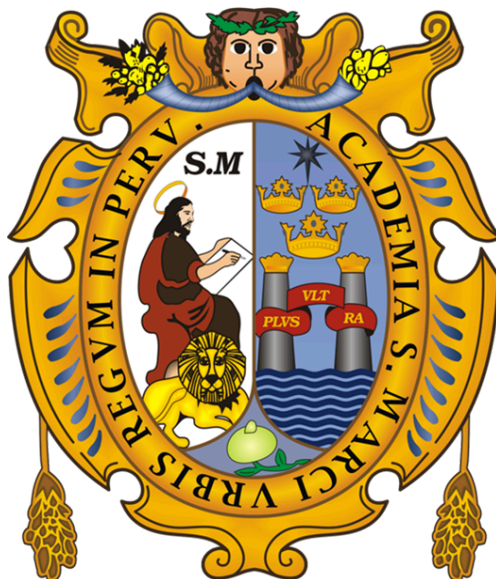


“Año del Bicentenario, de la consolidación de nuestra Independencia, y de la conmemoración de las heroicas batallas de Junín y Ayacucho”

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

(Universidad del Perú, DECANA DE AMÉRICA)

FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
Escuela Profesional de Ingeniería de Software



“Grupo #7”

CURSO: Análisis y diseño de algoritmos

DOCENTE: Chavez Soto, Jorge Luis

INTEGRANTES:

- Cano Vasquez, Juan José
- Estrada Briceño, Erick Martin
- Vincula Nuñovero, Matias Sebastian

Lima, 2024

OBJETIVOS DEL PROYECTO:

- Evaluación y Análisis de Algoritmos

Diseñar y desarrollar una herramienta que permita a los usuarios ingresar el código fuente de un algoritmo y recibir un análisis sobre su tiempo de ejecución y complejidad temporal. La herramienta implementará un módulo que:

- Analiza el Código: Utilizará técnicas de análisis estático para identificar la estructura del código, incluyendo bucles, condiciones y recursiones.
- Calcula la Complejidad Temporal: Determinará la complejidad asintótica del algoritmo utilizando notaciones como $O(1)$, $O(n)$, $O(n^2)$ entre otras, basándose en el análisis realizado.

- Representación gráfica de complejidad

Creación de visualizaciones gráficas que representen la complejidad temporal de los algoritmos analizados. Permitiendo comparar algoritmos en un solo gráfico, facilitando la identificación de cuál es más eficiente bajo diferentes condiciones así como que muestran cómo varía el tiempo de ejecución en función del tamaño de entrada, permitiendo a los usuarios observar el comportamiento del algoritmo a medida que cambian los parámetros.

- Comparación de Algoritmos

Permite a los usuarios comparar diferentes algoritmos en términos de eficiencia y rendimiento mediante la implementación de un sistema que evalúe y contraste las funciones de tiempo calculadas para cada algoritmo ingresado, permitiendo determinar cuál es más eficiente en función de su complejidad temporal.

DESCRIPCIÓN DEL PROYECTO:

El proyecto consiste en el desarrollo de una aplicación diseñada para la evaluación y análisis de algoritmos, con el propósito de proporcionar a los usuarios una herramienta que les permita ingresar código fuente, analizar su complejidad temporal.

- Funcionalidades Principales
 - Recepción y Clasificación de Algoritmos: La aplicación permitirá a los usuarios ingresar fragmentos de código y clasificarlos automáticamente según su tipo (por ejemplo, búsqueda, ordenamiento, recursivo). Facilitando una organización efectiva del código ingresado.
 - Análisis de Complejidad Temporal: Se implementará un módulo que calculará la función tiempo y la notación asintótica del algoritmo ingresado. Esto proporcionará a los usuarios una comprensión clara de cómo se comporta su algoritmo en función del tamaño de la entrada.
 - Representación Gráfica: La herramienta incluirá capacidades para graficar las funciones de tiempo calculadas, permitiendo a los usuarios visualizar el rendimiento de sus algoritmos en comparación con otros. Esta visualización es esencial para identificar rápidamente cuál algoritmo es más eficiente.
 - Comparación de Algoritmos: Los usuarios podrán seleccionar múltiples algoritmos para compararlos directamente en términos de eficiencia. La aplicación mostrará claramente las diferencias en sus complejidades temporales.


CONJUNTO DE DATOS DE PRUEBAS PARA ENTRADA Y SALIDA DEL SISTEMA

Búsqueda Binaria

```
public int binarySearch(int[] arr, int target) {
    int left = 0;
    int right = arr.length - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == target) {
            return mid;
        }
        if (arr[mid] < target) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return -1;
}
```

Detalles del Algoritmo

 **Nombre:** Binario
ID: 1
Código:
public int binarySearch(int[] arr, int target) {
 int left = 0;
 int right = arr.length - 1;

 while (left <= right) {
 int mid = left + (right - left) / 2;
 if (arr[mid] == target) {
 return mid;
 }
 if (arr[mid] < target) {
 left = mid + 1;
 } else {
 right = mid - 1;
 }
 }
 return -1;
}


Complejidad: O(log n)
Tipo: Búsqueda

OK

Búsqueda Lineal

```
public int linearSearch(int[] arr, int target) {
    for (int i = 0; i < arr.length; i++) {
        if (arr[i] == target) {
            return i;
        }
    }
    return -1;
}
```

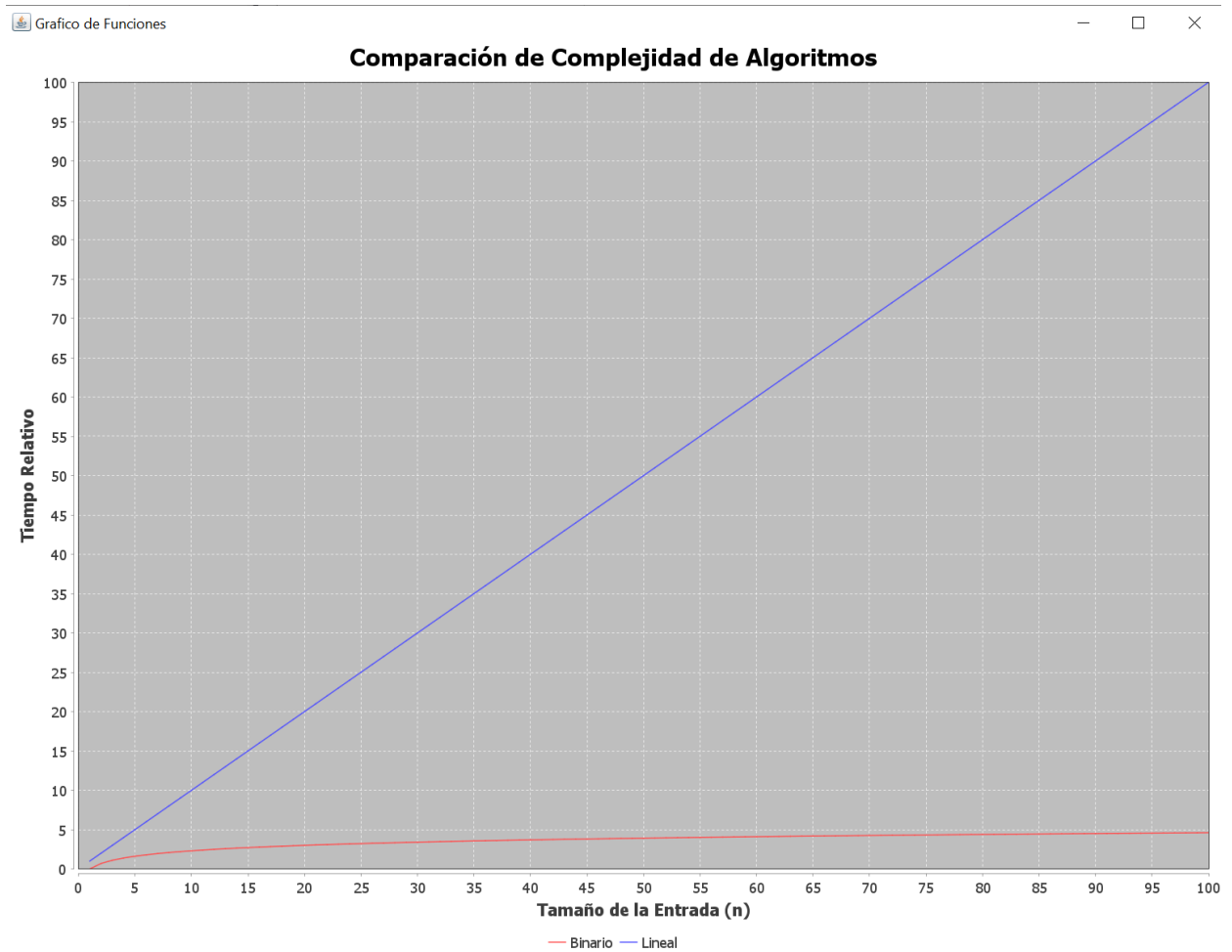
Detalles del Algoritmo

 **Nombre:** Lineal
ID: 2
Código:
public int linearSearch(int[] arr, int target) {
 for (int i = 0; i < arr.length; i++) {
 if (arr[i] == target) {
 return i;
 }
 }
 return -1;
}

Complejidad: O(n)
Tipo: Búsqueda

OK

Tabla de comparación de algoritmos para Búsqueda Lineal y Burbuja



GUIA DE USUARIO

La aplicación permite el ingreso de fragmentos de código de algoritmos para evaluar su complejidad temporal, clasificar el tipo de algoritmo y visualizar gráficamente los resultados. La interfaz gráfica posee detalles como el Ingreso, Listado, Detalles, Eliminación y Comparación de Algoritmos estos poseen funciones para el uso del programa:

- Ingreso Algoritmos: Opción para ingresar un nuevo algoritmo en base a un formulario de ingreso el cual posee campos para el Ingreso del nombre del algoritmo así como su código fuente, una vez completo estas partes se presiona el botón "Agregar" para guardar el algoritmo en el repositorio.
- Mostrar Detalles: Una vez seleccionado un algoritmo al presionar esta opción se mostrará los detalles del algoritmo siendo estos el Nombre, ID del

algoritmo agregado, código del algoritmo así como su Complejidad y Tipo de algoritmo.

- Comparar Algoritmos: Para la comparación se seleccionará dos algoritmos y se hará click en “Comparar Algoritmos” para ver un gráfico comparativo entre sus complejidades temporales.
- Eliminar Algoritmos: Esta opción se basa en seleccionar un algoritmo y hacer click en el botón de “Eliminar Algoritmo” para eliminar aquel algoritmo agregado del listado y repositorio.

GUIA DE REFERENCIA DE BIBLIOTECAS DE LAS CLASES

Clase Algoritmo:

- `java.util.ArrayList`: Manejo de listas dinámicas de algoritmos, permitiendo agregar y gestionar múltiples instancias de la clase Algoritmo.
- `java.io`: Manejo de operaciones de entrada/salida, como la lectura y escritura de archivos que almacenan los algoritmos.

Clase Complejidad:

- `java.util.regex.Pattern`: Realizar análisis del código utilizando expresiones regulares. Para Identificar patrones específicos en el código que ayudan a determinar la complejidad temporal.
- `java.util.regex.Matcher`: Complementa a Pattern para buscar coincidencias en el texto del código fuente.

Clase Repositorio

- `java.util.List`: Definir el listado de algoritmos, facilitando operaciones como búsqueda y listado.
- `java.util.ArrayList`: Utilizada para almacenar los objetos Algoritmo.
- `java.io.*`: Manejo de la persistencia de datos, incluyendo la lectura y escritura de archivos que almacenan información sobre los algoritmos ingresados.

Clase InterfazGrafica

- `javax.swing.*`: Creación de la interfaz gráfica del usuario (GUI), incluyendo componentes como JFrame, JPanel, JButton, JList, etc.
- `org.jfree.chart.*`: Generación de gráficos que representan visualmente las funciones de tiempo de los algoritmos analizados. Esencial para la visualización gráfica.
- `java.awt.*`: Gestión de aspectos gráficos adicionales, como el diseño y disposición de componentes en la interfaz.

CONCLUSIONES

El desarrollo del proyecto para la creación de una es una herramienta utilizada para la evaluación y análisis de algoritmos, cumpliendo objetivos planteados mediante la integración de funcionalidad como análisis de complejidad temporal, clasificación de algoritmos, representación gráfica y comparación de eficiencia, con la finalidad de brindar una visión práctica sobre el comportamiento de algoritmos ingresados.

Este proyecto se centra en el entendimiento de conceptos algoritmos como la notación asintótica y la eficiencia, mediante el uso de gráficos de funciones.

Además, mediante el uso de librerías relacionadas a java como seria org.jfree.chart para la generación de gráficos o regex.pattern para la identificación de patrones en el código lo cual ayuda a proporcionar experiencia en el uso de herramientas relacionadas a Java.

RECOMENDACIONES

Se debe considerar que el proyecto en estado actual está en base a lo que es un prototipo de cómo se podría ampliar el sistema en base a funcionalidades como sería el caso de análisis de espacio de memoria junto al análisis temporal para una evaluación más extensa de los algoritmos así como soporte a lenguajes diversos como Python o C++,

Además de las consideraciones para funcionalidad se debe considerar la escalabilidad del proyecto mediante integración de base de datos para el almacenamiento o migrar a una web para acceso remoto y así poder probar diversos códigos y la inclusión de un módulo para retroalimentación en base a los algoritmos ingresados al sistema.

REFERENCIAS Y BIBLIOGRAFÍA

Jose Rivera (2021). Introducción a la complejidad temporal de los algoritmos.

<https://www.freecodecamp.org/espanol/news/introduccion-a-la-complejidad-temporal-de-los-algoritmos/>

JHU cs226fa22 (2022). Data Structures Asymptotic Growth Rate

<https://cs226fa22.github.io/notes/09-growth/step05.html>

Sandeep Jain (2023). Exponential Search.

<https://www.geeksforgeeks.org/exponential-search/>

Sergio Alexander (2018). Analisis comparativo de algoritmos de ordenamiento.

<https://www.pereiratechtalks.com/analisis-de-algoritmos-de-ordenamiento/>