UNIVERSIDAD NACIONAL DE SAN AGUSTÍN

Facultad de Ingeniería de Producción y Servicios

Ciencia de la Computación

QuadTree

ALUMNO:
Pfuturi Huisa, Oscar David

PROFESOR:
MSc. Vicente Machaca Arceda

CURSO:
Estructura de Datos Avanzada

2 de octubre de 2020

Índice

1. Introducción

QuadTree es un árbol usado para almacenar eficientemente datos de puntos en un espacio bidimensional. En este árbol, cada nodo padre puede tener hasta cuatro nodos hijo. Esto se puede usar para separar nuestro escenario en cuatro áreas iguales recursivamente.

2. Ejercicios

2.1. Archivo main.html

Creación de una página web para la demostración del funcionamiento de la estructura Quadtree.

2.1.1. Código Fuente

```
<!DOCTYPE html>
<html>
<head>
    <title>QuadTree</title>
    <meta charset="utf-8">
    <script src="p5.min.js"></script>
    <script src="quadtree.js"></script>
    <script src="sketch.js"></script>
</head>
<body>
</body>
</html>
```

2.2. Archivo quadtree.js

En este archivo se muestra la estructura de QuadTree. La clase Point indicará un punto en el espacio creado por la clase Rectangle. Dentro de la clase Rectangle hay dos funciones:

- **contains(point)**, indicará si un punto pertenece a cierto espacio.
- **intersects(range)**, indicará si hay espacios que se intersecan.

2.2.1. Código Fuente

```
class Point {
    constructor (x, y, userData ) {
        this.x = x;
        this.y = y;
        this.userData = userData;
    }
}

class Rectangle {
    constructor (x, y, w, h) {
        this.x = x; // center
        this.y = y;
```

```
        this.w = w; // half width
        this.h = h; // half height
    }
    // verifica si este objeto contiene un objeto Punto
    contains(point){
        if (point.x >= this.x - this.w && point.x <= this.x + this.w &&
            point.y >= this.y - this.h && point.y <= this.y + this.h){
            return 1;
        }
        return 0;
    }
    // verifica si este objeto se intersecta con otro objeto Rectangle
    intersects(range){
        return !(range.x - range.w > this.x + this.w ||
                 range.x + range.w < this.x - this.w ||
                 range.y - range.h > this.y + this.h ||
                 range.y + range.h < this.y - this.h);
    }
}
```

La clase QuadTree contiene tres funciones:

- **subdivide()**, como indica su nombre, dividirá el espacio en subespacios, cumpliendo ciertas condiciones.
- **insert(point)**, insertará un punto en un espacio solo si no sobrepasa el límite de capacidad, de lo contrario ejecutará la función *subdivide()*.
- **show()**, mostrará los datos ingresados de forma gráfica.

2.2.2. Código Fuente

```
class QuadTree {
    constructor(boundary, n) {
        this.boundary = boundary; // Rectangle
        this.capacity = n; // capacidad maxima de cada cuadrante
        this.points = []; // vector , almacena los puntos a almacenar
        this.divided = false;
    }
    // divide el quadtree en 4 quadtrees
    subdivide() {
        let x = this.boundary.x;
        let y = this.boundary.y;
        let w = this.boundary.w;
        let h = this.boundary.h;

        let r_northeast = new Rectangle(x + w/2, y + h/2, w/2, h/2);
        let r_northwest = new Rectangle(x - w/2, y + h/2, w/2, h/2);
        let r_southeast = new Rectangle(x + w/2, y - h/2, w/2, h/2);
        let r_southwest = new Rectangle(x - w/2, y - h/2, w/2, h/2);
        this.northeast = new QuadTree(r_northeast, this.capacity);
```

```
        this.northwest = new QuadTree(r_northwest, this.capacity);
        this.southeast = new QuadTree(r_southeast, this.capacity);
        this.southwest = new QuadTree(r_southwest, this.capacity);

        this.divided = true;
    }

    insert(point) {
        if (!this.boundary.contains(point)) {
            return ;
        }
        if (this.points.length < this.capacity) {
            this.points.push(point);
        }
        else {
            if (!this.divided) {
                this.subdivide();
            }
            this.northeast.insert(point);
            this.northwest.insert(point);
            this.southeast.insert(point);
            this.southwest.insert(point);
        }
    }

    show() {
        stroke(255);
        strokeWeight(1);
        noFill();
        rectMode(CENTER);
        rect(this.boundary.x, this.boundary.y, this.boundary.w*2 , this.boundar
        if (this.divided) {
            this.northeast.show();
            this.northwest.show();
            this.southeast.show();
            this.southwest.show();
        }
        for (let p of this.points) {
            strokeWeight(4);
            point(p.x, p.y);
        }
    }
}
```

2.3. Archivo sketch.js

En este archivo se crea un QuadTree y se mostrará en la página web creada anteriormente.

2.3.1. Código Fuente

```
let qt;
let count=0;

function setup() {
    createCanvas(400, 400);
    // centre point and half of width and height
    let boundary = new Rectangle(200, 200, 200, 200);
    // each leave just could have 4 elements
    qt = new QuadTree(boundary, 4);
    console.log(qt);
    for (let i=0; i<3; i++) {
        let p=new Point(Math.random() * 400, Math.random() * 400) ;
        qt.insert(p);
    }
    background(0);
    qt.show();
}
```

2.3.2. Función dibujar

```
let qt;
let count=0;

function setup() {
    createCanvas(400, 400);
    // centre point and half of width and height
    let boundary = new Rectangle(200, 200, 200, 200);
    // each leave just could have 4 elements
    qt = new QuadTree(boundary, 4);
    /*console.log(qt);
    for (let i=0; i<17; i++) {
        let p=new Point(Math.random() * 400, Math.random() * 400);
        qt.insert(p);
    }
    background(0);
    qt.show();*/
}

function draw() {
    background(0) ;
    if (mouseIsPressed) {
        for (let i = 0; i < 1; i++) {
            let m = new Point(mouseX + random(-5, 5), mouseY + random(-5, 5));
            qt.insert(m);
        }
    }
    background(0);
    qt.show();
}
```

El código modificado del archivo *sketch.js*, nos permite insertar puntos en el espacio generado. A continuación una breve explicación de esta función.
Dentro del espacio creado, cada vez que ocurra el evento de *hacer click*, se creará un objeto de la clase Point. Esta instancia recibe como parámetros las coordenadas del puntero del mouse, sumando un valor aleatorio entre -5 y 5 a cada uno. Luego lo inserta al QuadTree para finalmente mostrarlo en pantalla.

- **createCanvas(x,y)**, define las dimensiones en pixeles que ocupará un documento en pantalla.
- **background(0)**, define un color de fondo.
- **mouseIsPressed**, variable booleana.
- **mouseX, mouseY**; representan la coordenada x, y del mouse, respectivamente.

3. Enlaces

El código fuente se encuentra en el siguiente enlace:
https://github.com/oscar-pfuturi-h/EDA-git/tree/main/QuadTree

4. Pruebas

Figura 1: Resultado de ejecución con 3 datos

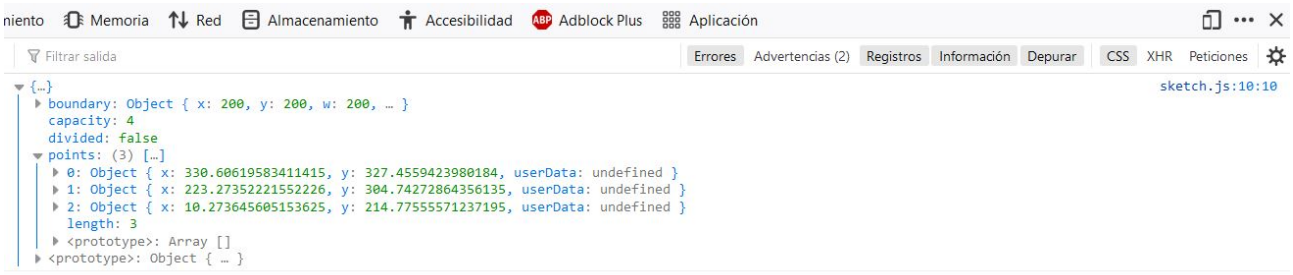

Figura 2: Vista de la consola

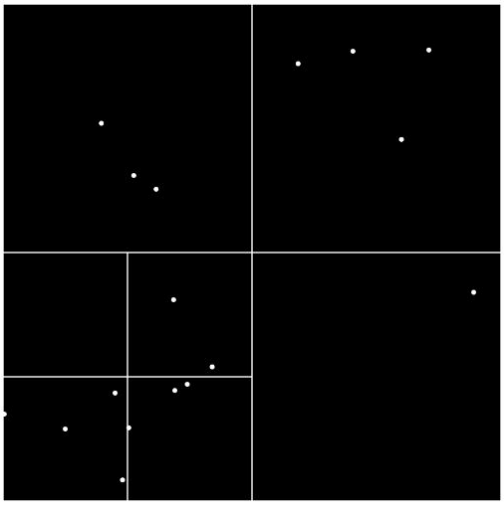

Figura 3: Resultado de ejecución con 17 datos

Figura 4: Vista de la consola

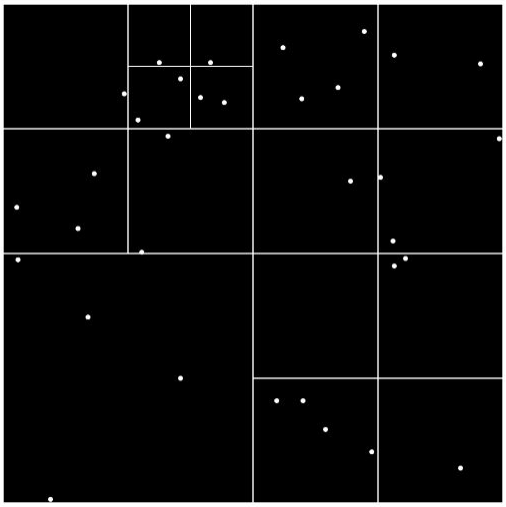

Figura 5: Resultado de ejecución con 33 datos

Figura 6: Vista de la consola

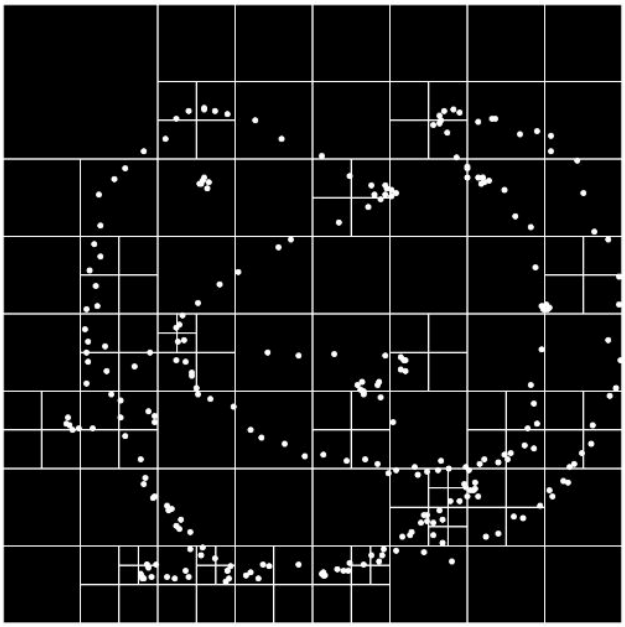

Figura 7: Resultado de ejecución con la función draw()