Algorítmica

Práctica 2

Algoritmos Divide y Vencerás

Trabajo realizado por:	León Elliott Fuller Toni Martí Albons Alejandro Madueño Buciegas

Índice:

Método Básico (Fuerza Bruta) **3**
Introducción 3
Eficiencia Teórica 3
Divide y Vencerás - Método 1 **4**
Introducción 4
Eficiencia Teórica 4
Divide y Vencerás - Método 2 **6**
Introducción 6
Eficiencia Teórica 6
Comparación entre Método 1 y Método 2 **7**
Comparación Método Básico y Divide y Vencerás **8**
Compilación y ejecución **10**

Método Básico (Fuerza Bruta)

Introducción

Disponemos de un algoritmo básico iterativo que resuelve el problema planteado.Si analizamos el problema que se nos plantea podemos deducir que es necesario comparar todos los puntos entre sí, y para dicha comparación es necesario consultar todos los valores de las coordenadas de estos puntos. Por lo tanto una posible implementación podría ser:

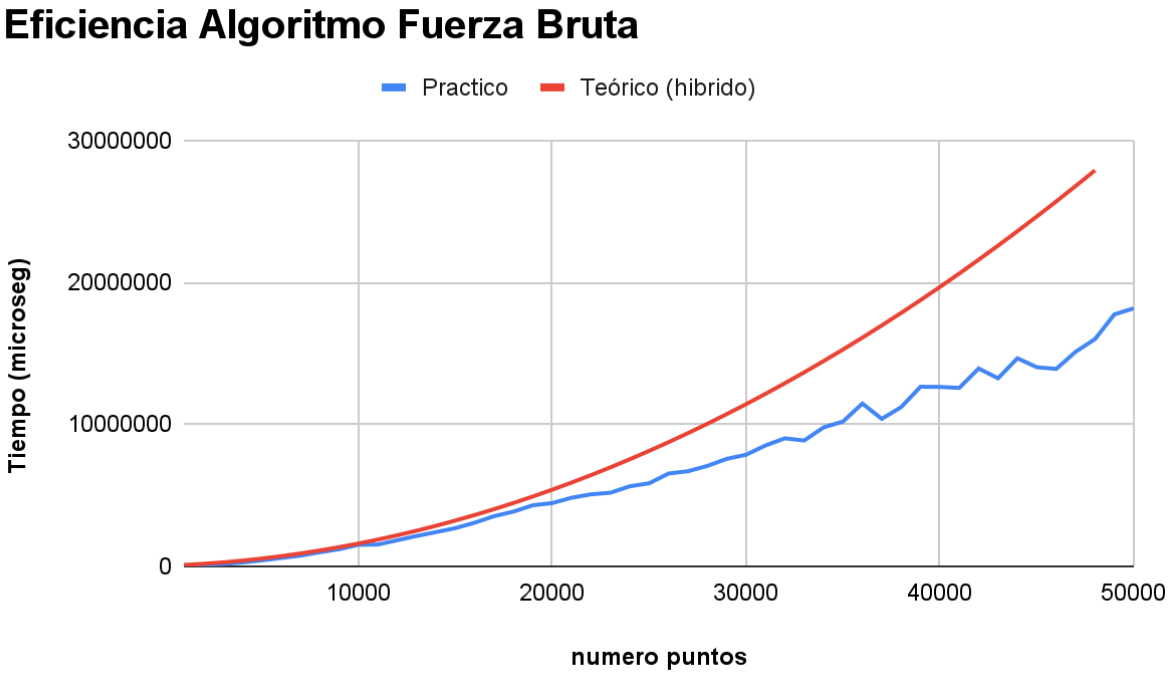
```
/**
 * @brief Encuentra los puntos no dominados en un conjunto de
puntos.
 * @return std::vector con los puntos no dominados en el conjunto.
 */
std::vector<Punto> noDominados(const std::vector<Punto>& puntos) {
    std::vector<Punto> no_dominados;
    for (const Punto & candidato : puntos) {
      bool dominado = false;

      for (const Punto & posible_dominante : puntos) {
            if (posible_dominante > candidato) {
                  dominado = true;
                  break;
            }
      }

      if (not dominado) {
            no_dominados.push_back(candidato);
      }
    }
    return no_dominados;
}
```

Eficiencia Teórica

En base a la implementación dada, podemos deducir que la eficiencia teórica de dicho algoritmo es de orden $O(n^2)$ ya que todas las operaciones hechas dentro de los bucles son de orden constante (incluso la inserción en el vector, ya que hemos reservado previamente toda la memoria que podría necesitar) y los bucles se ejecutan cada uno tantas veces como hay número de puntos.



Divide y Vencerás - Método 1

Introducción

El método 1 del algoritmo Divide y Vencerás que hemos implementado para encontrar los puntos no dominados en un conjunto de datos multidimensionales sigue los siguientes pasos:

1. Si el conjunto de puntos tiene un solo elemento, ese punto es un punto no dominado y se devuelve como solución, finalizando así la ejecución del método, en caso contrario continua.
2. Dividir el conjunto de puntos en dos mitades.
3. Encontrar los puntos no dominados en cada subconjunto de manera recursiva utilizando el mismo algoritmo.
4. Fusionar las soluciones de los subproblemas para obtener el conjunto de puntos no dominados del conjunto original.

El algoritmo Divide y Vencerás aprovecha la estructura del problema de búsqueda de puntos no dominados al reducir el número de comparaciones necesarias para determinar si un punto es dominado o no. Al dividir el conjunto de puntos en subconjuntos más pequeños, podemos comparar menos puntos entre sí y, por lo tanto, reducir el tiempo de ejecución del algoritmo.

Eficiencia Teórica

Vamos a analizar la eficiencia del código propuesto.

Primero, analizamos la función fusionar:

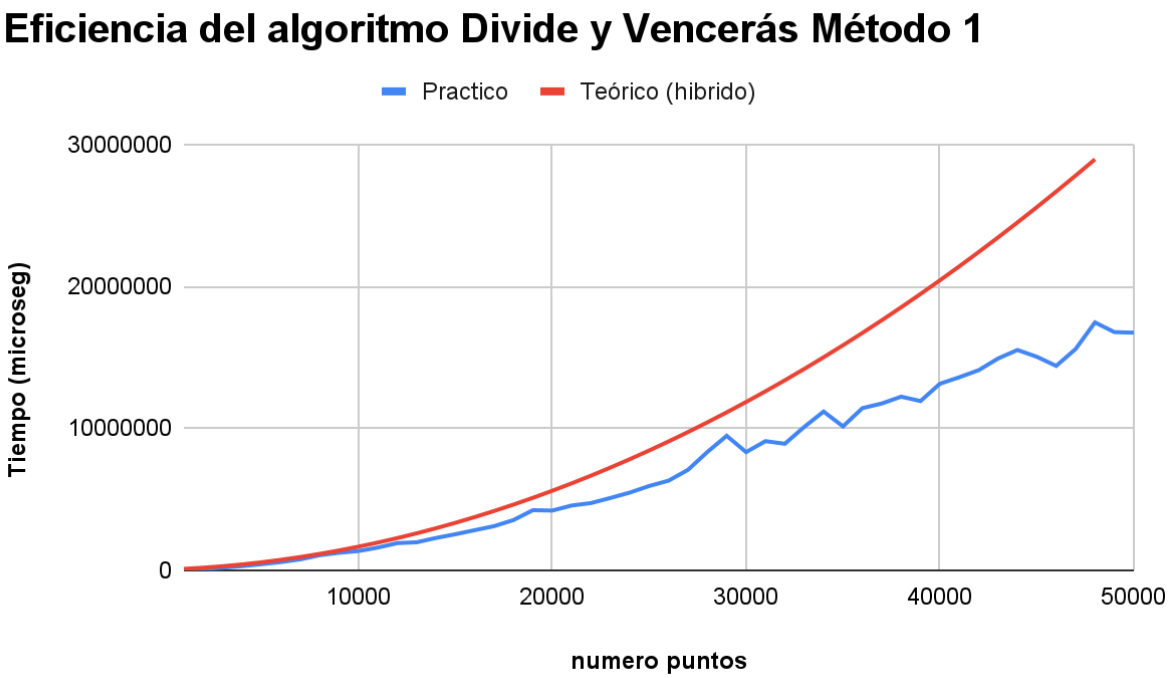
1. Para cada punto en A, se compara con todos los puntos en B, lo que implica una complejidad de O(A * B * K).
2. Para cada punto en B, se compara con todos los puntos en A, lo que implica una complejidad de O(A * B * K).

La complejidad total de la función fusionar es O(A * B * K), siendo K el número de dimensiones.

Ahora analizamos la función divide_venceras:

1. Si la cantidad de puntos es 1, la función retorna inmediatamente el conjunto de puntos, lo que implica una complejidad de O(1).
2. Se divide el conjunto de puntos en dos subconjuntos de tamaño aproximadamente igual, lo que implica una complejidad de O(1).
3. Se resuelve el problema en cada subconjunto de manera recursiva:
    a. Para el subconjunto izquierdo: T(n/2)
    b. Para el subconjunto derecho: T(n/2)
4. Se fusionan las soluciones de los subproblemas con la función fusionar, lo que implica una complejidad de $O(n^2*K)$, porque en el peor de los casos, A y B tendrán n/2 puntos cada uno, y la función fusionar tiene una complejidad de O(A * B * K), siendo K el número de dimensiones.

La relación de recurrencia para la función divide_venceras es:
$T(n) = 2 * T(n/2) + O(n^2*K)$
La solución de esta relación de recurrencia es $T(n) = O(n^2*K)$, así que la eficiencia de nuestro algoritmo es $O(n^2*K)$



Divide y Vencerás - Método 2

Introducción

Es una variante del algoritmo Divide y Vencerás para encontrar los puntos no dominados en un conjunto de puntos en un espacio K-dimensional. Este algoritmo busca encontrar los puntos que no son dominados por ningún otro punto en el conjunto, es decir, aquellos puntos para los cuales no existe otro punto que tenga mejores valores en todas las dimensiones.

El algoritmo se basa en dividir el conjunto de puntos en dos subconjuntos de igual tamaño, resolver el problema de encontrar los puntos no dominados en cada subconjunto de manera recursiva y luego combinar los resultados de ambos subconjuntos utilizando la función fusiona_no_dominados.

La función fusiona_no_dominados toma como entrada dos vectores de puntos no dominados, A y B, y devuelve un vector que contiene todos los puntos no dominados de ambos conjuntos. Para hacer esto, itera sobre los puntos del conjunto B y verifica si cada punto es dominado por algún punto en el conjunto de puntos no dominados. Si el punto de B no es dominado, se agrega al conjunto de puntos no dominados. Si el punto de B domina a un punto no dominado, se elimina el punto dominado del conjunto.

Eficiencia Teórica

Para analizar la eficiencia teórica del algoritmo que utiliza Divide y Vencerás, primero analicemos las funciones fusiona_no_dominados y noDominados.

La función merge_skyfusiona_no_dominadosline tiene una complejidad de tiempo en el peor caso de O(n^2*K), donde n es la cantidad total de puntos en los conjuntos A y B y K la dimensión del espacio. Esto se debe a que, en el peor caso, para cada punto en B, se debe iterar sobre todos los puntos en el conjunto no_dominados, que en el peor caso podría ser de tamaño n.

La función noDominados es una función recursiva de Divide y Vencerás que divide el conjunto de puntos en dos subconjuntos de igual tamaño y resuelve el problema de encontrar los puntos no dominados en cada subconjunto de manera recursiva. La función tiene la siguiente relación de recurrencia:

T(n) = 2 * T(n/2) + O(n^2*K)

Donde T(n) representa la cantidad de tiempo que tarda el algoritmo en procesar n puntos. La relación de recurrencia se deriva de la división del conjunto de puntos en dos subconjuntos de igual tamaño y la combinación de los resultados utilizando la función fusiona_no_dominados.
Entonces, la complejidad de tiempo total del algoritmo que utiliza Divide y Vencerás es:

T(n, K) = O(n^2 * K)

Donde n es el número de puntos en el conjunto y K es la cantidad de dimensiones.

Comparación entre Método 1 y Método 2

La comparativa entre el método 1 y el método 2 dependerá de la complejidad de ambos algoritmos. Aquí está la comparación de la eficiencia asintótica de ambos algoritmos:

- Método 1: O(n^2 * K), donde n es el número de puntos y K es el número de dimensiones.

- Método 2: O(n^2 * K), donde n es el número de puntos y K es el número de dimensiones.

Comparativa:
Ambos métodos tienen una complejidad de tiempo de O(n^2 * d) en el peor caso. Sin embargo, en la práctica, la eficiencia de los dos métodos puede variar según la distribución de los puntos y la naturaleza de los datos de entrada.

La principal diferencia entre los dos métodos es la forma en que combinan las soluciones de los subconjuntos. El Método 1 de Divide y Vencerás utiliza la función fusionar, que compara cada punto de un subconjunto con todos los puntos del otro subconjunto. Por otro lado, la implementación Skyline con Divide y Vencerás utiliza la función fusiona_no_dominados, que compara cada punto de un subconjunto con los puntos no dominados del otro subconjunto.

- Caso en que el método 1 podría ser mejor:
    - Este método puede funcionar mejor cuando hay una mayor cantidad de puntos no dominados en el conjunto de puntos, ya que el proceso de fusión compara cada punto de un subconjunto con todos los puntos del otro subconjunto. Esto significa que, en tales casos, el método 1 puede ser más eficiente al comparar menos puntos entre sí.

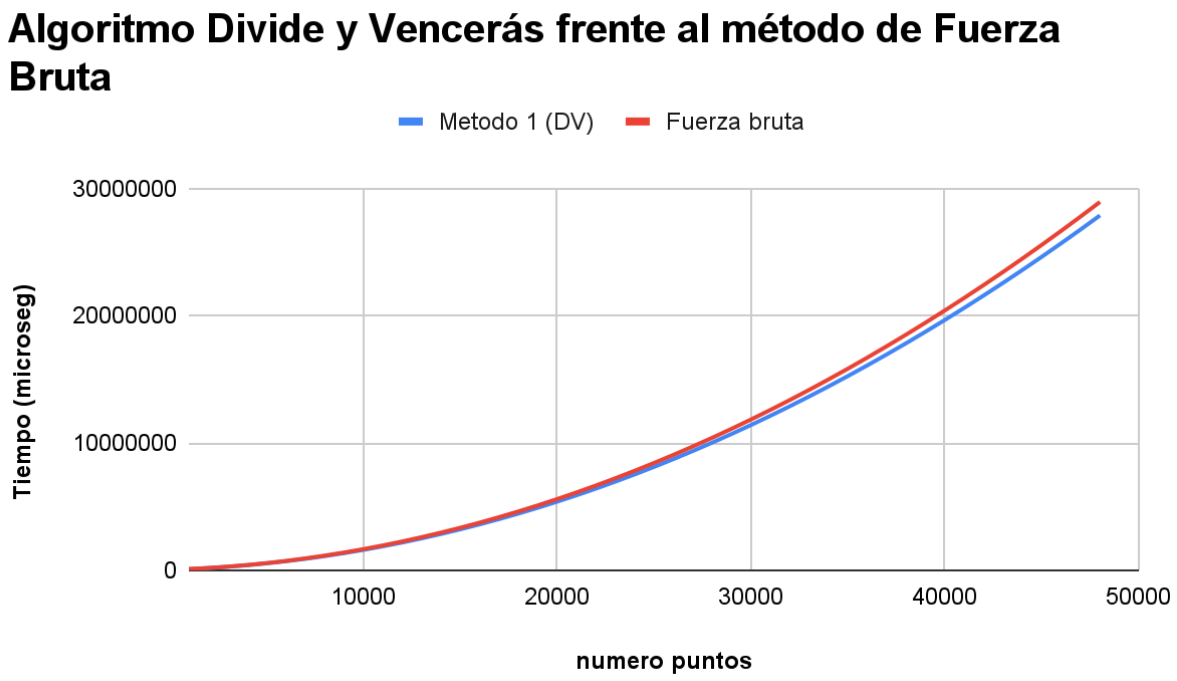
- Caso en que el método 2 podría ser mejor:
    - Este método puede funcionar mejor cuando hay una menor cantidad de puntos no dominados en el conjunto de puntos, ya que la función fusiona_no_dominados compara cada punto de un subconjunto solo con los puntos no dominados del otro subconjunto. Esto significa que, en tales casos, la implementación puede ser más eficiente al eliminar

rápidamente los puntos dominados y realizar menos comparaciones en general.

Dicho esto, es importante recordar que estos son solo casos generales y puede haber variaciones dependiendo de la distribución de los puntos y la naturaleza de los datos de entrada. La mejor manera de determinar cuál método es más eficiente para un conjunto de datos específico es mediante la realización de pruebas experimentales y el análisis de los resultados en diferentes situaciones.

Comparación Método Básico y Divide y Vencerás

Como se ha explicado anteriormente, a una escala general, no se pueden distinguir unas diferencias significativas entre el Método 1 y 2, es por ello que nosotros escogeremos el método 1, que es el método básico de divide y vencerás.
Una vez planteados los métodos “Básico” y “Divide y Vencerás” a continuación los compararemos de forma gráfica para, finalmente, determinar el umbral de forma experimental, generando puntos aleatorios para mejorar la comparativa.



Podemos observar que el método Divide y Vencerás (método 1) y el método de fuerza bruta tienen una complejidad similar de O(n^2 * d) en el caso general,

además de ser muy similares en la práctica. Sin embargo, hay algunas situaciones en las que uno podría ser preferible al otro:

- Caso en que el método Divide y Vencerás (método 1) podría ser mejor:
    - Si la distribución de los puntos en el espacio multidimensional tiende a tener muchos puntos no dominados en cada mitad, entonces el método Divide y Vencerás (método 1) puede ofrecer una mejor eficiencia en la práctica debido a la división y combinación de subproblemas.

- Caso en que el método de fuerza bruta podría ser mejor:
    - Si la distribución de puntos en el espacio multidimensional es tal que hay muchos puntos dominados por otros puntos, entonces la función de comparación en el método de fuerza bruta podría descartar rápidamente estos puntos dominados, y el algoritmo de fuerza bruta podría ser más eficiente en la práctica.

En general, la elección entre el método Divide y Vencerás (método 1) y el método de fuerza bruta dependerá de la distribución de puntos en el espacio multidimensional y del conocimiento previo sobre el conjunto de datos en cuestión.

Si no se tiene información sobre la distribución de los puntos, es posible que se deba realizar una evaluación empírica para determinar qué método funciona mejor en la práctica para el conjunto de datos específico. En este caso, ejecutar ambos algoritmos en varias instancias del problema y comparar los tiempos de ejecución puede ayudar a determinar qué método es más adecuado.

Compilación y ejecución

La estructura de directorios usada es:

- ***bin***: Archivos binarios.
- ***src***: Códigos fuente.
- ***data***: Archivos de datos acerca de los tiempos de ejecución.

En la carpeta src sólo hay un archivo, que es el que hemos usado para sacar las medidas de tiempo. En sus dos primeras líneas se encuentran los comandos que se han de usar para compilarlo y ejecutarlo. Permite elegir que algoritmos queremos probar y el tamaño de dichas pruebas.