**UNIVERSIDAD DE LAS FUERZAS ARMADAS-ESPE**

**SEDE SANTO DOMINGO DE LOS TSÁCHILAS**

**DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN - DCCO-SS**

**CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN**



**PERIODO**         :       202351 noviembre 23 – marzo24

**ASIGNATURA**               :          Estructura de Datos

**ESTUDIANTE** :           Gualpa Mathias

Saldarriga Gissela

**NIVEL-PARALELO –NRC** :    Tercero A – Nrc 16132

**DOCENTE**       :           Ing. Javier Cevallos

**FECHA DE ENTREGA**      :       19/02/2024

**SANTO DOMINGO - ECUADOR**

**Contenido**

[**1.** **Introducción** 3](#_Toc159082241)

[**2.** **Objetivos** 3](#_Toc159082242)

[- **Objetivo General:** 3](#_Toc159082243)

[- **Objetivos Específicos:** 3](#_Toc159082244)

[**3.** **Desarrollo:** 4](#_Toc159082245)

[**3.1.** **Técnicas de Análisis de Algoritmos.** 4](#_Toc159082246)

[**3.2.** **Eficiencia de los algoritmos** 4](#_Toc159082247)

[- **¿Cómo se puede analizar la eficiencia de los algoritmos?** 4](#_Toc159082248)

[- **Pasos para analizar la eficiencia de los algoritmos** 5](#_Toc159082249)

[- **Ejemplo de código (Copiado)** 6](#_Toc159082250)

[- **Ejemplo de código (Capturas)** 7](#_Toc159082251)

[- **Ejecución del código** 7](#_Toc159082252)

[**4.** **Conclusiones** 8](#_Toc159082253)

[**5.** **Recomendaciones** 8](#_Toc159082254)

[**6.** **Bibliografía** 8](#_Toc159082255)

# **Introducción**

En el vasto campo de la ciencia de la computación, donde se enfrentan desafíos computacionales de diversas magnitudes y complejidades, el análisis de algoritmos emerge como un pilar fundamental. Este análisis no solo busca comprender cómo funcionan los algoritmos, sino también evaluar su eficiencia en términos de tiempo y espacio, dos recursos preciosos en cualquier sistema computacional. Desde la resolución de problemas simples hasta la gestión de enormes conjuntos de datos en aplicaciones de inteligencia artificial o análisis de big data, la eficiencia de un algoritmo puede marcar la diferencia entre el éxito y el fracaso de un proyecto.

Las técnicas de análisis de algoritmos abarcan una variedad de enfoques, desde el análisis teórico hasta la experimentación práctica. En el ámbito teórico, se emplean herramientas matemáticas y modelos abstractos para entender el comportamiento de los algoritmos en diferentes situaciones. Esto puede implicar el cálculo de complejidades temporales, como el tiempo de ejecución en el peor caso, el caso promedio o el mejor caso, así como la evaluación de la complejidad espacial, es decir, la cantidad de memoria que utiliza un algoritmo en función del tamaño de entrada.

Además del análisis teórico, la evaluación empírica juega un papel crucial en la comprensión de la eficiencia de los algoritmos. Esto implica implementar los algoritmos y medir su rendimiento en situaciones reales o simuladas, utilizando herramientas y técnicas especializadas para registrar y analizar datos. La experimentación práctica proporciona información valiosa sobre el rendimiento real de los algoritmos en diferentes entornos y condiciones de entrada, lo que complementa y valida los resultados obtenidos mediante análisis teóricos.

La importancia del análisis de algoritmos y la evaluación de su eficiencia no puede ser subestimada en el desarrollo de software y la optimización de sistemas computacionales. Un algoritmo eficiente puede reducir significativamente el tiempo de procesamiento, mejorar la escalabilidad del sistema y reducir los costos operativos asociados con el uso de recursos computacionales. Por otro lado, un algoritmo ineficiente puede llevar a retrasos en la ejecución, consumo excesivo de recursos y, en última instancia, a una experiencia deficiente para el usuario final (Diaz, 2004).

# **Objetivos**

# **Objetivo General:**

Comprender y mejorar el rendimiento de los algoritmos en sistemas computacionales mediante la aplicación de técnicas de análisis teórico y evaluación empírica, con el fin de optimizar el tiempo de ejecución y el consumo de recursos, garantizando así la eficiencia y escalabilidad de las soluciones de software en una variedad de contextos y aplicaciones.

# **Objetivos Específicos:**

Realizar un análisis teórico exhaustivo para calcular la complejidad temporal y espacial de los algoritmos en diferentes situaciones, incluyendo el peor caso, el caso promedio y el mejor caso. Esto implica identificar y comprender cómo varía el tiempo de ejecución y el consumo de memoria a medida que cambian los tamaños de entrada y otros parámetros relevantes.

Realizar experimentos prácticos para evaluar el rendimiento real de diferentes algoritmos al resolver el mismo problema. Esto implica implementar los algoritmos, ejecutarlos en conjuntos de datos representativos y medir métricas de rendimiento como el tiempo de ejecución y el uso de memoria. Luego, se comparan y analizan los resultados para identificar qué algoritmo es más eficiente en términos de tiempo y recursos.

Utilizar los resultados del análisis teórico y experimental para identificar áreas de mejora en los algoritmos, así como oportunidades para optimizar su rendimiento. Esto puede incluir la aplicación de técnicas de optimización algorítmica, como la eliminación de redundancias, la mejora de estructuras de datos o la paralelización de tareas, con el objetivo de reducir el tiempo de ejecución y el consumo de recursos sin comprometer la corrección o la calidad del resultado.

# **Desarrollo:**

# **Técnicas de Análisis de Algoritmos.**

El análisis de algoritmos es una parte importante de la teoría de la complejidad computacional, que proporciona una estimación teórica de los recursos necesarios de un algoritmo para resolver un problema computacional específico. La mayoría de los algoritmos están diseñados para trabajar con entradas de longitud arbitraria. El análisis de algoritmos es la determinación de la cantidad de recursos de tiempo y espacio necesarios para ejecutarlo (Baez, 2024).

Por lo general, la eficiencia o el tiempo de ejecución de un algoritmo se establece como una función que relaciona la longitud de entrada con el número de pasos, conocido como complejidad de tiempo o volumen de memoria, conocido como complejidad de espacio .

Complejidad Algorítmica.

¿De qué hablamos cuando hablamos de complejidad? Resulta evidente que el tiempo real requerido por una computadora para la ejecución de algoritmo es directamente proporcional al número de operaciones básicas que la computadora debe realizar en su ejecución. Medir por lo tanto el tiempo real de ejecución equivale a medir el número de operaciones elementales realizadas. Desde ahora supondremos que todas las operaciones básicas se ejecutan en una unidad de tiempo. Por esta razón se suele llamar tiempo de ejecución no al tiempo real físico, sino al número de operaciones elementales realizadas. Otro de los factores importantes, en ocasiones decisivo, para comparar algoritmos es la cantidad de memoria del computador requerida para almacenar los datos durante el proceso. La cantidad de memoria utilizada durante el proceso se suele llamar espacio requerido por el algoritmo (Baez, 2024)

Al no ser única la manera de representar un algoritmo mediante un programa, y al no ser único el computador en el cual se ejecutará, resulta que la medida del tiempo será variable dependiendo fundamentalmente de los siguientes factores:

1) El lenguaje de programación elegido

2) El programa que representa

3) El computador que lo ejecuta

Por eso surge la necesidad de medir el tiempo requerido por un algoritmo independientemente de su representación y del computador que lo ejecute.

El análisis del algoritmo es el proceso de analizar la capacidad de resolución de problemas del algoritmo en términos del tiempo y el tamaño requeridos (el tamaño de la memoria para el almacenamiento durante la implementación). Sin embargo, la principal preocupación del análisis de algoritmos es el tiempo o rendimiento requerido. En general, realizamos los siguientes tipos de análisis:

* El peor de los casos: el número máximo de pasos dados en cualquier instancia de tamaño N.
* El mejor caso : el número mínimo de pasos dados en cualquier instancia de tamaño N.
* El caso promedio : un número promedio de pasos dados en cualquier instancia de tamaño N.
* El amortizado : una secuencia de operaciones aplicadas a la entrada de tamaño promediada en el tiempo.

Para resolver un problema, debemos tener en cuenta el tiempo y la complejidad del espacio, ya que el programa puede ejecutarse en un sistema donde la memoria es limitada pero hay suficiente espacio disponible o viceversa. En este contexto, si comparamos el ordenamiento de burbuja y el de fusión. El ordenamiento de burbuja no requiere memoria adicional, pero el ordenameinto por fusión requiere espacio adicional. Aunque la complejidad temporal del ordenamiento de burbuja es mayor en comparación con el ordenamiento de fusión, es posible que necesitemos aplicar el ordenamiento de burbuja si el programa necesita ejecutarse en un entorno, donde la memoria es muy limitada. (Baez, 2024)

Para medir el consumo de recursos de un algoritmo, se utilizan diferentes estrategias

* Análisis asintótico
* Resolución de ecuaciones de recurrencia (o recursivas)
* Análisis Amortizado

# **Eficiencia de los algoritmos**

La eficiencia de un algoritmo es una medida de los recursos utilizados para su ejecución, estos son básicamente de dos tipos: el primero es el tiempo necesario para el proceso ("time complexity") y el segundo la memoria utilizada ("space complexity"). En la medida que el algoritmo utilice una menor cantidad de estos recursos es más eficiente (Spisto, 2021)

En si la eficiencia de un algoritmo es crucial para determinar su utilidad en aplicaciones del mundo real un algoritmo puede dar la solución correcta a un problema, pero si requiere demasiado tiempo o memoria para ejecutarse, probablemente no sea práctico implementarlo.

# **¿Cómo se puede analizar la eficiencia de los algoritmos?**

El análisis de algoritmos te permitirá constatar si las elecciones fueron óptimas para dar solución al problema computacional en cuestión. En este escenario existen varias formas que te indicarán cómo medir la eficiencia de un algoritmo. (Gomez, 2022)

|  |  |
| --- | --- |
| Te sugerimos que la incógnita a resolver sea cómo o qué tanto aumenta el tiempo en el que se ejecuta la función si es mayor la entrada. | Es decir, cómo responde el algoritmo si se incrementa la cantidad de información que debe procesar, toda vez que es posible que tome poco tiempo con entradas menores, pero que, al mismo tiempo, su performance se vea afectada al hacer el trabajo con un volumen de información mucho más grande. |
| No es suficiente con evaluar los tiempos destinados a la ejecución de las funciones para las cuales fue creado el algoritmo, pues aquí hay otros factores que entran directamente en la ecuación y podrían hacer variar el resultado | |

# **Pasos para analizar la eficiencia de los algoritmos**

Son varios los criterios que se tienen en cuenta para determinarla, entre ellos la legibilidad, la facilidad de la codificación, el uso de memoria, la portabilidad, etc., lo que se considera un uso eficiente de los recursos y a lo que se le conoce también con el nombre de complejidad o incluso costo de un algoritmo. (Gomez, 2022)

A fin de evaluar qué tanto aumenta el tiempo de ejecución, realizaremos las siguientes acciones

|  |  |
| --- | --- |
| Encontrar un valor estimado respecto de la frecuencia de ejecución | Determinar el número de datos de entrada, su estructura y distribución, además de la calidad del código de fuente y de la máquina |

Después de esto, podrás echar mano de los casos que te llevarán calcular la eficiencia de los algoritmos. Para obtenerlos tendrás que contar el número de operaciones que ejecutó el algoritmo para finalmente llegar al valor que le fue solicitado al inicio. (Gomez, 2022)

|  |  |
| --- | --- |
| **Peor caso:** | Ayuda a plantear un escenario en el que las acciones estarán encaminadas sí o sí a reducir el tiempo que toma en la ejecución de la función. |
| **Caso promedio:** | Carece de los elementos suficientes para plantear un solo escenario de este tipo, pues son muchas las variables involucradas en la práctica. |
| **Mejor caso:** | Es el menos utilizado, justamente por presentar un panorama que escapa de la realidad. |
| **Amortizado:** | Contempla el promedio de tiempo que lleva al algoritmo completar la función tras varias ejecuciones que ocurren sucesivamente |
| **Probabilístico:** | Como su nombre lo indica, toma las probabilidades que arrojan las entradas sobre las cuales se hace el análisis. |

# **Ejemplo de código (Copiado)**

|  |
| --- |
| package tarea1u3;  public class Tarea1U3 {  // Algoritmo 1 - Menos eficiente  public static void sumaArregloV1(int[] arr) { //Declaración de función sumaArregloV1  for (int i = 0; i < arr.length; i++) { //Primer ciclo for que recorre arreglo  for (int j = 0; j < arr.length; j++) { //Segundo ciclo for anidado  arr[i] = arr[i] + arr[j]; //Suma elemento actual con cada elemento  }  }  }  // Algoritmo 2 - Más eficiente  public static void sumaArregloV2(int[] arr) { //Declaración de función sumaArregloV2  int suma = 0; //Variable para guardar suma  for (int i = 0; i < arr.length; i++) { //Primer ciclo para calcular suma  suma += arr[i];  }  for (int i = 0; i < arr.length; i++) { //Segundo ciclo  arr[i] = arr[i] + suma; //Suma elemento actual con suma total  }  }  public static void main(String[] args) {  // Arreglo de tamaño mayor  int[] arr = new int[27000]; //Declaración e inicialización de arreglo grande  // Inicialización del arreglo grande  for (int i = 0; i < arr.length; i++) {  arr[i] = i; //Inicializa elementos con valores de 0 a 499,999  }  long inicio = System.currentTimeMillis(); //Tiempo inicial antes de V1  sumaArregloV1(arr); //Llama a función sumaArregloV1  long fin = System.currentTimeMillis(); //Tiempo final después de V1  System.out.println("Tiempo V1: " + (fin - inicio) + " ms"); //Imprime tiempo V1  inicio = System.currentTimeMillis(); //Tiempo inicial antes de V2  sumaArregloV2(arr); //Llama a función sumaArregloV2  fin = System.currentTimeMillis(); //Tiempo final después de V2  System.out.println("Tiempo V2: " + (fin - inicio) + " ms"); //Imprime tiempo V2  }  } |

# **Ejemplo de código (Capturas)**

|  |
| --- |
|  |

# **Ejecución del código**

|  |
| --- |
|  |

# **Conclusiones**

* El análisis de algoritmos y la evaluación de su eficiencia son fundamentales para el desarrollo de software de alto rendimiento. Comprender cómo se comportan los algoritmos en términos de tiempo de ejecución y uso de recursos permite tomar decisiones informadas sobre la selección y diseño de algoritmos para resolver problemas específicos.
* Tanto el análisis teórico como la evaluación empírica son importantes para comprender la eficiencia de los algoritmos. Mientras que el análisis teórico proporciona una comprensión profunda de la complejidad algorítmica en diferentes escenarios, la evaluación empírica valida y complementa estos resultados mediante pruebas prácticas en entornos reales.
* El análisis de algoritmos no termina con la selección inicial de un algoritmo eficiente. Es crucial continuar evaluando y optimizando los algoritmos a medida que evolucionan los requisitos del proyecto y cambian las condiciones de operación. La optimización continua garantiza que el software mantenga un rendimiento óptimo a lo largo del tiempo.

# **Recomendaciones**

* Emplear herramientas de análisis de rendimiento y técnicas de profiling para identificar cuellos de botella y áreas de mejora en el código. Estas herramientas proporcionan información detallada sobre el tiempo de ejecución y el uso de recursos, facilitando la identificación de oportunidades de optimización.
* Incorporar principios de diseño de algoritmos eficientes y utilizar estructuras de datos adecuadas puede tener un impacto significativo en la eficiencia del software. Se deben preferir algoritmos y estructuras de datos con complejidades temporales y espaciales óptimas para el problema en cuestión.
* Al evaluar la eficiencia de los algoritmos, es importante tener en cuenta la escalabilidad y la robustez del software en entornos de producción. Los algoritmos eficientes deben ser capaces de manejar grandes volúmenes de datos y adaptarse a cambios en las condiciones de operación sin degradación significativa del rendimiento.

# **Bibliografía**

Gomez, I. (2022). Obtenido de crehana: https://www.crehana.com/blog/transformacion-digital/eficiencia-de-los-algoritmos/

Spisto, F. (2021). Obtenido de quora: https://es.quora.com