Análisis de Algoritmos

Proyecto Final: Manual del Usuario

Nombres:

Martínez Partida Jair Fabian

Martínez Rodríguez Alejandro

Monteros Cervantes Miguel Angel

Luis Fernando Ramírez Cotonieto

Fecha de entrega:21 de Junio del 2021

Grupo:3CM13

INSTITUTO POLITÉCNICO NACIONAL

ESCOM

Índice

Proyecto Final

Análisis de Algoritmos

1. Planteamiento del problema

En geometría computacional, el problema del par de puntos más cercano es un problema clásico donde "Dados n puntos en un espacio métrico, se pide encontrar un par de puntos con la distancia más pequeña entre ellos". El problema del par de puntos más cercano en el plano euclidiano fue de los primeros problemas tratados en el estudio sistemático de la complejidad computacional de algoritmos geométricos.
Un algoritmo ingenuo para resolver el problema consiste en çalcular las distancias entre todos los pares de puntos del conjunto y seleccionar el mínimo", que requiere un tiempo O(n^2). Pero resulta que el problema puede ser solucionado en tiempo O(n log n) en un espacio euclídeo. Este tiempo puede incluso ser mejorado: Si asumimos que la función de parte entera (floor) es computable en tiempo constante, el problema puede ser solucionado en tiempo O(n log log n). Si además permitimos utilizar aleatorización, el problema puede ser solucionado en tiempo O(n).

2. Planteamiento del Algoritmo y Solución

2.1. Tecnologías utilizadas

Se utilizó **JavaScript** (abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.
Junto con la librería **Raphael.js** llamado así por el artista Raffaello Sanzio da Urbino, Rapahel.js es una biblioteca javascript diseñada específicamente para artistas y diseñadores gráficos. Es el pincel que puedes utilizar para aplicar imágenes directamente al lienzo del navegador.

2.2. Divide y Conquista

Un enfoque simple es hacer una búsqueda lineal, es decir: Se nos da una matriz de n puntos en el plano, y el problema es averiguar el par de puntos más cercano en la matriz. Este problema surge en una serie de aplicaciones. Por ejemplo, en el control de tráfico aéreo, es posible que desee monitorear los aviones que se acercan demasiado, ya que esto puede indicar una posible colisión. Recuerde la siguiente fórmula para la distancia entre dos puntos p y q.

$$\|pq\| = \sqrt{(px - qx)^2 + (py - qy)^2}$$

La solución de fuerza bruta es O(n^2), calcula la distancia entre cada par y devuelve la más pequeña. Se puede calcular la distancia más pequeña en el tiempo O(nLogn) usando la estrategia Divide y Conquer.
Pasos:

Entrada: Un array de n puntos P[]
Salida: La distancia más pequeña entre dos puntos en el array dado.

Como paso de procesamiento previo, la matriz de entrada se ordena según las coordenadas x.

1. Encontrar el punto medio en el array ordenado, podemos tomar P[n/2] como punto medio.
2. Divida el array dado en dos mitades. El primer subarray contiene puntos de P[0] a P[n/2]. El segundo subarray contiene puntos de P[n/2+1] a P[n-1].
3. Encontrar recursivamente las distancias más pequeñas en ambos subarrays. Dejando que las distancias sean dl y dr. Encontrar el mínimo de dl y dr. Dejando que el mínimo sea d.
4. A partir de los 3 escalones anteriores, tenemos un límite superior d de distancia mínima. Ahora tenemos que considerar los pares de tal manera que un punto en el par sea de la mitad izquierda y el otro sea de la mitad derecha. Considere la línea vertical que pasa por P[n/2] y encuentre todos los puntos cuya coordenada x está más cerca que d de la línea vertical media. Construya una franja de matriz[] de todos esos puntos.
5. Se ordena la franja de matriz[] según las coordenadas y este paso es O(nLogn). Se puede optimizar a O(n) ordenando y fusionando recursivamente.
6. Se encuentra la distancia más pequeña en la tira[]. Esto es complicado. Desde el primer vistazo, parece ser un paso O(n^2), pero en realidad es O(n). Se puede probar geométricamente que para cada punto de la tira, solo necesitamos comprobar como máximo 7 puntos después de ella (tenga en cuenta que la tira está ordenada de acuerdo con la coordenada Y).
7. Finalmente devuelva el mínimo de d y la distancia calculada en el paso anterior. Implementación

3. Manual de uso de la animación

3.1. Usando Raphael para dibujar algunas formas en multiples navegadores

Primeramente enfoquemonos en que es Raphaël. Raphaël es una libreria Javascript para gráficos SVG destinados a todos los navegadores actuales incluyendo a Internet Explorer que como todos sabemos es el que mas problemas da a la hora de programar puesto que no es sencillo encontrar librerias que sean compatibles con este navegador.

Su modo de uso es sencillo primero debemos obtener la libreria que la descargamos de la pagina:
http://raphaeljs.com/ y ademas de esto necesitaremos jQuery (no imprescindible)

Para comenzar debemos colocar en nuestro código HTML las librerias a usarse para esto incluimos el siguiente codigo:

Luego debemos poner una funcion inicializadora en este caso hare uso de el inicializador de jQuery:

```
$(
  function(){
    ...
  }
);
```

Dentro del mismo podremos hacer uso de la libreria Raphaël y el conjunto de funciones de dibujo que deseemos.
Y para esto necesitamos primero crear el canvas de dibujo de Raphaël,

procedemos a hacerlo con el siguiente codigo:

```
$(
        function(){
                var paper = Raphael(10, 50, 700, 700);
        }
);
```

Donde paper es el objeto que contiene el canvas de dibujo para Raphaël, el cual esta posicionado en la posición 10,50 y tiene un tamaño tanto en alto como en ancho de 700px.

Para lograr dibujar cualquier objeto o interactuar con objetos de ahora en adelante usaremos el objeto paper.

Nuestro objetivo es dibujar solo cuatro objetos que nos den alguna forma mas o menos simétrica de un objeto, esto lo conseguimos asi:

```
//crear un circulo
var circle = paper.circle(200, 200, 100);
//dibujar un cuadrado
var square = paper.rect(200, 200, 100,100);
//crear un cuadrado
var squaretwo = paper.rect(100, 100, 100,100);
//poner un texto de hola mundo
var t = paper.text(150,150, "Hola Mundo con Raphaël...");
```

Y por ultimo procedemos a darle color para decorarlos con el siguiente código:

```
// poner un atributo de llenado de color del circulo
circle.attr("fill", "#f00");
// poner un atributo de linea de color #f77
circle.attr("stroke", "#f77");


// poner un atributo de llenado de color del cuadrado
square.attr("fill", "#ccc");
// poner un atributo de linea de color #f77
square.attr("stroke", "#f77");
// poner un atributo de llenado de color del cuadrado
squaretwo.attr("fill", "#ccc");
```

Y obtenemos un interesante resultado que podemos probar en algún navegador de internet.

4. Bibliografía

- POSNER, Eric A.; SPIER, Kathryn E.; VERMEULE, Adrian. Divide and conquer. Journal of Legal Analysis, 2010, vol. 2, no 2, p. 417-471.
- CARO, Miguel Rodrigo Pincheira. Busqueda del par de puntos mas cercanos sobre conjuntos no indexados. 2012.
- CHANDÍA, Nahuel; FERNANDO, Álvaro. Implementación de un sistema web y móvil para obtener los K-Pares de vecinos más cercanos entre dos conjuntos de puntos espaciales representados en la estructura de datos compacta k2-Tree. 2017.
- PASCUAL, D.; PLA, F.; SÁNCHEZ, S. Algoritmos de agrupamiento. Método Informáticos Avanzados, 2007, p. 164-174.