Tiempo de Ejecución de un Algoritmo

Bonilla Caiza David Alejandro

Andrade Chicaiza Julio Aaron

Quiroz Herrera María Laura

Tiupul Guamán Danny Alexander

Universidad de las Fuerzas Armadas ESPE

28436: Estructuras de Datos

Ing. Washington Loza H. Mgs.

22 de enero del 2025

Índice

Introducción

El análisis de algoritmos forma parte fundamental del estudio de la ingeniería de software, nos ayuda a evaluar el desempeño y eficiencia de nuestros algoritmos antes de implementarse en la práctica. Entre todos los aspectos que se pueden analizar el tiempo de ejecución es uno de los más importantes, ya que cuantifica que tan eficiente es la solución al enfrentarse a diferentes tamaños de datos de entrada y escenarios de uso.

El tiempo de ejecución se refiere al número de operaciones elementales que un algoritmo realiza según el tamaño de los datos de entrada que proporcione el usuario, independientemente del hardware o lenguaje de programación que se utilice. Su estudio nos permite comparar algoritmos y a la par identificar cuellos de botella, de esta manera poder seleccionar el algoritmo que mas se adecua a nuestras necesidades como ingenieros de software. Entender como el tiempo de ejecución crece a medida del aumento del tamaño de los datos de entrada es sumamente importante para diseñar algoritmos óptimos y eficientes.

En el presente informe se investiga y sobre todo, aprende sobre el tiempo de ejecución de un algoritmo, describiendo los objetivos del equipo como resultado de esta investigación y a la par desarrollar los conceptos teóricos del tema a investigar, analizando paso a paso cada aspecto importante del mismo y desarrollando el análisis de un ejemplo practico para entender mucho más fácil el tema.

Objetivos

Objetivo General

Investigar que es el tiempo de ejecución de un algoritmo, sus principales conceptos teóricos, temas auxiliares importantes, con el fin de comprender como funciona y para qué sirve.

Objetivos Específicos

Definir el concepto de tiempo de ejecución de un algoritmo y explicar su importancia dentro del análisis de algoritmos.

Identificar los factores que influyen en el tiempo de ejecución, considerando el tamaño de la entrada y las operaciones básicas del algoritmo.

Identificar los factores que influyen en el tiempo de ejecución, considerando el tamaño de la entrada y las operaciones básicas del algoritmo.

Desarrollo

Concepto de Tiempo de Ejecución de un Algoritmo

Definición formal de tiempo de ejecución

El tiempo de ejecución es la cantidad de tiempo que un algoritmo requiere para completar su ejecución sobre un conjunto de datos de entrada específicos. Formalmente, el tiempo de ejecución se cuantifica como una función $T(n)$ del tamaño de entrada n, representando cuántas operaciones elementales realiza el algoritmo a medida que se procesa la entrada. (Wikipedia,2025)

Entender esta definición es sumamente importante para comparar distintas soluciones y conforme a el tamaño de los datos de entrada predecir el comportamiento de la misma. En muchos libros se distinguen distintos tipos de análisis temporal, tales como el peor caso, el mejor caso y el caso promedio, estos permiten comprender no solo cuanto tiempo llega a tardar un algoritmo, sino también como varia en el tiempo dependiendo de los datos que ingresen. (Campos, 2025)

Factores que influyen en el tiempo de ejecución

El tiempo de ejecución de un algoritmo no depende únicamente de la función teórica $T(n)$, existen diversos factores tanto internos y externos que afectan cuánto tarda un algoritmo en ejecutarse, entre ellos tenemos:

- Tamaño de los datos de entrada: El número de elementos o la estructura de los datos influyen directamente en el número de operaciones que el algoritmo debe realizar. Entradas más grandes o con características particulares (ordenadas o desordenadas) pueden provocar variaciones significativas en el tiempo de ejecución de un algoritmo, aunque la función $T(n)$ sea la misma. (Gil, s.f.)

- Diseño del algoritmo: La forma en que un algoritmo está estructurado, por ejemplo, si usa bucles simples, bucles anidados o recursión, determina cuántas operaciones se ejecutan y con qué frecuencia. Diseños más eficientes tienden a ejecutar menos operaciones para lograr el mismo resultado. (Muñiz, s.f.)

- Calidad del código generado por el compilador: La eficiencia del código resultante depende de cómo el compilador traduce el algoritmo al lenguaje máquina. Compiladores con mejores optimizaciones suelen producir código más eficiente, reduciendo operaciones innecesarias y mejorando el tiempo de ejecución. (Muñiz, s.f.)

- Recursos hardware y sistema de ejecución: Factores como la velocidad del procesador, la cantidad de memoria RAM disponible, la arquitectura de la máquina y la carga del sistema operativo afectan el tiempo real que toma ejecutar un algoritmo, aunque estos factores no se consideran en el análisis teórico de $T(n)$. (SDGE, 2010)

Aplicación Formal del Análisis del Tiempo de Ejecución

Cálculo del tiempo de ejecución en función de n

El tiempo de ejecución se expresa como una función T(n), donde n representa el tamaño de los datos de entrada (por ejemplo, el número de elementos en un arreglo).

Para calcular T(n) se siguen estos pasos teóricos:

- Se identifican las operaciones básicas del algoritmo.
- Se determina cuántas veces se ejecuta cada operación en función de n.
- Se suman todas las operaciones para obtener una función total.

Expresión matemática del tiempo de ejecución

La expresión matemática que describe el tiempo de ejecución se representa a través de notaciones asintóticas, las cuales nos ayudan a entender cómo se comporta un algoritmo cuando el tamaño de la entrada, n, es grande:

O grande (O): esto se refiere al peor de los casos, es decir, el límite superior del tiempo de ejecución.

Ω (Omega): representa el mejor de los casos, el límite inferior.

Θ (Theta): se utiliza para el caso promedio o exacto, cuando los límites superior e inferior son iguales.

Un ejemplo general sería: $T(n) = 3n^2 + 5n + 10$

En el análisis asintótico, se eliminan los términos constantes y de menor grado, quedando así: $T(n) \in O(n^2)$

Esto nos indica que el tiempo de ejecución aumenta de manera proporcional al cuadrado del tamaño de la entrada.

Justificación del análisis realizado

El análisis de tiempo de ejecución nos permite:

Comparar algoritmos de forma objetiva.

Prever cómo se comportará un algoritmo con grandes volúmenes de datos.

Evaluar la escalabilidad y la eficiencia de una solución.

Tomar decisiones de diseño sin necesidad de ejecutar el programa.

Este análisis se basa en supuestos teóricos, como que todas las operaciones básicas tienen un costo constante y que el tamaño de entrada n es el factor más influyente. Aunque no refleja con exactitud los tiempos reales, sí describe de manera

precisa cómo crece el costo computacional, lo cual es clave para diseñar algoritmos eficientes.

Técnicas para el Análisis del Tiempo de Ejecución

Mediante estas técnicas, se pueden estimar el costo computacional que va a implicar la ejecución de un algoritmo. Estas se centran en analizar el comportamiento del algoritmo frente a condiciones variables.

Estas técnicas permiten evaluar el número de operaciones ejecutadas, comparar los algoritmos de manera objetiva y analizar el comportamiento del algoritmo al ingresar datos muy grandes.

Modelo de máquina abstracta

Según Aho, Hopcroft y Ullman (1988), una máquina abstracta "es un modelo matemático que define un conjunto de operaciones permitidas y sus costos asociados, permitiendo al analista predecir el tiempo de ejecución de un algoritmo en términos de la entrada".

Este modelo permite analizar la ejecución de un algoritmo independientemente de una arquitectura física específica. La más utilizada es la máquina de acceso aleatorio (RAM), esta proporciona una abstracción muy cerca y real a una computadora con la finalidad de realizar un análisis matemático exhaustivo y riguroso.

Operaciones básicas y conteo de instrucciones

Para realizar un adecuado análisis de la complejidad y tiempos de ejecución de un algoritmo, es de suma importancia considerar y evaluar con mucho cuidado todas las operaciones elementales que se encuentran en el mismo.

Las operaciones elementales son pequeñas acciones que realiza el procesador en un determinado tiempo mientras el algoritmo se encuentra en ejecución. Estas operaciones se clasifican en asignaciones, operaciones aritméticas, comparaciones lógicas y accesos a memoria.

Sedgewick y Wayne (2011) definen las operaciones básicas como "las instrucciones primitivas del modelo computacional que se ejecutan en un número fijo de ciclos de reloj, tales como la suma de enteros o la comparación de dos valores"

Para el conteo de instrucciones, se suman la cantidad de veces que cada operación básica se ejecuta considerando el tamaño de la entrada (n), con esto se obtiene el orden de complejidad T(n).

Análisis teórico vs medición empírica

Un análisis teórico permite estudiar el algoritmo matemáticamente, empleando modelos abstractos y conteo de operaciones, lo que facilita analizar los algoritmos eficientemente antes de la implementación. Además de ser generales y no depender del hardware.

La medición empírica permite ejecutar el algoritmo, a la vez de medir su tiempo de ejecución. Es de gran utilidad para validar resultados teóricos. Sin embargo, al depender del lenguaje de programación, compilador y hardware, la medición podría resultar afectada.

Al realizar la comparación entre ambos, se observa que estos se complementan, debido a que el análisis teórico predice y compara, mientras que la medición empírica verifica en la práctica todo lo planteado.

Análisis del Tiempo de Ejecución en Algoritmo de Ejemplo

Algoritmo de ejemplo

```
// Algoritmo de búsqueda binaria en un arreglo ordenado

int busquedaBinaria(int A[], int n, int x) {
    int inicio = 0;
    int fin = n - 1;

    while (inicio <= fin) {
        int medio = (inicio + fin) / 2;

        if (A[medio] == x) {
            return medio;
        } else if (A[medio] < x) {
            inicio = medio + 1;
        } else {
            fin = medio - 1;
        }
    }

    return -1;
}
```

Funcionamiento paso a paso del algoritmo

```
int inicio = 0; 1 OE

int fin = n - 1; 2 OE

Ciclo while 1OE

int medio = (inicio + fin) / 2; 3 OE

if (A[medio] == x) 2OE

else if (A[medio] < x) 2 OE

inicio = medio + 1; 2 OE

fin = medio - 1; 2 OE
```

```
return -1; 1 OE

Armando la ecuación de tiempo de ejecución:

T(n) = (1 + 2 + log2(n) [1 + 3 + 2 + 2 + 2 + 2] + 1) OE

T(n) = (4 + 12 log2(n)) OE

Por lo que su orden de complejidad será de O(log2(n))
```

Conclusión

Concluimos que el cálculo del tiempo de ejecución de un algoritmo permite evaluar su eficiencia y predecir su comportamiento frente a distintos tamaños de entrada, mediante el análisis de las operaciones básicas que lo componen y su expresión como una función $T(n)$. Este análisis facilita comparar algoritmos bajo las mismas condiciones y seleccionar la solución más eficiente conforme aumenta la cantidad de datos a procesar.

Además, también se concluye que el análisis práctico del tiempo de ejecución, mediante la identificación y conteo de operaciones básicas dentro de un algoritmo, permite modelar su comportamiento real y tomar decisiones fundamentadas al momento de seleccionar o diseñar algoritmos más eficientes para problemas concretos.

Bibliografía

Aho, A. V., Hopcroft, J. E., & Ullman, J. D. (1988). *Estructuras de datos y algoritmos*. Addison-Wesley.

Sedgewick, R., & Wayne, K. (2011). *Algorithms* (4th ed.). Addison-Wesley.

Lutkevich, B. (2021b, diciembre 2). runtime. Search Software Quality. https://www-techtarget-com.translate.goog/searchsoftwarequality/definition/runtime?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

Eye on Tech. (2019, 12 diciembre). What is Source Code and What Does it Do? [Vídeo]. YouTube. https://www.youtube.com/watch?v=exUCyp8PbD4

Campos, J. (2025). *Eficiencia de algoritmos: función T(n) y conteo de operaciones*. Universidad de Zaragoza. Recuperado de https://webdiis.unizar.es/~jcampos/ab/material/varios/eficienciaAlgoritmos.pdf

Wikipedia. (2025). *Complejidad temporal*. Recuperado de https://es.wikipedia.org/wiki/Complejidad_temporal

Gil, J. (s.f.). *Complejidad Computacional y Asintótica.* Recuperado de https://www.scribd.com/document/461002523/Complejidad-Computacional-pdf

Muñiz, F. (s.f.). *Estructura de datos*. Recuperado de https://slidetodoc.com/unidad-1-anlisis-de-algoritmos-1-1-concepto

SDGE (s.f.). Tiempo de ejecución y eficiencia de algoritmos. Recuperado de https://matematicas.uam.es/~pablo.angulo/doc/laboratorio/b2s2.html