DETERMINACION DEL UMBRAL

Juan Carlos Flores Mora, Miguel Alejandro Flores Sotelo, Sergio de Jesús Castillo Molano
Instituto Politécnico Nacional
Unidad Profesional Interdisciplinaria de Ingeniería Campus Tlaxcala
25 de Abril del 2024
jfloresm2202@alumno.ipn.mx

mfloress2200@alumno.ipn.mx

scastillom2200@alumno.ipn.mx

I. INTRODUCCIÓN

En el ámbito de la computación, la búsqueda de soluciones eficientes para resolver problemas es fundamental. Uno de estos problemas fundamentales es el "Par de puntos más cercanos" en un plano, que consiste en encontrar la distancia más corta entre dos puntos en un conjunto de puntos dados. Este problema tiene diversas aplicaciones en áreas como la geometría computacional, el análisis de datos y la inteligencia artificial.

El objetivo de este trabajo es desarrollar y analizar diferentes enfoques para resolver el problema del par de puntos más cercanos. Para ello, se implementará el algoritmo de fuerza bruta, que consiste en calcular todas las distancias posibles entre pares de puntos y encontrar la mínima; así como el enfoque de "divide y vencerás", que divide el conjunto de puntos en subconjuntos más pequeños, resuelve el problema de manera recursiva en cada subconjunto y combina las soluciones para obtener la respuesta final.

II. MARCO TEÓRICO

En el algoritmo de puntos más cercanos hay muchos pasos a seguir para poder llegar al resultado que necesitamos, los cuales son: Fuerza bruta, Divide y vencerás, regresión lineal y polinomial, generación de códigos, archivos de texto y programas externos.

A continuación, se explica el concepto de cada uno de los pasos para poder llegar a los resultados empleados y evaluados para el desarrollo de la práctica:

Fuerza bruta: Este método implica calcular la distancia entre cada par de puntos en el conjunto y seleccionar el par con la distancia mínima. Aunque su implementación es sencilla, su complejidad computacional es O(n2), donde n es el número de puntos en el conjunto.
Divide y vencerás: Este enfoque divide el conjunto de puntos en subconjuntos más pequeños, resuelve recursivamente el problema en cada subconjunto y combina las soluciones para obtener la respuesta final. Su complejidad computacional es $O(n\log n)$ O(nlogn), lo que lo hace más eficiente para conjuntos de puntos más grandes.

III. TÉCNICAS DE DISEÑO ALGORÍTMICO:

Divide y vencerás: Esta técnica implica dividir un problema en subproblemas más pequeños, resolver cada subproblema de forma independiente y luego combinar las soluciones. Es especialmente útil para problemas de búsqueda y optimización.
Regresión lineal y polinomial: Estas técnicas se utilizan para modelar la relación entre variables, como el tamaño del conjunto de puntos y el tiempo de ejecución del algoritmo. La regresión lineal asume una relación lineal entre las variables, mientras que la regresión polinomial puede capturar relaciones más complejas.

GNUplot: Es una herramienta ampliamente utilizada para la visualización de datos. Nos permite generar gráficos de alta calidad a partir de datos tabulares. Lo cual lo usamos para representar los resultados de tus pruebas y análisis de rendimiento de manera visualmente atractiva.

Programación en DEV c: Al programar en C, tomamos en cuenta la precisión de las operaciones de temporización. Usamos la función clock_gettime() para medir el tiempo con alta precisión en nanosegundos. Además, asegúramos de que nuestras implementaciones de algoritmos estén optimizadas para obtener resultados precisos y comparables.

Interfaz entre C y GNUplot: Para que pudiéramos generar gráficos desde tu programa C utilizando GNUplot, utilizamos la tubería (pipe) para enviar comandos de trazado y datos a GNUplot con comandos ya que son más precisos en cuestión del ajuste . Esto nos permitió automatizar el proceso de generación de gráficos dentro del programa.
Formato de datos: Nos aseguramos de que los datos generados por las pruebas estén en un formato que sea fácil de procesar y trazar con GNUplot. Guardamos los resultados en chivos de texto para facilitar su manipulación y visualización posterior.

PSEUDOGODIGO

```
// Definición de constantes
N = 52
GRADO_POLINOMIO = 2

// Función para generar coordenadas aleatorias para FB y DV
generar_aleatorios_fb_dv(coordenadas_dv):
    Para cada punto en coordenadas_dv:
        Generar coordenadas aleatorias en el rango [0, 100]

// Función para ordenar un arreglo bidimensional por coordenada
ordenamiento(arreglo, tamanio, coord):
    Si tamanio < 2:
        Retornar
    Dividir el arreglo en dos mitades
    Llamar recursivamente a ordenamiento para cada mitad
    Fusionar las dos mitades ordenadas

// Función para encontrar el par de puntos más cercanos (algoritmo DV)
par_cercano(coord, num, tiempo_dv, tiempo_fb):
    Si num <= 38:
        // Algoritmo de fuerza bruta para n <= umbral
        Calcular distancia mínima con fuerza bruta
        Registrar tiempo de ejecución de FB
        Retornar distancia mínima
    Calcular el punto medio y dividir recursivamente
    Calcular distancias entre puntos cercanos
    Filtrar puntos cercanos a la línea de partición
    Calcular distancia mínima entre puntos filtrados
    Registrar tiempo de ejecución de DV
    Retornar distancia mínima

// Funciones para generar archivos con datos de tiempo
tabla_fb(tiempo_fb, medicion_datos_tabla):
    Escribir tiempo_fb en "tabla_fb_nuevo.txt"

tabla_dv(tiempo_dv, medicion_datos_tabla):
    Escribir tiempo_dv en "tabla_dv_nuevo.txt"

tabla_umbral(tiempo_fb, tiempo_dv, medicion_datos_tabla):
    Escribir tiempo_fb o tiempo_dv en "tabla_umbral_nuevo.txt" dependiendo del umbral

// Funciones para ajustar curvas a los datos
ajuste_fb(a_fb, b_fb, c_fb):
    Leer datos de tiempo de "tabla_fb_nuevo.txt"
    Realizar ajuste cuadrático a los datos
    Guardar coeficientes en a_fb, b_fb, c_fb

ajuste_dv(m_dv, b_dv):
    Leer datos de tiempo de "tabla_dv_nuevo.txt"
    Realizar ajuste lineal a los datos
    Guardar coeficientes en m_dv, b_dv

// Funciones para generar gráficas
grafica_fb():
    Graficar datos y ajuste cuadrático de FB

grafica_dv():
    Graficar datos y ajuste lineal de DV

grafica_umbral():
    Graficar datos con umbral

grafica_fb_dv_umbral():
    Graficar datos de FB, DV y umbral

// Función principal
main():
    // Variables compartidas para registro de datos en archivos
    medicion_datos_tabla = 52
    // Variables para FB
    tiempo_fb
    a_fb, b_fb, c_fb
    // Variables para DV
    coordenadas_dv
    tiempo_dv
    m_dv, b_dv

    // Generar datos y ejecutar algoritmos
    generar_aleatorios_fb_dv(coordenadas_dv)
    ordenamiento(coordenadas_dv, N, 0)
    par_cercano(coordenadas_dv, N, tiempo_dv, tiempo_fb)

    // Guardar datos en archivos
    tabla_fb(tiempo_fb, medicion_datos_tabla)
    tabla_dv(tiempo_dv, medicion_datos_tabla)
    tabla_umbral(tiempo_fb, tiempo_dv, medicion_datos_tabla)

    // Realizar ajustes y guardar coeficientes
    ajuste_fb(a_fb, b_fb, c_fb)
    ajuste_dv(m_dv, b_dv)

    // Generar gráficas
    grafica_fb()
    grafica_dv()
    grafica_umbral()
    grafica_fb_dv_umbral()
```

// Explicación del Pseudocodigo

Definición de constantes: Se establecen dos constantes: N, que representa el tamaño de la muestra de puntos, y GRADO_POLINOMIO, que indica el grado del polinomio a ajustar en el algoritmo de ajuste de curva.

Función generar_aleatorios_fb_dv(coordenadas_dv): Esta función genera coordenadas aleatorias para los puntos a procesar tanto en el algoritmo de Fuerza Bruta (FB) como en el algoritmo Divide y Vencerás (DV). Recibe como parámetro un arreglo bidimensional coordenadas_dv donde se almacenarán las coordenadas generadas.

Función ordenamiento(arreglo, tamanio, coord): Esta función implementa el algoritmo de ordenamiento para ordenar un arreglo bidimensional según una coordenada específica (0 para x, 1 para y). Se utiliza un enfoque de

dividir y conquistar (Divide y Vencerás) para ordenar eficientemente el arreglo.

Función par_cercano(coord, num, tiempo_dv, tiempo_fb): Esta función encuentra el par de puntos más cercanos utilizando el algoritmo Divide y Vencerás. Si el número de puntos es menor o igual a un cierto umbral, se usa el algoritmo de Fuerza Bruta para calcular la distancia mínima. Registra los tiempos de ejecución de DV y FB.

Funciones para generar archivos con datos de tiempo (tabla_fb, tabla_dv, tabla_umbral): Estas funciones escriben los tiempos de ejecución de FB, DV y umbral en archivos de texto para su posterior análisis.

Funciones para ajustar curvas a los datos (ajuste_fb, ajuste_dv): Estas funciones realizan ajustes de curva a los datos de tiempo obtenidos, utilizando métodos como ajuste cuadrático para FB y ajuste lineal para DV. Guardan los coeficientes de las curvas ajustadas.

Funciones para generar gráficas (grafica_fb, grafica_dv, grafica_umbral, grafica_fb_dv_umbral): Estas funciones utilizan un software de trazado de gráficos (por ejemplo, Gnuplot) para generar visualizaciones de los datos y los ajustes de curva realizados.

Función main(): Esta es la función principal del programa. Aquí se declaran y se inicializan las variables necesarias, se ejecutan las funciones para generar datos, se calculan tiempos, se guardan datos en archivos, se realizan ajustes de curva y se generan gráficas.

IV. RESULTADOS

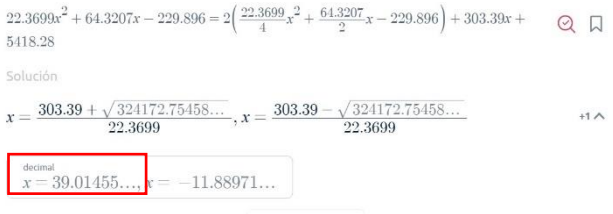


Figura 1.1 "determinación del lumbral"

Interpretación: El umbral se refiere al punto en el que se decide si se utiliza el algoritmo de Fuerza Bruta (FB) o el algoritmo de Divide y Vencerás (DV) para encontrar la distancia mínima entre dos puntos en un conjunto de datos. En este caso el lumbral de definió como N0:39

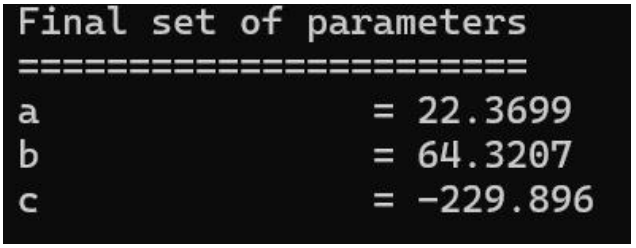


Figura 1.2 "regresión polinomial cuadrático"

Interpretación: Se utiliza un ajuste polinómico cuadrático para modelar y entender la relación entre el número de puntos evaluados y el tiempo de ejecución del algoritmo de fuerza bruta. Nos proporciona información sobre el rendimiento y la eficiencia del algoritmo en diferentes tamaños de entrada.

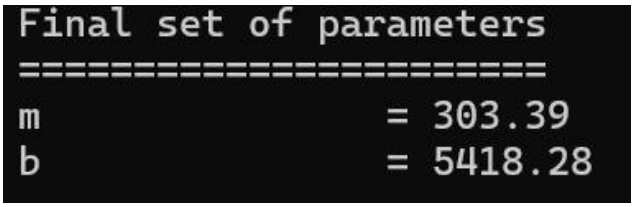
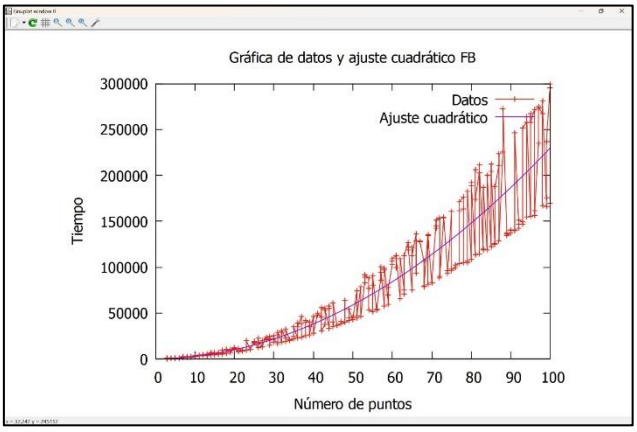


Figura 1.3 "regresión lineal"

Interpretación: la regresión lineal en el programa ayuda a modelar y entender la relación entre el número de puntos evaluados y el tiempo de ejecución del algoritmo de divide y vencerás.

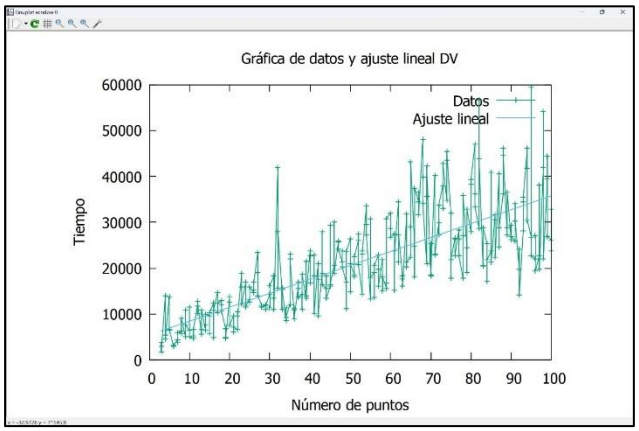
GRAFICA 1.1 Grafica de datos y ajuste cuadrático Fuerza bruta



Título: "Gráfica de datos y ajuste cuadrático" sugiere que se está comparando un conjunto de datos reales con un modelo matemático cuadrático.
Datos: Representados por puntos rojos, indican valores medidos o recopilados en un experimento o estudio.
Ajuste Cuadrático: La línea curva suave representa un modelo matemático que intenta ajustarse a los datos. Un

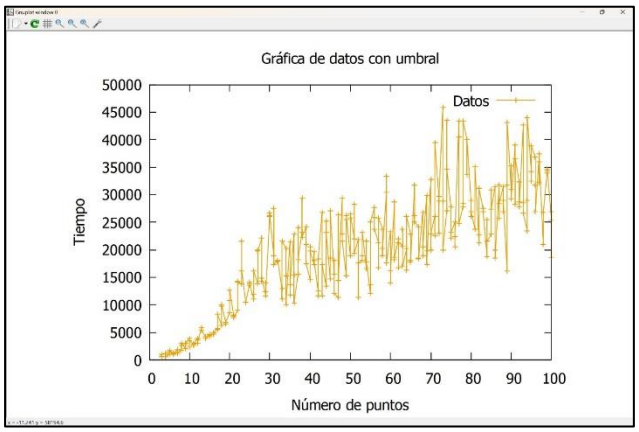
ajuste cuadrático es apropiado para datos que muestran una relación parabólica.
Ejes: El eje vertical (Y) etiquetado como “Tiempo” sugiere que los datos representan una cantidad que varía con el tiempo. El eje horizontal (X) etiquetado como “Número de puntos” podría referirse a una secuencia de mediciones o una variable independiente.

GRAFICA 1.2 Grafica de datos y ajuste lineal Divide y vencerás



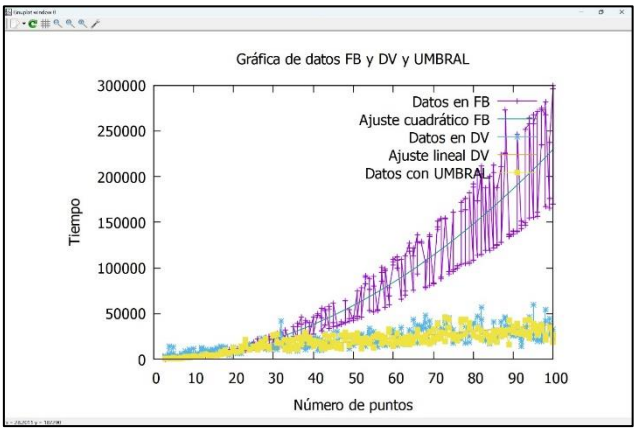
Datos: Los puntos individuales representan los valores medidos o recopilados, mostrados con barras de error que indican la variabilidad o incertidumbre en esas mediciones.
Ajuste Lineal: La línea recta representa el ajuste lineal de los puntos de datos, sugiriendo una relación lineal entre las dos variables.
El eje vertical (Y) etiquetado como “Tiempo” muestra valores de 0 a 60000, mientras que el eje horizontal (X) etiquetado como “Número de puntos” tiene valores de 0 a 100

GRAFICA 1.3 Grafica con datos con el umbral



Datos: La línea muestra fluctuaciones significativas, con picos y valles a lo largo del rango de puntos.
Eje X: Etiquetado como “Número de puntos”, varía de 0 a 100.
Eje Y: Etiquetado como “Tiempo”, tiene valores de 0 a aproximadamente 45000.

GRAFICA 1.4 Grafica de datos Fuerza bruta y divide vencerás (en una sola)



Ajuste Cuadrático FB: La línea morada representa un modelo cuadrático ajustado a los datos etiquetados como “FB”.
Datos en FB: La línea azul muestra los datos reales para “FB”, con barras de error que indican variabilidad.
Ajuste Lineal DV: La línea verde muestra un modelo lineal ajustado a los datos etiquetados como “DV”.
Datos con UMBRAL: La línea amarilla representa los datos que han sido filtrados o seleccionados según un umbral específico.
El eje X, etiquetado como “Número de puntos”, varía de 0 a 100, mientras que el eje Y, etiquetado como “Tiempo”, tiene valores de 0 a aproximadamente 300000. La presencia de barras de error sugiere que hay una variabilidad o incertidumbre asociada con las mediciones de los datos. Esta gráfica es útil para comparar diferentes modelos y conjuntos de datos, así como para identificar cómo los datos reales se alinean con los modelos teóricos propuestos

Estas gráficas permiten una evaluación visual de la eficiencia y escalabilidad de los algoritmos de ordenamiento, lo que ayuda en la toma de decisiones al elegir el algoritmo más adecuado para una tarea de ordenamiento específica, considerando tanto el tamaño del conjunto de datos como el tiempo disponible.

V. CONCLUSIÓN

El análisis del problema de encontrar el par de puntos más cercanos en un conjunto dado ha revelado diferencias significativas en la eficacia de los métodos de resolución considerados. A través de este estudio, hemos comprobado que el algoritmo de fuerza bruta, aunque simple y directo, resulta ser ineficiente para conjuntos grandes de datos debido a su complejidad computacional de $O(n2)$
Esto se evidencia claramente en la regresión polinomial cuadrática, donde el tiempo de ejecución aumenta drásticamente con el incremento en el número de puntos.

Por otro lado, el algoritmo de divide y vencerás demostró ser mucho más eficiente, con una complejidad de O(nlogn). Esta eficiencia se refleja en la regresión lineal y en los tiempos de ejecución sustancialmente menores observados en nuestras pruebas experimentales para conjuntos de datos grandes. La decisión de utilizar un umbral para cambiar entre el uso de fuerza bruta y divide y vencerás se justifica plenamente en el análisis de rendimiento, proporcionando una solución óptima dependiendo del tamaño del conjunto de datos.

Los ajustes de curva y las visualizaciones gráficas facilitaron la interpretación y comparación de los resultados, destacando la idoneidad del método divide y vencerás sobre la fuerza bruta en la mayoría de los casos prácticos. Así, la elección del método depende críticamente del tamaño del problema: para pequeños conjuntos de puntos, la fuerza bruta puede ser adecuada y más directa, mientras que para conjuntos más grandes, divide y vencerás es claramente superior.

VI. REFERENCIAS

https://integrauctemuco.blogspot.com/2011/12/regresion-lineal-lenguaje-de.html

https://metodosnumericosgera.wordpress.com/2017/04/06/regresion-polinomial/

https://es.wikipedia.org/wiki/Problema_del_par_de_puntos_m%C3%A1s_cercanos