**UNIVERSIDAD AMERICANA**

**Logotipo

Descripción generada automáticamente**

**Metodología y Programación Estructurada**

***Investigación Divide y Vencerás***

**Estudiantes**

Steven Leonel Sequeira Reyes

Alfredo Josué Sánchez Bojorge

Julio Cesar Delgadillo Pineda

**Fecha:**

Viernes, 11 de octubre del 2024

**Managua, Nicaragua**

Índice

[1. Introducción 3](#_Toc179523330)

[1.1. Definición del paradigma 3](#_Toc179523331)

[1.2. Importancia del paradigma 3](#_Toc179523332)

[2. Estructura 4](#_Toc179523333)

[2.1 Dividir 4](#_Toc179523334)

[2.2 Conquistar 4](#_Toc179523335)

[2.3 Combinar 5](#_Toc179523336)

[3. Algoritmos 5](#_Toc179523337)

[3.1 Merge Sort 5](#_Toc179523338)

[3.2 Quick Sort 7](#_Toc179523339)

[Aplicaciones de Quicksort 7](#_Toc179523340)

[Limitaciones y Consideraciones 7](#_Toc179523341)

[3.3 Búsqueda Binaria 8](#_Toc179523342)

[Funcionamiento del algoritmo 8](#_Toc179523343)

[Aplicaciones 9](#_Toc179523344)

[Ejemplo 9](#_Toc179523345)

[3.4 Algoritmo de Strassen 10](#_Toc179523346)

[Funcionamiento 10](#_Toc179523347)

[Aplicaciones 11](#_Toc179523348)

[Limitaciones 11](#_Toc179523349)

[4. Análisis de Eficiencia 12](#_Toc179523350)

[5. Conclusiones 13](#_Toc179523351)

[6. Bibliografía 14](#_Toc179523352)

# 1. Introducción

En esta investigación se aborda el paradigma de Divide y Vencerás (Divide and Conquer), y se destaca su relevancia y aplicabilidad en el campo de la informática. Se busca explicar el concepto central, el cual consiste en resolver un problema grande dividiéndolo en problemas más pequeños, resolviéndolos de manera independiente hasta llegar a una solución conjunta.

## 1.1. Definición del paradigma

El enfoque "Divide y Vencerás" es una estrategia para resolver problemas complejos, que se basa en descomponerlos en subproblemas más pequeños y fáciles de manejar. Cada subproblema se resuelve de forma recursiva y luego se combinan las soluciones para obtener la respuesta al problema original. Esta técnica es comúnmente empleada en algoritmos eficientes como el merge sort, Quicksort y la búsqueda binaria. La clave de este método radica en la descomposición recursiva, lo que permite enfrentar grandes problemas de manera más estructurada y organizada (Academia Lab, s.f.).

## 1.2. Importancia del paradigma

Divide y Vencerás, es una estrategia fundamental en el ámbito de la resolución de problemas en la informática. Al dividir un problema complejo en subproblemas pequeños y manejables, resuelve cada uno de ellos de manera independiente. Su importancia radica en varios aspectos: Primero, su simplicidad y claridad, ya que, al dividir los problemas, reduce la carga cognitiva y permite centrarse en una tarea más específica, lo cual hace cada problema más fácil de comprender y abordar. Segundo, la optimización de recursos, ya que permite que los subproblemas se puedan resolver en paralelo, y aprovechar los recursos de hardware de manera más eficiente. Este paradigma no solo se aplica en la informática, también tiene diversas aplicaciones en la inteligencia artificial, y en cualquier situación que requiera una gestión y liderazgo.

# 2. Estructura

## 2.1 Dividir

Durante esta etapa, la cuestión inicial se fragmenta en diversos subproblemas de menor tamaño, que tienen la misma naturaleza que el problema original. El objetivo de estos subproblemas es que se vuelvan más manejables y fáciles de resolver. Como ejemplo, en un algoritmo de ordenamiento tal como Merge Sort, el arreglo original se divide en dos partes iguales. Luego se sigue dividiendo hasta que todos los subproblemas se reducen a un problema de tamaño pequeño que se pueda resolver directamente.

Ejemplos:

* En el algoritmo Quick Sort, se elige a un pivote y se reorganiza el arreglo de tal manera que todos los elementos menores queden a su izquierda, y todos los mayores, a su derecha. Si el pivote divide el arreglo de manera equilibrada, el rendimiento será optimo.
* En el algoritmo Merge sort, el problema se divide recursivamente en mitades hasta que cada sublista tiene un solo elemento.

## 2.2 Conquistar

En esta fase, cada subproblema se resuelve de manera independiente, normalmente de manera recursiva. Este paso de combinación es crucial en algoritmos como Merge Sort, donde se van organizando las soluciones a todos los subproblemas (arreglos ordenados), para luego fusionarse como uno solo.

## 2.3 Combinar

Para finalizar, se reúnen las soluciones de los subproblemas para formar una solución al problema original. El paso de combinación debe ser eficiente para que el algoritmo sea efectivo en general. Si este paso toma demasiado tiempo, puede afectar negativamente el rendimiento del algoritmo.

En el merge sort, después de dividir y resolver las dos mitades de la lista, el paso de combinación implica fusionar dos listas ordenadas. Aunque este paso se realiza en tiempo lineal respecto al tamaño de las listas, es fundamental para mantener la eficiencia global del algoritmo.

# 3. Algoritmos

## 3.1 Merge Sort

Merge sort es un algoritmo de ordenación eficiente basado en el paradigma de divide y vencerás. Fue desarrollado por John von Neumann en 1945 y es conocido por su estabilidad y consistencia en el rendimiento, especialmente cuando se trata de grandes conjuntos de datos.

Merge sort sigue los tres pasos principales del paradigma **divide y vencerás**: dividir, resolver (conquistar) y combinar. El algoritmo funciona de la siguiente manera:

* Divide: El conjunto de datos o lista se divide repetidamente en dos mitades hasta que cada sublista contiene un solo elemento. Un solo elemento ya está ordenado por definición.
* Conquer: Cada sublista se ordena recursivamente. Dado que las sublistas de un solo elemento están inherentemente ordenadas, este paso es trivial.
* Combine: Las sublistas ordenadas se combinan de manera que el resultado también esté ordenado. Este proceso de fusión toma dos listas ordenadas y las combina en una lista ordenada, comparando los elementos de ambas listas una a una.

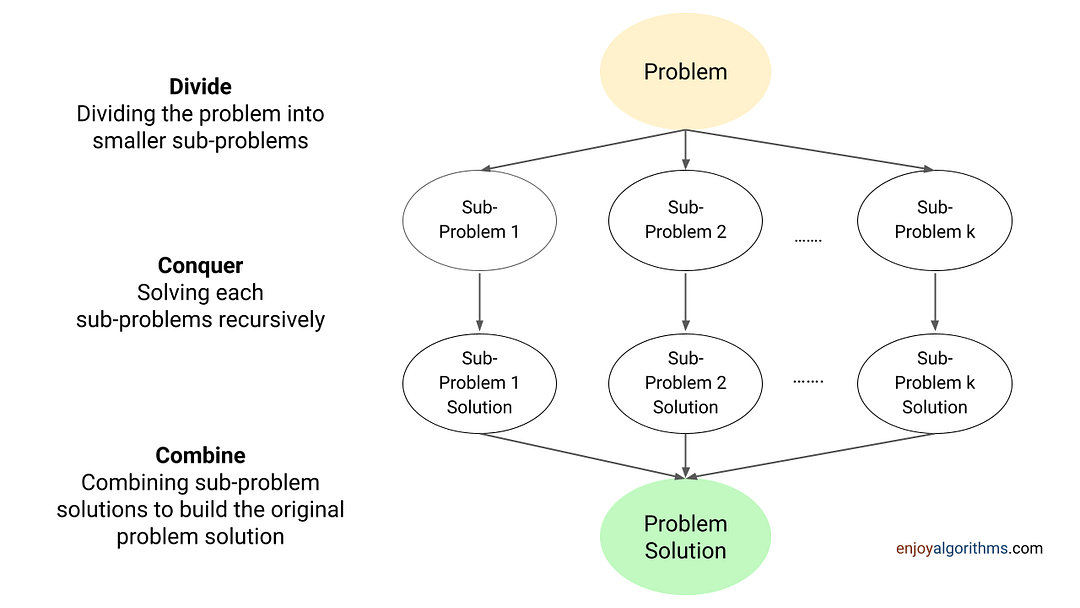
Este enfoque de dividir hasta llegar a sublistas muy pequeñas y luego combinarlas es lo que garantiza que el algoritmo funcione de manera eficiente. (Gautam, 2023)

Ejemplo de su uso:

Ejemplo:

Dada la lista desordenada [38, 27, 43, 3, 9, 82, 10], se divide en:

* [38, 27, 43] y [3, 9, 82, 10]
* [38, 27, 43] se divide en [38] y [27, 43]
* [27, 43] se divide en [27] y [43]
* Este proceso continúa hasta que todas las sublistas contienen un solo elemento.



-Pasos del algoritmo Merge Sort

## 3.2 Quick Sort

Quick Sort es un algoritmo de ordenación ampliamente utilizado en el campo de la informática debido a su eficiencia y simplicidad. Desarrollado por Tony Hoare en 1960, se basa en el paradigma de divide y vencerás. A diferencia de otros algoritmos de ordenación, como Merge Sort, Quick Sort es un algoritmo in-place, lo que significa que realiza la ordenación directamente sobre el arreglo de entrada, sin requerir una cantidad significativa de espacio adicional.

Este aspecto lo convierte en una opción especialmente atractiva para la ordenación de grandes volúmenes de datos, donde el uso eficiente de la memoria es crucial. Quick Sort ha demostrado ser eficaz en la mayoría de las implementaciones prácticas y, por esta razón, es uno de los algoritmos más utilizados en la programación moderna.

### Aplicaciones de Quicksort

En términos de aplicaciones, Quick Sort se utiliza en una amplia gama de entornos, desde la ordenación de datos en sistemas operativos hasta la ordenación de grandes volúmenes de información en bases de datos. Su bajo consumo de memoria y su rendimiento en la práctica lo convierten en una opción predilecta en la mayoría de los lenguajes de programación modernos, como C, Java y Python, donde se implementa en las bibliotecas estándar de ordenación.

### Limitaciones y Consideraciones

Quicksort no es un algoritmo estable, lo que significa que no preserva el orden relativo de los elementos con valores iguales. Esto puede ser una limitación en ciertas aplicaciones donde la estabilidad es importante. Sin embargo, su versatilidad y velocidad lo convierten en uno de los algoritmos de clasificación más utilizados en la práctica.

## 3.3 Búsqueda Binaria

En informática, búsqueda binaria, también conocida como búsqueda de medio intervalo, búsqueda logarítmica o corte binario, es un algoritmo de búsqueda que encuentra la posición de un valor objetivo dentro de una matriz ordenada.

La búsqueda binaria compara el valor objetivo con el elemento central de la matriz. Si no son iguales, se elimina la mitad en la que el objetivo no puede estar y la búsqueda continúa en la mitad restante, tomando nuevamente el elemento central para compararlo con el valor objetivo, y repitiendo esto hasta encontrar el valor objetivo. Si la búsqueda termina con la mitad restante vacía, el objetivo no está en la matriz. (Lin, 2019)

### Funcionamiento del algoritmo

La búsqueda binaria funciona dividiendo el conjunto de datos en mitades repetidamente, lo que reduce el espacio de búsqueda en cada paso. Dado un arreglo o lista ordenada, el algoritmo siempre va a seguir estos pasos básicos:

1. Se comparan el valor buscado con el elemento central de la lista.
2. Si el valor coincide con el elemento central, el algoritmo termina y devuelve la posición de dicho valor.
3. Si el valor es menor que el elemento central, se repite el proceso con la mitad izquierda de la lista.
4. Si el valor es mayor que el elemento central, el proceso se repite con la mitad derecha de la lista.
5. El proceso se continúa hasta que se encuentre el elemento o hasta que no queden más elementos por comparar.

### Aplicaciones

La búsqueda binaria es ampliamente utilizada en una variedad de problemas donde los datos están ordenados, como:

* Búsqueda en bases de datos indexadas: Es fundamental en motores de búsqueda y en sistemas de almacenamiento de datos donde los registros están ordenados.
* Verificación de existencia de elementos: Usado en estructuras de datos como árboles binarios de búsqueda.
* Solución de problemas de optimización: En algunos casos, se puede utilizar la búsqueda binaria para encontrar soluciones a problemas de optimización que involucran intervalos numéricos.
* Algoritmos de división y conquista: Se utiliza como parte de otros algoritmos más complejos que dividen los problemas en partes más pequeñas.

### Ejemplo

Si se tiene el siguiente arreglo ordenado:

A = [1,3,5,7,9,11,13,15]

Y queremos buscar el número 7. El algoritmo de búsqueda binaria funciona de la siguiente manera: Primero, marca el inicio = 0, y el fin = 7, por lo que la mitad seria 3. Se compara A [3] = 7, con el valor buscado, al ser iguales, se devuelve la posición 3 y el algoritmo termina.

## 3.4 Algoritmo de Strassen

El algoritmo de Strassen es un método eficiente para la multiplicación de matrices que mejora el enfoque tradicional, que requiere O(n^3) operaciones. Este algoritmo fue propuesto por Volker Strassen en 1969 y utiliza un enfoque de división y conquista que reduce el número de multiplicaciones necesarias, haciendo que el proceso sea más rápido, especialmente en matrices de gran tamaño.

### Funcionamiento

El algoritmo de Strassen descompone matrices grandes en submatrices más pequeñas. Strassen particiona las matrices A, B y C en matrices de bloque de igual tamaño.

Texto

Descripción generada automáticamente

Strassen luego reduce estas matrices en 7 multiplicaciones nuevas mediante las siguientes combinaciones:

Texto, Carta

Descripción generada automáticamente

Con este resultado, las submatrices se calculan como:

Texto, nombre de la empresa

Descripción generada automáticamente

### Aplicaciones

El algoritmo de Strassen es particularmente útil en situaciones donde se necesita multiplicar matrices de gran tamaño, como en:

* Cálculos científicos: Multiplicación de matrices en simulaciones numéricas y análisis de grandes cantidades de datos.
* Procesamiento de imágenes y gráficos por computadora: Donde el procesamiento eficiente de grandes matrices es esencial.
* Aprendizaje automático: Para el cálculo de redes neuronales y otros algoritmos que dependen de la multiplicación de grandes matrices.

### Limitaciones

Si bien el algoritmo de Strassen mejora el rendimiento para matrices grandes, tiene algunas desventajas:

* Mayor complejidad en la implementación: Comparado con el algoritmo de multiplicación tradicional.
* Precisión: Puede no ser tan preciso en ciertos contextos, debido a la acumulación de errores de redondeo en las operaciones de suma y resta.
* Costos de suma y resta: Las operaciones adicionales pueden ser significativas en matrices pequeñas, donde no se observa una gran ventaja en comparación con el algoritmo ingenuo.

# 4. Análisis de Eficiencia

El paradigma "Divide y Vencerás" es eficiente en la resolución de problemas que pueden dividirse en subproblemas de la misma naturaleza, y luego combinarse sus soluciones parciales para formar la solución completa. La eficiencia del paradigma radica en su capacidad de reducir el tamaño del problema original a subproblemas más manejables y resolverlos recursivamente. Esta estrategia permite que los algoritmos diseñados bajo este paradigma aprovechen mejor los recursos computacionales, especialmente en problemas grandes y complejos.

Cada algoritmo implementado bajo el paradigma "Divide y Vencerás" tiene una complejidad temporal característica, influenciada por la forma en que se divide el problema y cómo se combinan las soluciones. A continuación, se describe la complejidad temporal de algunos algoritmos representativos:

**Merge Sort:**

Mejor caso: O(n log n)

Peor caso: O(n log n)

Caso promedio: O(n log n)

Merge Sort es muy eficiente para ordenar grandes conjuntos de datos, ya que divide el arreglo de manera uniforme y siempre requiere el mismo número de pasos para combinar las soluciones parciales, independientemente de la disposición inicial de los elementos.

**Quick Sort:**

Mejor caso: O(n log n)

Peor caso: O(n²)

Caso promedio: O(n log n)

Quick Sort, aunque tiene un peor caso de O(n²) cuando se selecciona un mal pivote, es extremadamente eficiente en promedio debido a su capacidad de dividir rápidamente el conjunto de datos alrededor del pivote. Su eficiencia es alta en la mayoría de los casos prácticos.

# Conclusiones

El paradigma "*Divide y Vencerás*" se ha consolidado como una técnica fundamental en la resolución de problemas complejos en el ámbito de la informática. A través de la descomposición de problemas en subproblemas más manejables, este enfoque no solo optimiza el tiempo de ejecución de los algoritmos, sino que también facilita su paralelización, lo que es crucial en sistemas modernos con múltiples procesadores. La implementación de algoritmos clásicos como Merge Sort, Quick Sort y búsqueda binaria demuestra la efectividad de este paradigma en la práctica, evidenciando su relevancia en el desarrollo de soluciones eficientes y escalables.

Las ventajas del paradigma "*Divide y Vencerás*" son múltiples y significativas, lo que lo convierte en un enfoque preferido en la resolución de problemas en informática. En primer lugar, la capacidad de dividir un problema en partes más pequeñas permite un enfoque más sistemático y organizado, lo que facilita la identificación de soluciones. Este proceso de descomposición no solo simplifica la resolución de problemas, sino que también permite a los desarrolladores concentrarse en cada subproblema de manera individual, lo que puede llevar a soluciones más innovadoras y efectivas.

Por último, la aplicabilidad de "*Divide y Vencerás*" se extiende más allá de la programación, influyendo en la forma en que se abordan problemas en diversas disciplinas, desde la ingeniería hasta la investigación científica. Esta versatilidad resalta la importancia de enseñar y promover este paradigma en la educación informática, asegurando que las futuras generaciones de profesionales estén equipadas con las herramientas necesarias para enfrentar los desafíos del mundo moderno.

# Bibliografía

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

Gautam. (2023). Divide and conquer.

<https://www.enjoyalgorithms.com/blog/divide-and-conquer>

Lin, A. (2019). Algoritmo de búsqueda binaria. <https://academia-lab.com/enciclopedia/algoritmo-de-busqueda-binaria/>

Javier, E. (2020). Algoritmo Divide y Vencerás. <https://adictosaltrabajo.com/2020/06/10/algoritmia-divide-y-venceras/>

Harshgav. (2023). Comparison among Greedy, Divide and Conquer and Dynamic Programming algorithm. <https://www.geeksforgeeks.org/comparison-among-greedy-divide-and-conquer-and-dynamic-programming-algorithm/>