UNIVERSIDAD NACIONAL DE SAN AGUSTÍN

Facultad de Ingeniería de Producción y Servicios

Ciencia de la Computación

QuadTree

ALUMNOS:
Barrionuevo Paredes, Fabricio José

DOCENTE:
Mg.Vicente Machaca Arceda

CURSO:
Estructuras de Datos Avanzadas

30 de septiembre de 2020

Índice

1. Introducción

El QuadTree es una estructura de datos que nos ayuda a distribuir datos multidimensionales, cuya jerarquía se basa en un nodo principal que será todo un espacio donde se harán las divisiones conforme se llegue al límite de puntos o datos. Cuando ese límite se sobrepase, se procederá a crear nodos hijos en el respectivo cuadrante.

2. Ejercicios

2.1. Ejercicio 1

Cree un archivo main.html, este llamará a los archivos javascript que vamos a crear. El archivo p5.min.js es una librería para gráficos, la puede descargar de internet o se la puede pedir al profesor. En el archivo quadtree.js estará todo el código de nuestra estructura y en el archivo sketch.js estará el código donde haremos pruebas con nuestro Quadtree.

```
<html>
<head>
<title> QuadTree </title >
<script src="p5.min.js"></script>
<script src=" quadtree.js" ></script>
<script src="sketch.js"></script>
</head>
<body>
</body>
</html>
```

2.2. Ejercicio 2

En el archivo quadtree.js digitemos el siguiente código, además debe completar las funciones contains e intersects (ambas, devolverán true o false):

```
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;
        this.y=y;
        this.w=w;
        this.h=h;
    }
    contains(point){
        if (point.x >=this.x -this.w &&
            point.x <=this.w+this.x &&
```

```
            point.y>=this.y-this.h &&
            point.y <= this.y+this.h){
            return true;
        }
        return false
    }
    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))
    }
}
```

2.3. Ejercicio 3

En el archivo quadtree.js digitar el siguiente código y completar las funciones subdivide e insert:

```
class  QuadTree {
    constructor (boundary,n){
        this.boundary = boundary;
        this.capacity = n;
        this.points = [];
        this.divided = false;
    }
    subdivide(){
        let x=this.boundary.x;
        let y=this.boundary.y;
        let h=this.boundary.h;
        let w=this.boundary.w;

        let NO = new Rectangle(x + w/2,y-h/2,w/2,h/2);
        let NE = new Rectangle(x - w/2,y-h/2,w/2,h/2);
        let SO = new Rectangle(x + w/2,y+h/2,w/2,h/2);
        let SE = new Rectangle(x - w/2,y+h/2,w/2,h/2);

        this.hijoNO=new QuadTree(NO,this.capacity);
        this.hijoNE=new QuadTree(NE,this.capacity);
        this.hijoSO=new QuadTree(SO,this.capacity);
        this.hijoSE=new QuadTree(SE,this.capacity);

        this.divided=true;

    }
    insert(point){

        if (!this.boundary.contains(point)){
                return false;
```

```
            }
        if (this.points.length < this.capacity){
            this.points.push(point);
            return true;
        }else{
            if(!this.divided){
                this.subdivide();
            }
            this.hijoNO.insert(point);
            this.hijoNE.insert(point);
            this.hijoSO.insert(point);
            this.hijoSE.insert(point);
        }
    }
    show (){
        stroke (255);
        strokeWeight(1);
        noFill();
        rectMode(CENTER);
        rect(this.boundary.x,this.boundary.y,
        this.boundary.w*2,this.boundary.h*2);
        if (this.divided){
            this.hijoNO.show();
            this.hijoNE.show();
            this.hijoSO.show();
            this.hijoSE.show();
        }
        for ( let p of this.points){
            strokeWeight(4);
            point (p.x,p.y);
        }
    }
}
```

2.4. Ejercicio 4 y Ejercicio 5

Editemos el archivo sketch.js. En este archivo estamos creando un QuadTree de tamaño 400x400 con 3 puntos. Ejecute y comente los resultados (muestre capturas de pantalla).
Luego en la consola del navegador obtendremos las coordenadas de los puntos del QuadTree:

```
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<3;i++){
        let p= new Point(Math.random() * 400, Math.random()*400);
```

```
            qt.insert(p);
        }
        background(0);
        qt.show();
}
```

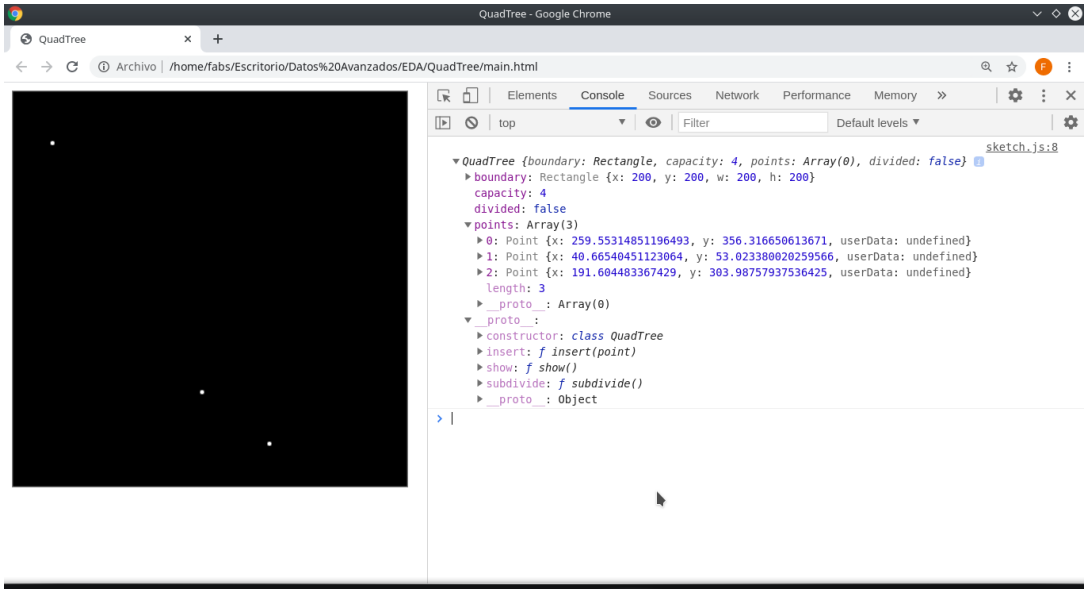

Figura 1: Points 3; Capacity 4

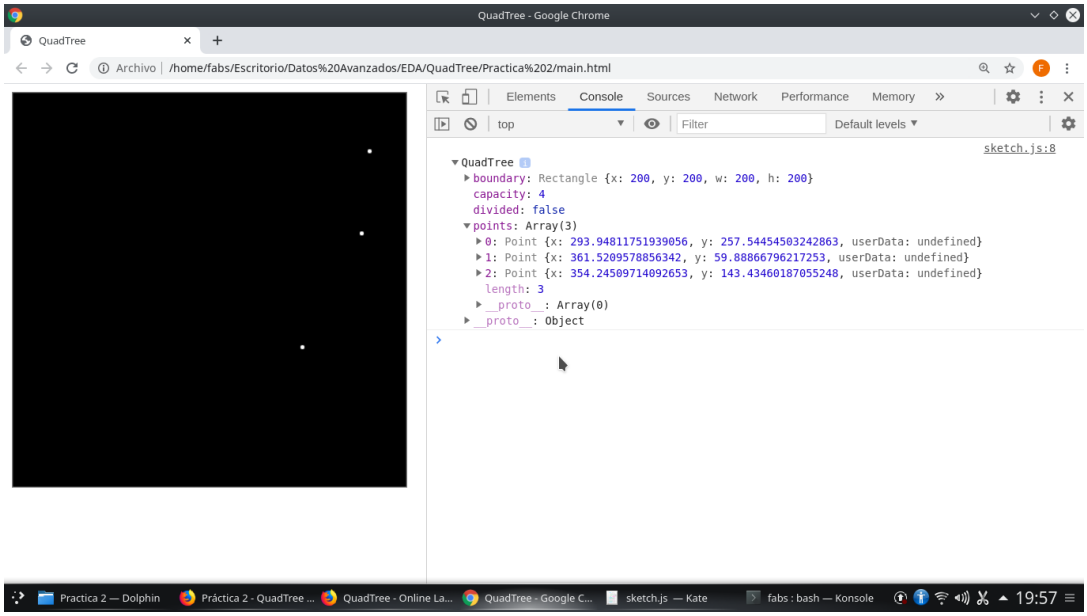

Figura 2: Points 3; Capacity 4

2.5. Ejercicio 6

Se insertan más puntos:

- En la figura 3 ingresamos 100 puntos y vemos a la derecha como se despliega la distribución de cada nodo hijo que se va creando conforme se llega al límite de capacidad.
- En la figura 4 aumentamos el límite de capacidad y vemos como no será necesario crear hijos dado que el rango es alto. Sólo se crean 4 hijos, teniendo un total de 5 nodos.

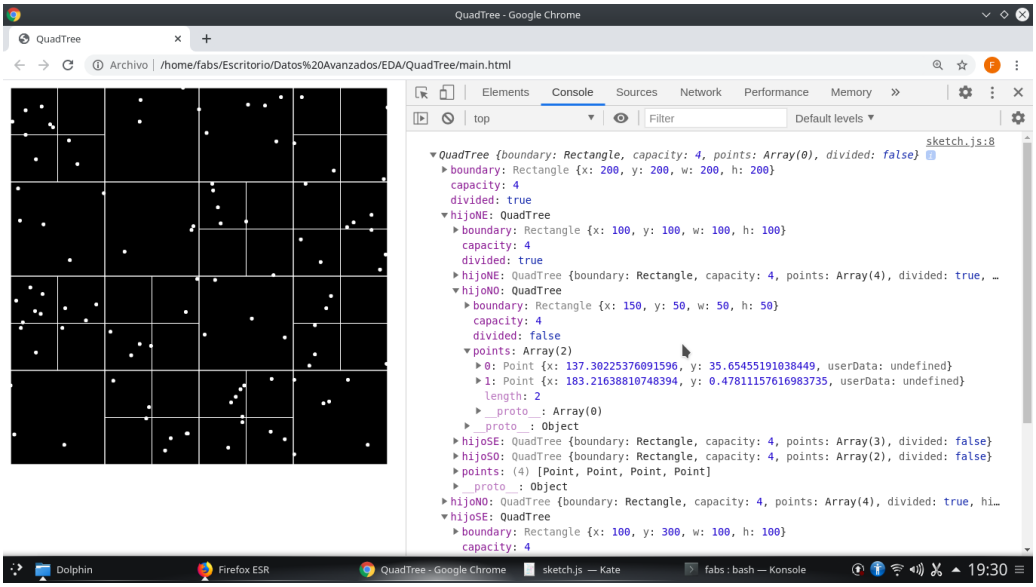

Figura 3: Points 100; Capacity 4

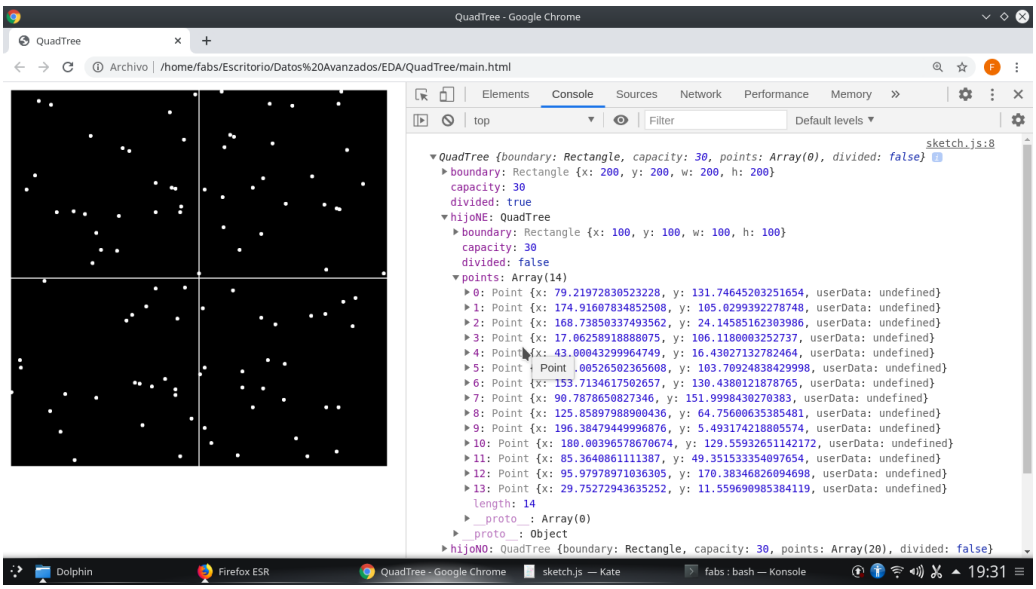

Figura 4: Poitns 100; Capacity 30

2.6. Ejercicio 7

Modificar el archivo sketch.js para insertar puntos con el mouse:

```
let qt;
let count=0;
function setup(){
    createCanvas(400,400);
    let boundary=new Rectangle(200,200,200,200);
    qt= new QuadTree(boundary,6);
}
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();
}
```

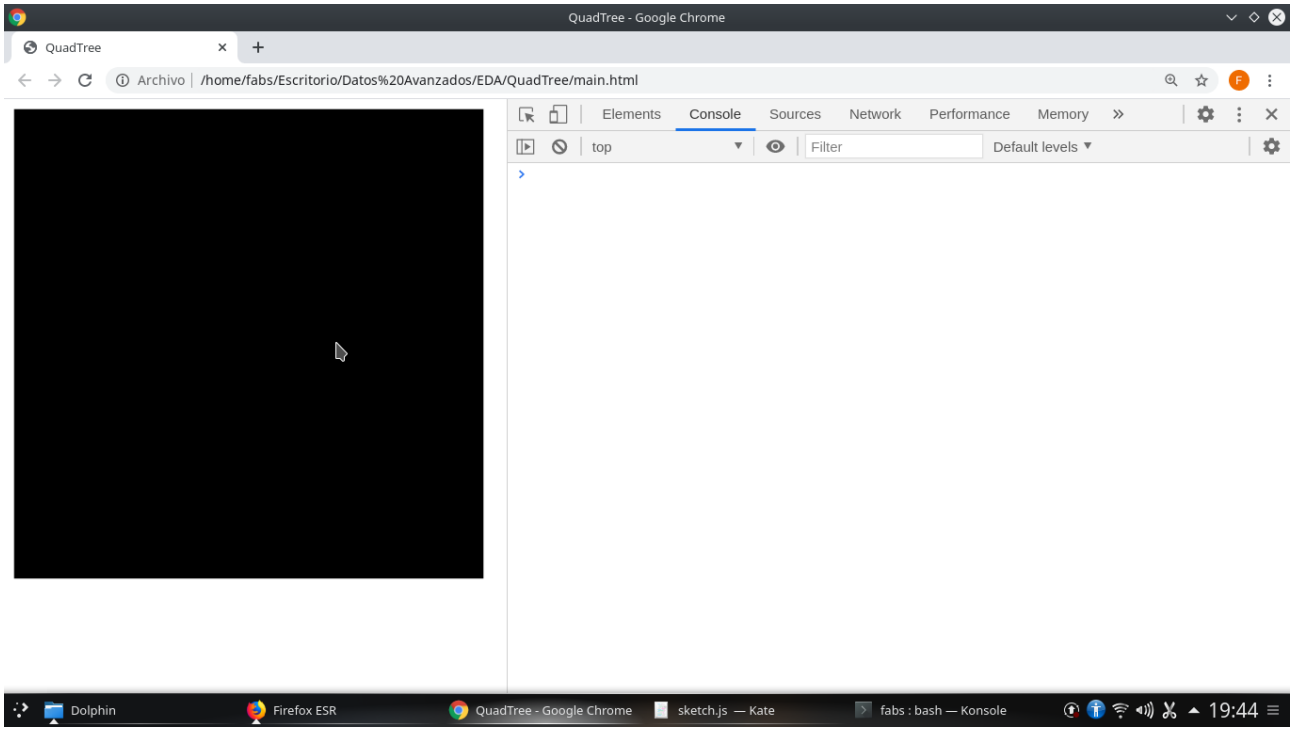

Figura 5: Cuadrante Vacío

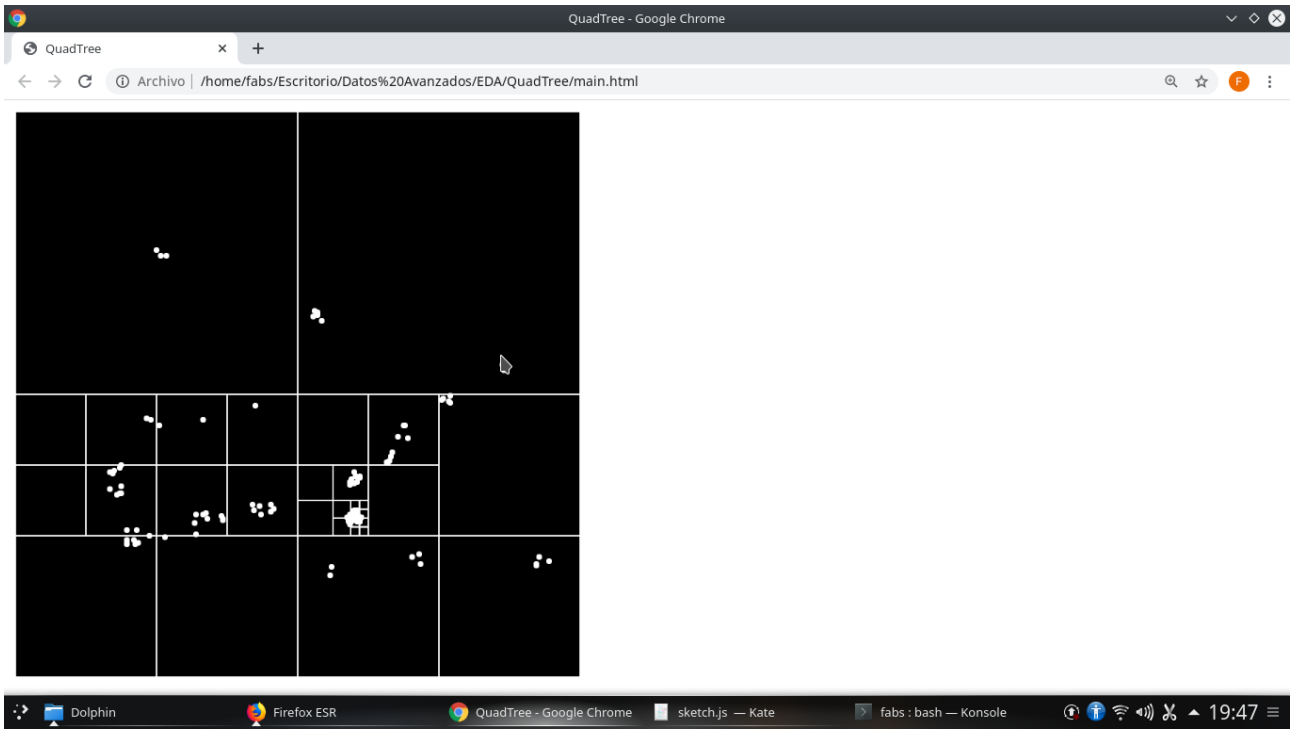

Figura 6: Insertado con Mouse

3. Enlaces

El código fuente puede encontrarse en el siguiente enlace:
https://github.com/Fabricio-Jose/EDA/tree/master/QuadTree/QuadTree_1