tabla anterior. IE, IB, INR son el número total de iteraciones para encontrar la raíz de la función en el intervalo.

4. Evaluación

La evaluación se basará en los códigos enviados y una sustentación oral sobre los temas de la práctica. Además, se tendrá en cuenta una bonificación para quien haga la parte opcional.