UNIVERSIDAD NACIONAL DE SAN AGUSTÍN

Facultad de Ingeniería de Producción y Servicios

Ciencia de la Computación

QuadTree: Query

ALUMNOS:
Barrionuevo Paredes, Fabricio José

DOCENTE:
Mg.Vicente Machaca Arceda

CURSO:
Estructuras de Datos Avanzadas

4 de octubre 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

Editar el archivo quadtree.js, agregando la funcion query:

```
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))
    }
}

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
        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);
        }
    }
    query(range, found){
        if(this.boundary.intersects(range) == false){
            return false;
        }else{
            for(let p of this.points){
                if(range.contains(p)){
                    found.push(p);
                }
            }
        }
        if(this.divided == true){
            this.hijoNO.query(range,found);
            this.hijoNE.query(range,found);
            this.hijoSO.query(range,found);
            this.hijoSE.query(range,found);
        }
        return true;
    }
}
```

2.2. Ejercicio 2

Edite el archivo sketch.js con el siguiente código. 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<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);
}
```

Luego de compilar, se puede ver un cuadro de color verde que va contabilizando los puntos existentes en esa área. En la consola se puede ver el número 0, que es el count que se devuelve gracias a la función console.log(count) y que, dado que no se modificó en ningún momento, devuelve 0.

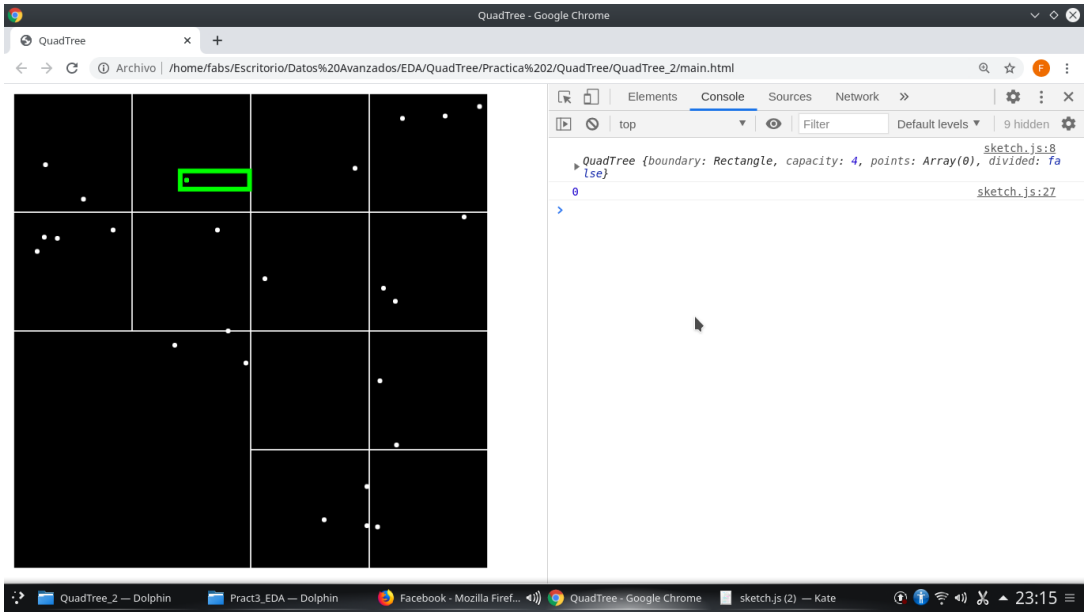

Figura 1: Query Estático

2.3. Ejercicio 3

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

```
query(range, found){
    if(this.boundary.intersects(range) == false){
        return false;
    }else{
        for(let p of this.points){
            if(range.contains(p)){
```

```
                    found.push(p);
                    //Variable que amentara por cada punto
                    count+=1;
                }
            }
        }
        if(this.divided == true){
            this.hijoNO.query(range,found);
            this.hijoNE.query(range,found);
            this.hijoSO.query(range,found);
            this.hijoSE.query(range,found);
        }
        return true;
    }
```

Archivo Sketch con mil datos

```
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++){
        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);
}
```

A continuación, se verá en la consola la cantidad de puntos que se encuentran en el área dentro del cuadrado verde seleccionado.

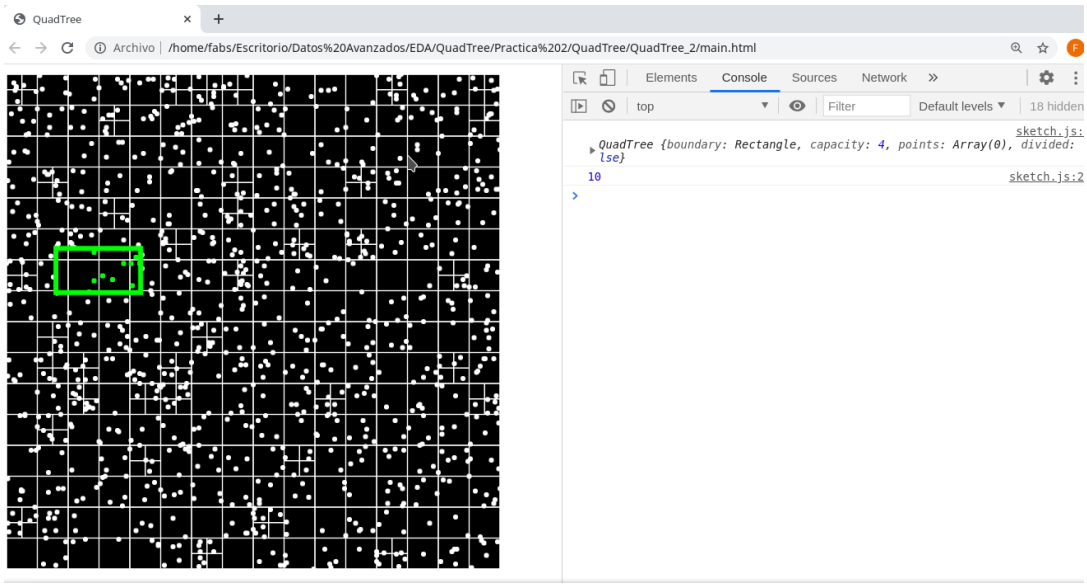

Figura 2: Count

2.4. Ejercicio 4

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<25;i++){
        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+" ");
}
```

Podemos ver ahora como se puede desplazar el cuadrado y contabilizarlos puntos automáticamente.

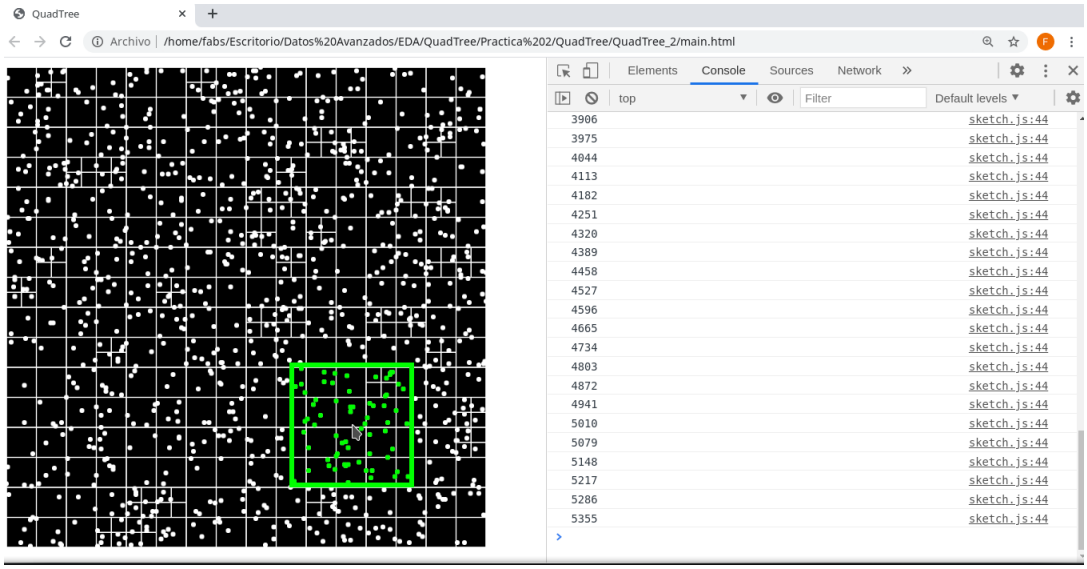

Figura 3

Figura 4

Figura 5

3. Enlaces

El código fuente puede encontrarse en el siguiente enlace:
https://github.com/Fabricio-Jose/EDA/tree/master/QuadTree/QuadTree_2