**Algorítmica y Modelos de Computación**

*Práctica 1*

Un dibujo de una persona

Descripción generada automáticamente con confianza baja

Cristian Delgado Cruz Universidad 49731061R de Huelva

**Índice**

1.- Análisis Teórico 1.1.-Algoritmo Exhaustivo 1.2.-Algoritmo Divide y Vencerás 1.2.1.-Resumen de diferencias. 1.3.-Algoritmo de Dijkstra

2.- Representación de los Resultados

3.- Explicación del funcionamiento general

4.- Explicación de la Interfaz Gráfica

Análisis Teórico

En este Análisis Teórico utilizaré mi código implementado para hacer el cálculo de los costes, para ello utilizaré los cálculos vistos en clase.

Además, compararé el caso exhaustivo con el divide y vencerás, primero teóricamente y luego empíricamente en un caso de 14 puntos y de 1000 puntos aleatorios gracias a una función que yo mismo implementé.

Algoritmo Exhaustivo

Mi código exhaustivo consiste en mirar todas las combinaciones de puntos desde el primero hasta el último y quedarme con aquella combinación de 3 que tenga la menor distancia del primero al segundo y del segundo al tercero.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Que al final se convierte en:

Cosa que tiene sentido porque son 3 bucles “for”

Empíricamente en una muestra pequeña de 14 puntos el tiempo en el que carga el resultado es de menos de medio segundo, casi incontable. Pero con los mil puntos, llegamos a 6 segundos.

Este algoritmo no tiene casos, peor mejor y base, porque comprueba todos los puntos siempre, es decir, su caso base, caso mínimo y caso mejor es el mismo, porque siempre recorre todos los puntos para encontrar el mejor camino.

Algoritmo Divide y Vencerás

Para el estudio teórico no voy a contar el coste de ordenar los elementos y el coste de combinarlo, ya que podríamos considerarlo de orden ya que solo se combinan pocos puntos en cada llamada.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Por ello nos queda la siguiente ecuación:

T(n) =

T(n) = 2T(n/2) + n;

Cambio:

n = 🡪 T() = 2T() + ;

Otro Cambio:

T() = 🡪 ;

= T(k) = \* p(k);

;

;

Deshacemos el cambio:

; 🡪 T(n) = C1 \*n\* log n + C2\* n;

Quedándonos así que el orden es de 0

Empíricamente la diferencia entre 14 puntos y 1000 puntos no tiene diferencia apreciable gracias a su orden de complejidad.

Su caso peor será cuando valga n^3, es decir el último trio de puntos que examine sea el valido.

Su caso mejor será encontrarlos instantáneamente es decir con la primera partición encontrar una distancia tan pequeña que no haya trio de puntos que al compararlos den algo menor que ese valor

Diferencias entre Exhaustivo y DyV

Como es de suponer, para los casos más pequeños la diferencia entre el Algoritmo exhaustivo y Divide y vencerás son completamente imperceptibles, ya que el funcionamiento del exhaustivo en casos pequeños en muy bueno, pero cuanto más puntos añadimos peor rendimiento da.

Por otra parte el Divide y Vencerás como va quitándose del medio puntos con cada llamada recursiva termina acabando muchísimo antes, y aunque en casos muy pequeños no se note la diferencia, en casos grandes tiene un rendimiento tan superior que deja al exhaustivo completamente fuera de poder hacer cualquier cosa.

Un ejemplo práctico podría ser el siguiente:

Si queremos calcular la distancia entre tres pueblos de una provincia, calculamos (según la provincia) la combinación de tres en tres de todos ellos, y quizá nos puede salir 20 pueblos en una provincia, eso significa que da igual que método usemos, nos valdrían los dos, por otra parte si queremos hacer lo mismo pero para toda España o aún peor para toda Europa, tendríamos que considerar todas las combinaciones haciendo casi infinito el método exhaustivo.

Por ello el DyV es muchísimo mejor algoritmo

Algoritmo Dijkstra

Imagen que contiene Texto

Descripción generada automáticamente

Imagen que contiene Texto

Descripción generada automáticamente

En este caso, tenemos un algoritmo donde la complejidad viene definida por el bucle for donde asignamos los pesos a la variable Resultado [], para ello habrá que recorrerla entera es decir, complejidad n. También hay que tener en cuenta el bucle donde buscamos el camino mejor, donde recorreremos toda estos nodos para hallar los caminos con los costes antes calculados. Por lo tanto, podemos afirmar que el algoritmo tiene una complejidad de n^2.

Los diferentes casos que podrían hacer el algoritmo dependen del número de intercambios que tenga que realizar para reasignar los caminos

El mejor caso se produciría cuando no se añade ningún punto al camino para ir de un vector a otro y todos los caminos son directos. Aun así, tendríamos que inicializar la variable Resultado [] , buscar la arista con el coste mínimo y hacer todas las comprobaciones por lo que la complejidad en el mejor caso y peor seguirían siendo n^2.

Representación de los resultados

***Puntos***

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Exhaustivo Divide y Vencerás

Un papalote volando en el cielo

Descripción generada automáticamente con confianza baja

Los Azules son los puntos resultados.

***Nodos***

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Imagen que contiene Gráfico

Descripción generada automáticamente

Explicación del Funcionamiento General

***Paquetes***

*Algoritmo:*

Contiene los métodos de los algoritmos propiamente dichos, Divide y Vencerás, Exhaustivo, QuickShort (Que ordena para el Divide y Vencerás) y Voraz.

*Controlador:*

Contiene las clases que se encargan de controlar el funcionamiento de las vistas, todas implementan ActionListener y para ello Sobrescriben el método actionPerformed, además de un addListener para escuchar los eventos.

*Modelo:*

Contiene las clases que se encargan de mantener los objetos que usamos en la práctica, además encontramos aquí Escritura y Lectura que son las clases que se encargan del uso de ficheros y son llamadas por el controlador principal.

*Vista:*

Contiene las clases que se encargan de mostrar por pantalla los resultados, mensajes y demás para informar al usuario de sus resultados.

Explicación de la interfaz gráfica

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

La interfaz gráfica es un JFrame con varios botones que son capturados como eventos en el controlador.

Por otra parte, los resultados son mostrados en JOptionpanel y la representación en Canvas montados en un Panel.

Los Botones 1 a 1 funcionan así:

Representar Puntos: Representan los puntos solo si se han cargado o creado antes, si no, salta un error en un JOptionpanel.

Representar Solución: Lo mismo que el de arriba pero solo si se han registrado unos resultados antes.

Crear Puntos: Crea 1000 puntos aleatorios, la cantidad puede ser cambiada en el código.

Cargar Puntos: Carga puntos de un fichero especifico, este fichero también puede ser cambiado en el código.

Guardar Nodos: Guarda los nodos en un fichero, que también se puede cambiar en el código.

Cargar Nodos: Crea los Nodos desde un código que también se puede retocar a gusto

Continuar: Encuentra la solución con uno de los métodos escogidos en el JComboBox, si escoges Exhaustivo o DyV sin cargar/crear los puntos o Dijkstra sin cargar los nodos dará un error en un JOptionpanel, si eliges Dijkstra también representará la solución.

Salir: Sale del programa.

El Combo Box sirve para elegir el algoritmo.