BENEMÉRITA
UNIVERSIDAD
AUTÓNOMA DE AGUASCALIENTES

Centro de Ciencias Básicas

TEORÍA DE LA COMPLEJIDAD COMPUTACIONAL

Ingeniería en Computación Inteligente

Semestre 6°A

P2T4: INVESTIGACIÓN Fecha: 18 de marzo de 2024

Estudiantes:

Ximena Rivera Delgadillo

José Luis Sandoval Pérez

Diego Emanuel Saucedo Ortega

Daniel Torres Macias

Profesor: Miguel Ángel Meza de Luna

INVESTIGACIÓN DE LA APLICACIÓN DE DIVIDE Y VENCERAS EN LOS 7 PROBLEMAS

El principio de "divide y vencerás" es una estrategia algorítmica fundamental en la informática y la programación, que implica resolver un problema dividiéndolo en subproblemas más pequeños y luego combinando las soluciones de estos subproblemas para obtener la solución final. Esta técnica es especialmente útil en la optimización del tiempo de ejecución de algoritmos.

1. Búsqueda binaria

La búsqueda binaria es un excelente ejemplo de la aplicación del principio de "divide y vencerás". La búsqueda binaria es un algoritmo eficiente para encontrar un elemento específico dentro de una lista ordenada de elementos. Funciona dividiendo repetidamente a la mitad el espacio de búsqueda hasta que se encuentra el elemento deseado o se determina que el elemento no está presente en la lista.

Aquí hay una descripción básica del proceso de búsqueda binaria y cómo se relaciona con "divide y vencerás":

1. Divide: Se comienza con la lista ordenada de elementos. Se divide esta lista en dos partes iguales (o aproximadamente iguales).

2. Conquista: Se determina en cuál de las dos mitades podría estar el elemento buscado. Si el elemento buscado es menor que el elemento central de la lista, entonces la búsqueda se realiza en la mitad inferior. Si es mayor, la búsqueda se realiza en la mitad superior. Este paso se repite recursivamente en la mitad elegida.

3. Combinación: En el contexto de la búsqueda binaria, no hay una combinación explícita de resultados, ya que el objetivo es encontrar un solo elemento. Sin embargo, se puede considerar que la combinación ocurre cuando se encuentra el elemento deseado o cuando se determina que el elemento no está presente en la lista.

La importancia de la búsqueda binaria radica en su eficiencia. A diferencia de otros métodos de búsqueda que pueden requerir una exploración secuencial de todos los elementos de la lista, la búsqueda binaria elimina la mitad de los elementos en cada paso de la búsqueda, reduciendo drásticamente el tiempo requerido para encontrar el elemento deseado. Esto hace que la búsqueda binaria sea especialmente útil cuando se trabaja con grandes conjuntos de datos.

2. Quicksort

El algoritmo de Quicksort es un claro ejemplo de la estrategia de "divide y vencerás". Esta estrategia se basa en dividir un problema en subproblemas más pequeños, resolver cada subproblema de manera independiente y luego combinar las soluciones de los subproblemas para obtener la solución final. En el caso de Quicksort, este algoritmo sigue estos pasos:

1. Divide: Selecciona un elemento como pivote y divide el arreglo en dos subarreglos: uno que contiene elementos menores que el pivote y otro que contiene elementos mayores que el pivote. Este proceso de partición se realiza de manera que los elementos menores que el pivote estén a la izquierda del pivote y los elementos mayores estén a la derecha.

2. Vence: Los subarreglos se ordenan de manera recursiva utilizando Quicksort. Cada subarreglo se considera como un problema más pequeño que el original.

3. Combina: Como Quicksort ordena los elementos in situ (es decir, sin necesidad de utilizar memoria adicional), no se requiere una etapa explícita de combinación. Una vez que los subarreglos se han ordenado, el arreglo completo también estará ordenado.

La importancia de esta estrategia radica en su eficiencia para resolver problemas, especialmente aquellos que pueden dividirse en subproblemas independientes que se pueden resolver de manera más rápida y eficiente que el problema original. Quicksort es uno de los algoritmos de ordenamiento más eficientes en términos de tiempo de ejecución promedio en la mayoría de los casos.

Al dividir el arreglo en cada paso, Quicksort logra reducir el tamaño de los subproblemas de manera exponencial, lo que conduce a una complejidad de tiempo promedio de O(n log n), donde n es el tamaño del arreglo. Esta eficiencia lo convierte en una opción popular para ordenar grandes volúmenes de datos en la práctica. Sin embargo, es importante destacar que la elección del pivote puede influir significativamente en el rendimiento del algoritmo, por lo que una implementación cuidadosa es crucial para garantizar su eficacia en todas las situaciones.

3. Mergue Sort

El algoritmo de ordenación Mergesort es un excelente ejemplo de cómo se aplica este principio. Mergesort es un algoritmo de ordenación eficiente que sigue el enfoque de "divide y vencerás". Funciona de la siguiente manera:

1. Divide la lista no ordenada en dos mitades de tamaño similar.

2. Ordena recursivamente cada mitad.
3. Combina las dos mitades ordenadas para formar una lista ordenada más grande.

La clave de la eficiencia de Mergesort radica en su capacidad para dividir el problema original en subproblemas más pequeños y solucionarlos de forma independiente antes de combinar las soluciones de manera ordenada. Esto garantiza que el algoritmo tenga un rendimiento predecible y eficiente, con una complejidad temporal de O(n log n) en el peor de los casos.

La aplicación del principio de "divide y vencerás" en Mergesort permite abordar problemas de ordenación de manera más sistemática y escalable. Además, Mergesort es especialmente útil cuando se trata de ordenar listas enlazadas o cuando la memoria disponible es limitada, ya que no requiere acceso aleatorio a elementos de la lista, sino que solo necesita enlaces para acceder a elementos consecutivos.

4. **Par de puntos más cercanos**

El problema del par de puntos más cercanos consiste en encontrar la distancia más corta entre dos puntos en un conjunto de puntos dados en un espacio euclidiano. Este problema es fundamental en la geometría computacional y tiene muchas aplicaciones prácticas, como en la computación gráfica, el análisis de datos y la visión por computadora.

El enfoque de "divide y vencerás" es particularmente útil para resolver el problema del par de puntos más cercanos. El algoritmo típico para resolver este problema utilizando esta técnica se puede describir de la siguiente manera:

1. Dividir: Se divide el conjunto de puntos en dos subconjuntos, preferiblemente de igual tamaño, dividiendo el espacio en dos mitades. Esto se puede hacer utilizando una línea vertical que atraviesa el conjunto de puntos.

2. Conquistar: Se resuelve el problema de forma recursiva en cada una de las mitades. Es decir, se encuentra el par de puntos más cercanos en cada una de las mitades.

3. Combinar: Después de resolver los subproblemas en las mitades izquierda y derecha, se encuentra la distancia mínima entre los puntos que están en diferentes mitades. Esto puede hacerse de manera eficiente considerando solo los puntos que están cerca de la línea de división.

4. Seleccionar: Finalmente, se selecciona el par de puntos más cercanos entre todos los pares de puntos considerados en los pasos anteriores.

Este enfoque tiene una complejidad de tiempo de O(n log n), donde "n" es el número de puntos en el conjunto, lo que lo convierte en uno de los algoritmos más eficientes para resolver el problema del par de puntos más cercanos.

5. **Algoritmo de Strassen**

El algoritmo de Strassen es un algoritmo eficiente para la multiplicación de matrices que se basa en el paradigma "divide y vencerás". Este paradigma consiste en resolver un problema dividiéndolo en subproblemas más pequeños, resolver cada subproblema de manera recursiva y luego combinar las soluciones de los subproblemas para obtener la solución al problema original.

En el contexto del algoritmo de Strassen, la multiplicación de matrices se divide en multiplicaciones de submatrices más pequeñas. En lugar de multiplicar directamente las matrices originales, se dividen en submatrices de igual tamaño y se multiplican estas submatrices utilizando multiplicaciones de matrices de menor tamaño. Luego, estas multiplicaciones de submatrices se combinan para formar la matriz resultante.

La importancia del algoritmo de Strassen radica en su eficiencia en términos de tiempo de ejecución. Aunque el algoritmo tiene un mayor costo computacional en comparación con el algoritmo de multiplicación de matrices estándar, que tiene complejidad O(n^3), el algoritmo de Strassen tiene una complejidad asintótica de O(n^log2(7)), que es aproximadamente O(n^2.81). Esto significa que para matrices lo suficientemente grandes, el algoritmo de Strassen puede ser más rápido que el algoritmo estándar de multiplicación de matrices.

La aplicación del paradigma "divide y vencerás" en el algoritmo de Strassen se refleja en cómo divide las matrices en submatrices más pequeñas, resuelve los productos de esas submatrices y luego combina los resultados para obtener la solución final. Esta técnica de dividir el problema en subproblemas más manejables y luego combinar las soluciones es fundamental para el diseño de algoritmos eficientes en muchos campos de la informática, como la optimización, el análisis de algoritmos y la programación paralela.

6. **El Algoritmo Cooley–Tukey de Transformación Rápida de Fourier (FFT)**

El algoritmo Cooley-Tukey de Transformación Rápida de Fourier (FFT) es un excelente ejemplo de la aplicación del paradigma "divide y vencerás". Este paradigma se basa en resolver un problema dividiéndolo en subproblemas más pequeños y luego combinando las soluciones de estos subproblemas para obtener la solución del problema original. Ahora, veamos cómo esto se aplica específicamente en el algoritmo Cooley-Tukey FFT:

1. División del problema: En la FFT, se trata de calcular la transformada discreta de Fourier (DFT) de una secuencia de números. La idea básica es dividir esta secuencia en dos subsecuencias, una consistente en los elementos de índice par y otra en los elementos de índice impar.

2. Resolución de subproblemas más pequeños: Una vez que se dividen las secuencias en subsecuencias más pequeñas, se calculan las transformadas de Fourier de estas subsecuencias de forma recursiva. Esto significa que para cada subsecuencia se repite el mismo proceso, dividiéndola aún más en subsecuencias más pequeñas y calculando sus transformadas de Fourier.

3.Combinación de soluciones: Una vez que se han calculado las transformadas de Fourier de las subsecuencias más pequeñas, estas soluciones se combinan de cierta manera para obtener la transformada de Fourier de la secuencia original. En el algoritmo Cooley-Tukey, se utiliza una estrategia específica para combinar estas soluciones de manera eficiente, lo que implica multiplicarlas por ciertos factores de twiddle (coeficientes complejos) y sumarlas.

La importancia de este enfoque radica en que reduce significativamente la complejidad computacional de calcular la DFT. En lugar de tener una complejidad de tiempo cuadrática (O(n^2)), como sería el caso si se usara directamente la definición de la DFT, el algoritmo Cooley-Tukey FFT reduce la complejidad a O(n log n), donde n es el tamaño de la secuencia original. Esto hace que la FFT sea fundamental en una amplia gama de aplicaciones donde el procesamiento de señales y la computación científica son importantes, como en telecomunicaciones, procesamiento de imágenes, física, y muchas otras áreas.

7. **El algortimo de Karatsuba**

El algoritmo de Karatsuba es un algoritmo de multiplicación de números grandes que utiliza la técnica de "divide y vencerás". Esta técnica consiste en dividir un problema grande en subproblemas más pequeños y luego combinar las soluciones de estos subproblemas para obtener la solución del problema original. La importancia de esta técnica radica en su capacidad para mejorar la eficiencia de algoritmos, especialmente cuando se trata de problemas que pueden ser divididos en partes más pequeñas de manera eficiente.

En el caso específico del algoritmo de Karatsuba, se utiliza la técnica de divide y vencerás para multiplicar números grandes de manera más eficiente que el algoritmo de multiplicación tradicional. En lugar de realizar la multiplicación directamente, el algoritmo de Karatsuba divide los números en partes más pequeñas, multiplica estas partes de manera recursiva y luego combina las soluciones parciales para obtener el resultado final. Esto

reduce el número total de multiplicaciones necesarias y, por lo tanto, mejora la eficiencia del algoritmo.

La aplicación del algoritmo de Karatsuba es particularmente útil en situaciones donde se necesitan multiplicar números grandes, como en la criptografía, el procesamiento de imágenes, el análisis de datos y otras áreas de las ciencias de la computación y la ingeniería donde se manejan cantidades masivas de datos. Al reducir el tiempo de cálculo requerido para la multiplicación de números grandes, el algoritmo de Karatsuba puede mejorar significativamente el rendimiento de estos sistemas. En resumen, la aplicación de la técnica de divide y vencerás en el algoritmo de Karatsuba permite manejar eficientemente la multiplicación de números grandes, lo que es fundamental en numerosos campos de la informática y la ingeniería.

INTEGRANTES	PROBLEMA	ENTENDIMIENTO
• Ximena Rivera Delgadillo • José Luis Sandoval Pérez • Diego Emanuel Saucedo Ortega • Daniel Torres Macias	**Búsqueda binaria**	Muy bien Muy bien Muy bien Muy bien
• Ximena Rivera Delgadillo • José Luis Sandoval Pérez • Diego Emanuel Saucedo Ortega • Daniel Torres Macias	**Quicksort**	Muy bien Muy bien Muy bien Muy bien
• Ximena Rivera Delgadillo • José Luis Sandoval Pérez • Diego Emanuel Saucedo Ortega • Daniel Torres Macias	**Mergue Sort**	Muy bien Muy bien Muy bien Muy bien
• Ximena Rivera Delgadillo • José Luis Sandoval Pérez • Diego Emanuel Saucedo Ortega • Daniel Torres Macias	**Par de puntos más cercanos**	Muy bien Muy bien Muy bien Muy bien
• Ximena Rivera Delgadillo • José Luis Sandoval Pérez • Diego Emanuel Saucedo Ortega • Daniel Torres Macias	**Algoritmo de Strassen**	Muy bien Muy bien Muy bien Muy bien
• Ximena Rivera Delgadillo • José Luis Sandoval Pérez • Diego Emanuel Saucedo Ortega • Daniel Torres Macias	**El Algoritmo Cooley–Tukey de Transformación Rápida de Fourier (FFT)**	Muy bien Muy bien Muy bien Muy bien
• Ximena Rivera Delgadillo • José Luis Sandoval Pérez • Diego Emanuel Saucedo Ortega • Daniel Torres Macias	**El algortimo de Karatsuba**	Muy bien Muy bien Muy bien Muy bien