UNIVERSIDAD NACIONAL DE SAN AGUSTÍN

Facultad de Ingeniería de Producción y Servicios

Ciencia de la Computación

Práctica 3 - QuadTree

ALUMNO:
Pfuturi Huisa, Oscar David

PROFESOR:
MSc. Vicente Machaca Arceda

CURSO:
Estructura de Datos Avanzada

4 de octubre de 2020

Índice

1. Ejercicios

1.1. Edite el archivo *quadtree.js* y complete la función *query*.

1.1.1. 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);
        }
    }
```

```
    query(range, found) {//Rectangle & points, resp.
        if (this.boundary.intersects(range)) {
            if (this.divided) {
                this.northeast.query(range, found);
                this.northwest.query(range, found);
                this.southeast.query(range, found);
                this.southwest.query(range, found);
            }
            for (let p of this.points) {
                if (range.contains(p)) {
                    found.push(p);
                }
            }
        }
        return found;
    }

    show() {
        stroke(255);
        strokeWeight(1);
        noFill();
        rectMode(CENTER);
        rect(this.boundary.x, this.boundary.y, this.boundary.w*2, this.boundary
        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);
        }
    }
}
```

La función *query* recibe como parámetros un rectángulo, el cual es el área de los puntos que buscamos, y un arreglo donde se guardarán esos puntos.

1.2. Edite el archivo *sketch.js* con el siguiente código. Muestre sus resultados y comente.

1.2.1. Código Fuente

```
let qt;
let count=0;

function setup() {
    createCanvas(400, 400);
```

```
    let boundary = new Rectangle(200, 200, 200, 200);
    qt = new QuadTree(boundary, 4);
    console.log(qt);
    for (let i=0; i<25; i++) {
        let p=new Point(Math.random() * 400, Math.random() * 400);
        qt.insert(p);
    }
    background(0);
    qt.show();

    stroke(0,255,0);
    rectMode(CENTER);
    let range = new Rectangle(random(200), random(200), random(50), random(50)
    rect(range.x, range.y, range.w*2, range.h*2);
    let points = [];
    qt.query(range, points);

    for (let p of points) {
        strokeWeight(4);
        point(p.x, p.y);
    }

    console.log(count);
}
```

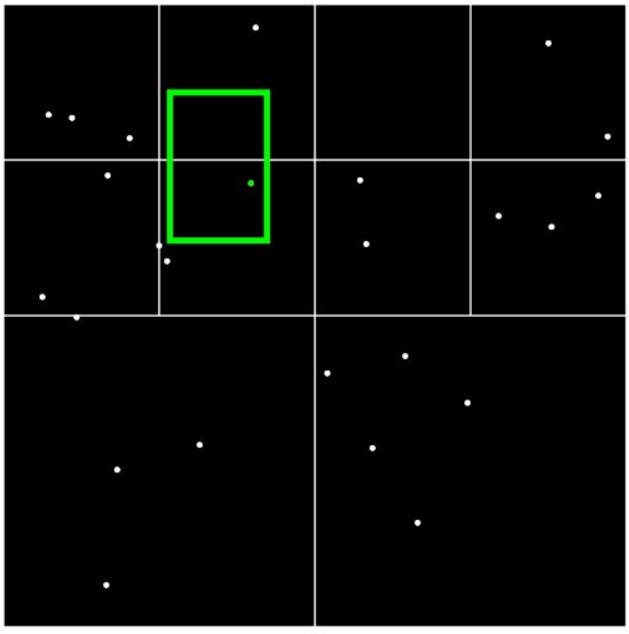

Figura 1: Resultado de ejecución de query con 25 datos

El código modificado crea un objeto Rectángulo de centro, ancho y altura con valores aleatorios y un arreglo.
Consiste en crear un área rectangular en la cual si un o varios puntos del QuadTree se encuentran

dentro de esa área, estas se almacenarán en el arreglo para luego mostrarlas en pantalla.

1.3. Conteo de puntos

En este caso vamos a verificar cuantas veces se consulta un punto en la función *query*, para esto usaremos la variable global count (ya esta definida en *sketch.js*). Agregue una instrucción en la función *query* donde se incremente el valor de count. Luego evalue y verifique cuantas veces se consultada un punto, prueba con mas de 1000 puntos en el Quadtree.

```
query(range, found) {//Rectangle & points, resp.
    if (this.boundary.intersects(range)) {
        if (this.divided) {
            this.northeast.query(range, found);
            this.northwest.query(range, found);
            this.southeast.query(range, found);
            this.southwest.query(range, found);
        }
        for (let p of this.points) {
            if (range.contains(p)) {
                found.push(p);
                count++;  // count increases
            }
        }

    }
}
```

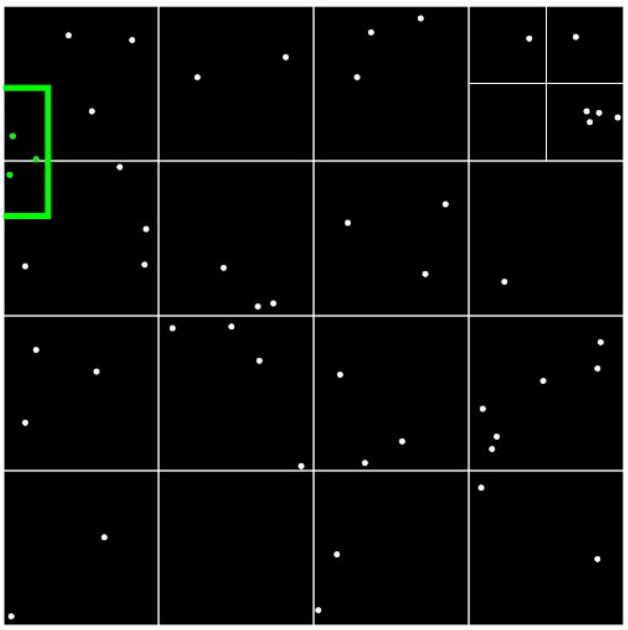

Figura 2: Resultado de ejecución con 50 datos

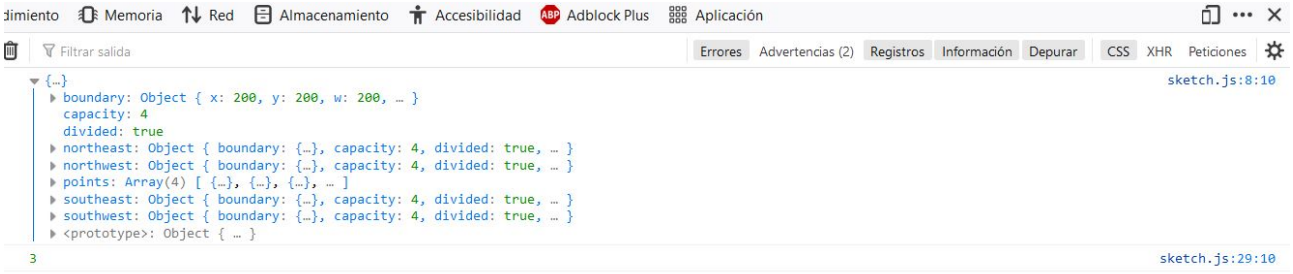

Figura 3: Vista de la consola con 50 datos

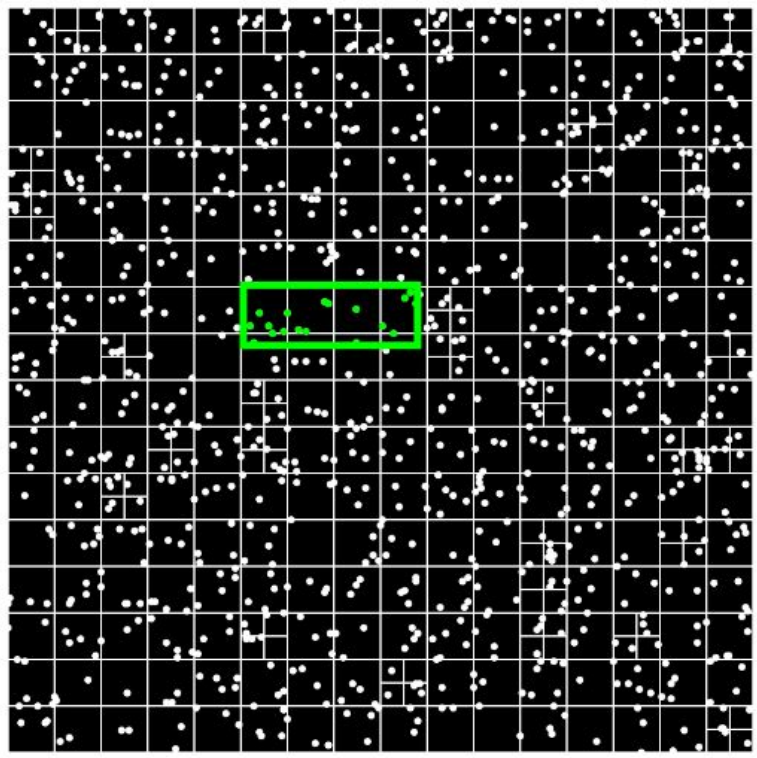

Figura 4: Resultado de ejecución con 1000 datos

1.4. Consultas con el mouse

Editemos el archivo *sketch.js*, en este caso haremos consultas con el mouse. Muestre sus resultados y comente.

```
let qt;
let count=0;

function setup() {
    createCanvas(400, 400);
    let boundary = new Rectangle(200, 200, 200, 200);
    qt = new QuadTree(boundary, 4);
    console.log(qt);

    for (let i=0; i<1000; i++) {
```

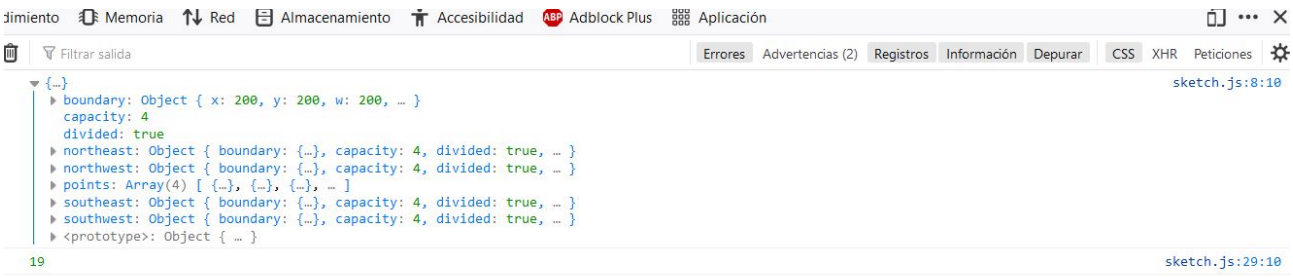

Figura 5: Vista de la consola con 1000 datos

```
        let p=new Point(Math.random() * 400, Math.random() * 400);
        qt.insert(p);
    }
    background(0);
    qt.show();
}

function draw() {
    background(0);
    qt.show();

    stroke(0,255,0);
    rectMode(CENTER);
    let range = new Rectangle(mouseX, mouseY, 50, 50);
    rect(range.x, range.y, range.w*2, range.h*2);
    let points = [];
    qt.query(range, points);

    for (let p of points) {
        strokeWeight(4);
        point(p.x, p.y);
    }
    console.log(count+" ");
}
```

Al hacer esta modificación, se asigna una área como centro al puntero del mouse, tal que los puntos del QuadTree que se encuentren dentro de esa área se agregarán al arreglo points y mostrar el cambio en pantalla.

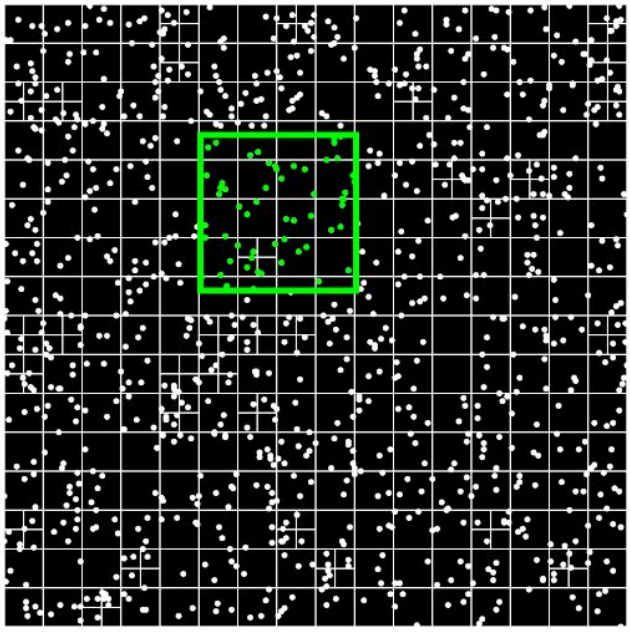

Figura 6: Resultado de ejecución con 1000 datos

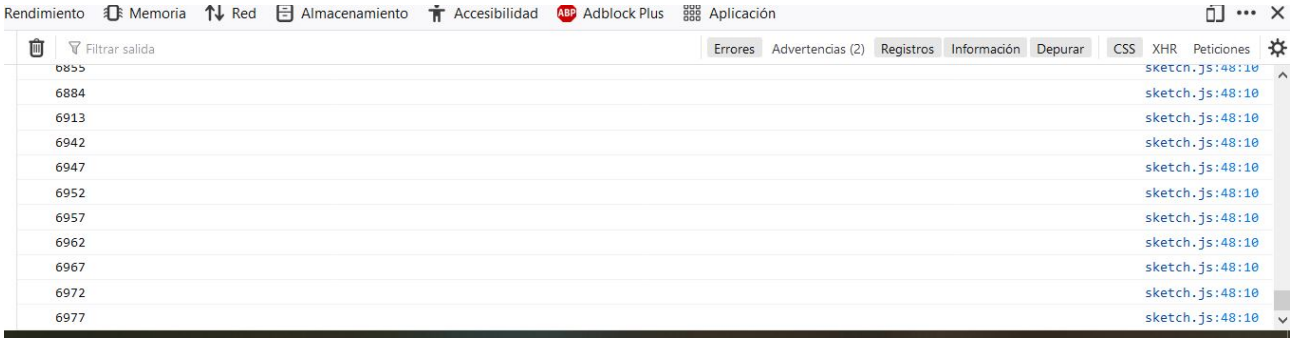

Figura 7: Vista de la consola

2. Enlaces

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