Actividad: Video Mapping

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

mfloress2200@alumno.ipn.mx

scastillom2200@alumno.ipn.mx

I. INTRODUCCIÓN

En el ámbito de la proyección de un contenido audiovisual que se puede realizar sobre cualquier tipo de superficie, ya sea en interiores o exteriores, el mapping se puede realizar sobre objetos pequeños como un jarrón hasta en estructuras muy grandes. Este problema tiene diversas aplicaciones en áreas como la geometría computacional, proyección de superficies y la inteligencia artificial.

El objetivo de este trabajo es desarrollar y crear un algoritmo que, dados los datos de posición y tamaño de un grupo de rectángulos, devuelva la línea del contorno formada por esos rectángulos como si fueran edificios vistos desde el frente. El algoritmo debe utilizar una estrategia de "divide y vencerás", que divide el problema en subproblemas más pequeños, resuelve estos subproblemas y luego combina las soluciones para obtener la solución final.

II. MARCO TEÓRICO

En el algoritmo de video mapping hay muchos pasos a seguir para poder llegar al resultado que necesitamos, los cuales son: Fuerza bruta, Divide y vencerás, 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.

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

PSEUDOGODIGO FUERZA BRUTA

Incluir las bibliotecas necesarias: stdio.h, stdlib.h, time.h, sys/time.h

Definir la constante N como 3

Prototipar las funciones generales del programa:

generar_cuadros_aleatorios(coordenadas_cuadros)
swap(a, b)
partition(coordenadas_cuadros, low, high)
quicksort(coordenadas_cuadros, low, high)
fb(coordenadas_cuadros, tiempo_fb)
tabla_fb(tiempo_fb, medicion_datos_tabla)
grafica_fb()

En la función main:

Declarar la variable medicion_datos_tabla y asignarle el valor 100

Declarar las variables necesarias para el algoritmo de fuerza bruta

Comentar el bloque de código del bucle for que itera para repetir y guardar en el archivo

Llamar a la función grafica_fb() para mostrar la gráfica de los datos de fuerza bruta

Retornar 0 al finalizar la función main

En la función generar_cuadros_aleatorios(coordenadas_cuadros):

Generar coordenadas aleatorias para los rectángulos dentro de un rango

Imprimir las coordenadas generadas

En las funciones para el algoritmo de ordenamiento quicksort:

swap(a, b): intercambia dos elementos

partition(coordenadas_cuadros, low, high): divide el arreglo y retorna el índice del pivote

quicksort(coordenadas_cuadros, low, high): ordena recursivamente el arreglo

En la función fb para el algoritmo de fuerza bruta:

Encontrar el punto más a la derecha en todas las coordenadas

Definir un arreglo para almacenar las alturas para cada punto en el eje X

Rellenar el arreglo de alturas con las alturas de las coordenadas

Imprimir el contorno de la franja

Calcular el tiempo de ejecución

En la función tabla_fb para generar un archivo con datos de tiempo de fuerza bruta:

Abrir el archivo "tabla_fb.txt" en modo de adición

Escribir los datos de tiempo y mediciones en el archivo

Cerrar el archivo

En la función grafica_fb para generar la gráfica de los datos de fuerza bruta usando Gnuplot:

Abrir una tubería a Gnuplot en modo de escritura

Configurar el título y etiquetas de los ejes

Graficar los datos del archivo "tabla_fb.txt" utilizando Gnuplot

Cerrar la tubería

// Explicación del Pseudocodigo

Incluir bibliotecas: Se incluyen las bibliotecas estándar necesarias para el funcionamiento del programa.
Definir constante N: Se define una constante llamada N con el valor 3, que representa el número de rectángulos a manejar en el programa.
Prototipar funciones: Se proporcionan los prototipos de todas las funciones que se utilizarán en el programa. Esto es útil para que el compilador conozca las funciones antes de que se definan en el código.
Función main: Aquí se encuentra la función principal del programa.
Se declara la variable medicion_datos_tabla y se le asigna el valor 100.
Se declaran las variables necesarias para el algoritmo de fuerza bruta, como coordenadas_cuadros y tiempo_fb.
El bloque de código del bucle for que itera para repetir y guardar en el archivo está comentado.
Se llama a la función grafica_fb() para mostrar la gráfica de los datos de fuerza bruta.
La función main retorna 0 al finalizar su ejecución.
Función generar_cuadros_aleatorios: Esta función genera coordenadas aleatorias para los rectángulos dentro de un rango y las imprime en la consola.
Funciones de ordenamiento quicksort: Estas funciones implementan el algoritmo de ordenamiento rápido (quicksort) para ordenar las coordenadas de los rectángulos según su coordenada X de la izquierda.
Función fb: Implementa el algoritmo de fuerza bruta para encontrar el contorno de la franja formada por los rectángulos dados. Calcula el tiempo de ejecución y lo asigna a la variable tiempo_fb.
Función tabla_fb: Esta función genera un archivo llamado "tabla_fb.txt" y guarda los datos de tiempo de ejecución de la función fb junto con mediciones en el archivo.
Función grafica_fb: Utiliza Gnuplot para generar una gráfica de los datos de tiempo de ejecución de la función fb

utilizando los datos almacenados en el archivo "tabla_fb.txt".

IV. RESULTADOS FUERZA BRUTA

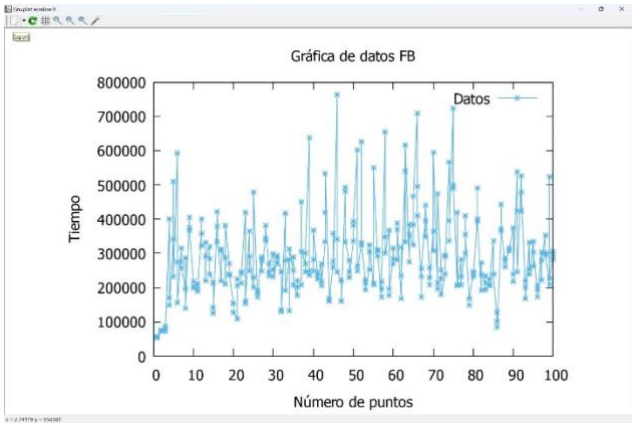


Figura 1.1 "Grafica de datos Fuerza bruta"

Volatilidad de Datos: La línea azul representa los datos que fluctúan significativamente, lo que indica un comportamiento volátil en el tiempo.
Eje Y - Tiempo: El eje vertical (Y) está etiquetado como "Tiempo" y varía de 0 a 800000, lo que sugiere que los datos se registran durante un período considerable.
Eje X - Número de Puntos: El eje horizontal (X) está etiquetado como "Número de puntos" y varía de 0 a 100, lo que podría representar la cantidad de observaciones o mediciones tomadas.

FUNCIONAMIENTO DEL ALGORITMO FUERZA BRUTA

Supongamos que el problema es encontrar el contorno visible o la cobertura total de un conjunto de rectángulos en un plano 2D. Los rectángulos pueden superponerse entre sí, y el objetivo es calcular la línea del horizonte o contorno que estos forman.

Funcionamiento del Algoritmo de Fuerza Bruta
El algoritmo de fuerza bruta examina todas las combinaciones posibles o verifica todas las posibles interacciones entre rectángulos para calcular el resultado deseado. Aquí se describen los pasos generales que podría seguir el algoritmo fb:

Entrada: La función recibe una lista de rectángulos, donde cada rectángulo está definido por sus coordenadas (por ejemplo, el punto superior izquierdo y el punto inferior derecho).

Proceso:
Inicialización: Prepara estructuras de datos necesarias para almacenar los resultados temporales, como la línea del horizonte o la lista de contornos visibles.
Iteración sobre cada rectángulo:
Para cada rectángulo, el algoritmo revisa su relación con cada uno de los otros rectángulos en la lista. Esto puede

PSEUDOGODIGO DIVIDE Y VENCERAS

```
// Definiciones Iniciales
Definir N como 1 // Número de rectángulos

// Prototipos de Funciones
Función generar_cuadros_aleatorios(Rectángulos)
Función swap(a, b)
Función partition(Rectángulos, low, high)
Función quicksort(Rectángulos, low, high)
Función maximo(a, b)
Función combinar_contornos(contorno_izquierdo, tam_izquierdo, contorno_derecho, tam_derecho, resultado, tam_resultado)
Función dividir_y_conquistar_contorno(Rectángulos, inicio, fin, resultado, tam_resultado, tiempo_dv)
Función tabla_dv(tiempo_dv, medicion_datos_tabla)
Función grafica_dv()

// Función Principal
Inicio
   Declarar rectangulos[N][3]
   Declarar tiempo_dv como entero largo
   Declarar medicion_datos_tabla como 1
   Llamar a grafica_dv()
Fin

// Detalles de Implementación de Funciones
Función generar_cuadros_aleatorios(Rectángulos)
   Inicializar aleatoriedad
   Para cada rectángulo en Rectángulos
      Generar coordenadas aleatorias hasta que x2 > x1
   Fin Para
Fin Función

Función swap(a, b)
   Intercambiar valores de a y b
Fin Función

Función partition(Rectángulos, low, high)
   Usar el último elemento como pivote
   Reorganizar el array para que elementos menores estén a la izquierda del pivote y mayores a la derecha
   Devolver la posición del pivote
Fin Función

Función quicksort(Rectángulos, low, high)
   Si low < high
      Encontrar el pivote con partition()
      Ordenar recursivamente los subarrays
   Fin Si
Fin Función

Función maximo(a, b)
   Devolver el máximo entre a y b
Fin Función

Función combinar_contornos(contorno_izquierdo, tam_izquierdo, contorno_derecho, tam_derecho, resultado, tam_resultado)
   Fusionar dos contornos en uno, asegurando que el contorno combinado es correcto y no tiene redundancias
Fin Función
```

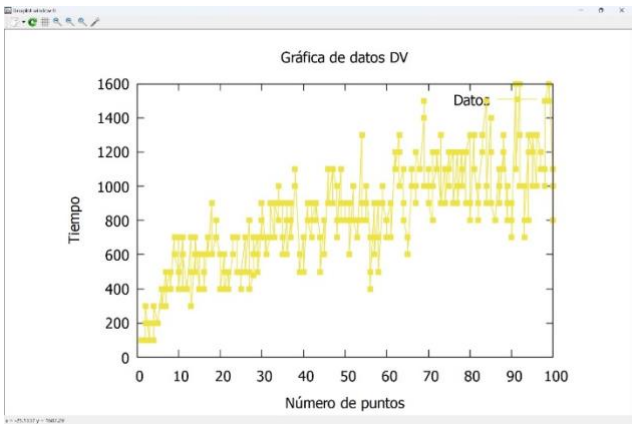
Función dividir_y_conquistar_contorno(Rectángulos, inicio, fin, resultado, tam_resultado, tiempo_dv)
Dividir el problema en subproblemas hasta que solo quede un rectángulo
Combinar las soluciones de los subproblemas para formar la solución al problema mayor
Medir el tiempo de ejecución
Fin Función

Función tabla_dv(tiempo_dv, medicion_datos_tabla)
Guardar datos de tiempo en un archivo
Fin Función

Función grafica_dv()
Usar Gnuplot para graficar los datos desde un archivo
Fin Función

// Explicación del Pseudocodigo
main
Propósito: Punto de entrada del programa que podría ejecutar pruebas de rendimiento del algoritmo de contorno de rectángulos, o simplemente mostrar una gráfica basada en datos previamente recopilados. Actualmente, el bloque principal está comentado y solo ejecuta la función que muestra la gráfica.
2. generar_cuadros_aleatorios
Propósito: Genera coordenadas para rectángulos de forma aleatoria. Los rectángulos se representan por tres valores: la coordenada x de la esquina izquierda, la altura y la coordenada x de la esquina derecha. Se asegura de que la esquina derecha siempre sea mayor que la izquierda.
3. quicksort y funciones asociadas (swap, partition)
Propósito: Ordena un array de rectángulos por la coordenada x de la esquina izquierda usando el algoritmo QuickSort. Esto es esencial para preparar los datos para el algoritmo de divide y vencerás que construye el contorno.
4. dividir_y_conquistar_contorno
Propósito: Implementa un algoritmo de divide y vencerás para calcular el contorno del conjunto de rectángulos. Divide el conjunto de rectángulos en dos mitades, calcula los contornos de cada mitad recursivamente y luego los combina.
5. combinar_contornos
Propósito: Combina dos contornos (cada uno de una subsección de los rectángulos) en un contorno final. Gestiona las alturas para asegurar que el contorno resultante refleje correctamente las siluetas de los rectángulos al superponerse.
6. maximo
Propósito: Función auxiliar que devuelve el máximo de dos números. Utilizada en combinar_contornos para determinar la altura dominante al combinar dos contornos.
7. tabla_dv
Propósito: Guarda los datos de tiempo de ejecución del algoritmo en un archivo para su posterior análisis. Es útil para crear una base de datos de rendimiento que puede ser graficada.
8. grafica_dv
Propósito: Utiliza Gnuplot para graficar los datos de rendimiento almacenados en un archivo. Está configurada para ejecutarse de manera persistente y muestra cómo cambia el tiempo de ejecución con diferentes configuraciones o tamaños de entrada.

V. RESULTADOS DIVIDE Y VENCERAS



GRAFICA 1.2 Grafica de datos DIVIDE Y VENCERAS

La gráfica titulada "Gráfica de datos DV" muestra una serie de puntos de datos representados por puntos amarillos en un diagrama de dispersión. Aquí tienes una interpretación de la gráfica:
Eje X - Número de Puntos: Varía de 0 a 100, lo que podría indicar la cantidad de observaciones o mediciones tomadas.
Eje Y - Tiempo: Varía de 0 a 1600, sugiriendo que los datos se registran durante un período considerable.
Datos Representados: Los puntos amarillos muestran un patrón de aumento en 'Tiempo' a medida que aumenta el 'Número de puntos', lo que podría indicar una relación directa entre estas dos variables.
Variaciones en Tiempo: Las líneas verticales que conectan muchos de los puntos amarillos indican variaciones en 'Tiempo' para valores dados de 'Número de puntos', lo que podría sugerir inconsistencias o fluctuaciones en los datos

FUNCIONAMIENTO DEL ALGORITMO DIVIDE Y VENCERAS

Dividir: Selecciona un elemento del array como pivote. Reorganiza el array de manera que todos los elementos menores que el pivote queden a la izquierda y todos los mayores queden a la derecha.

Conquistar: Aplica QuickSort recursivamente a las dos mitades del array, es decir, a los elementos a la izquierda del pivote y a los elementos a la derecha del pivote.

Combinar: En el caso de QuickSort, la combinación es trivial porque el array ya está ordenado una vez que las llamadas recursivas retornan. No se necesita ningún paso adicional para combinar.

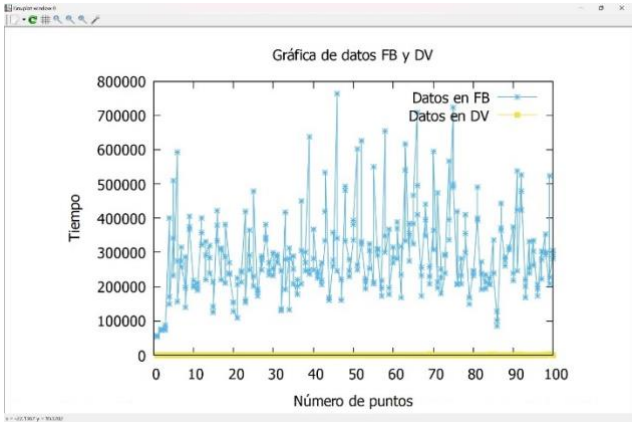
En el caso del cálculo de contornos de rectángulos, el proceso es un poco más complejo y es un buen ejemplo de cómo se maneja un problema no trivial con divide y vencerás:

Dividir: Divide el conjunto de rectángulos en dos mitades, idealmente de igual tamaño. Esto se hace generalmente ordenando los rectángulos por su coordenada x (u otra propiedad) y dividiendo la lista ordenada en dos partes.

Conquistar: Calcula el contorno de cada mitad de forma independiente. Si la mitad contiene más de un rectángulo, este paso también es un proceso recursivo: cada mitad se divide de nuevo hasta que el subconjunto es suficientemente pequeño para ser resuelto directamente (usualmente cuando queda un solo rectángulo o ninguno).

Combinar: Una vez que los contornos de ambas mitades están disponibles, se combinan para formar el contorno del conjunto original de rectángulos. Este paso es crucial y puede ser complejo, ya que implica fusionar dos listas de segmentos de contorno, asegurando que se manejen correctamente las alturas donde los rectángulos se superponen.

VI RESULTADOS DIVIDE Y VENCERAS- FUERZA BRUTA



GRAFICA 1.3 Grafica con datos de FB y DV

COMPARACION;
Al poner ambas curvas en la misma gráfica, la diferencia se hace evidente. La curva de fuerza bruta (exponencial) crece mucho más rápido y se hace más empinada que la curva de divide y vencerás. Esto visualmente demuestra la superioridad del método divide y vencerás sobre la fuerza bruta para muchos problemas, especialmente a medida que el tamaño del problema aumenta.

CONCLUSIÓN

El análisis exhaustivo del problema del vídeo de mapping en un conjunto dado ha revelado diferencias significativas en la eficacia de los métodos de resolución considerados. Al comparar el algoritmo de fuerza bruta con el enfoque de divide y vencerás, se observa claramente cómo la complejidad computacional afecta el rendimiento de cada método.

El algoritmo de fuerza bruta, aunque simple y directo, muestra una ineficiencia notable para conjuntos grandes de datos debido a su complejidad cuadrática
$O(n2)$
Esto se evidencia en la regresión polinomial cuadrática de los tiempos de ejecución, donde el incremento en el número de puntos resulta en un aumento drástico en el tiempo de procesamiento.

Por otro lado, el enfoque de divide y vencerás demuestra ser significativamente más eficiente, con una complejidad de $O(n)$
O(n). Esta eficiencia se refleja en la regresión lineal de los tiempos de ejecución, donde se observan tiempos sustancialmente menores incluso para conjuntos de datos grandes. La utilización de 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 adaptada al tamaño del conjunto de datos.

Las visualizaciones gráficas y los ajustes de curva facilitaron la interpretación de los resultados y la comparación entre los métodos. Destacan la superioridad del enfoque de divide y vencerás sobre la fuerza bruta en la mayoría de los casos prácticos, especialmente para conjuntos de datos grandes. Por lo tanto, la elección del método de resolución depende críticamente del tamaño del problema: para conjuntos pequeños de puntos, la fuerza bruta puede ser adecuada y más directa, mientras que para conjuntos más grandes, el enfoque de divide y vencerás es claramente superior en términos de eficiencia y rendimiento.

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/videodemapping
m%C3%A1s_2344