Universidad de Nacional de San Agustín
Facultad de Producción y Servicios
Escuela de Ciencia de la Computación

QuadTree

Practica 02

Integrantes: Mita Yagua Lesly Yaneth
Docente: Mg. Machaca Arceda Vicente

Fecha de entrega: 30 de septiembre de 2020
Arequipa, Perú

Índice de Contenidos

Índice de Figuras

Índice de Códigos

1. QuadTree

Quadtree es un árbol que se utiliza para almacenar de manera eficiente datos de puntos en un espacio bidimensional. En este árbol, cada nodo tiene como máximo cuatro hijos.

Esta estructura se utiliza en la compresión de imágenes, donde cada nodo contiene el color promedio de cada uno de sus hijos. Cuanto más profundo atraviese el árbol, mayor será el detalle de la imagen.

También como vimos en clase se utilizan para buscar nodos en un área bidimensional. Por ejemplo, si desea encontrar el punto más cercano a las coordenadas dadas, una estructura adecuada seria quadtree.

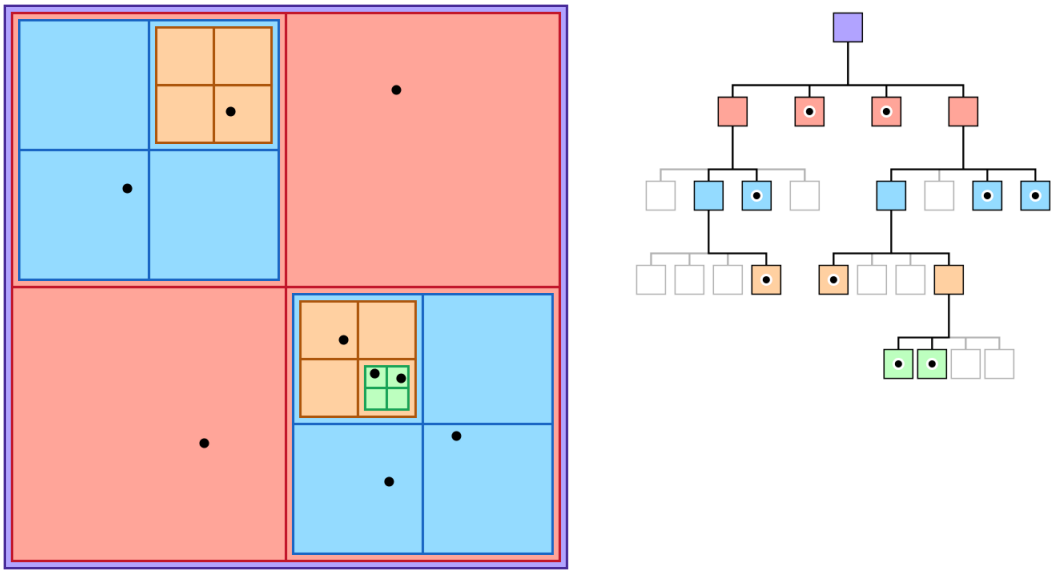


Figura 1: Quadtree

2. Resolución de Ejercicios

Ejercicio 01

Se crearon los siguientes archivos, dentro de una carpeta **src**:

- **main.html** Para llamar a todos los archivos javascript.
- **p5.min.js** Es una librería para los gráficos.
- **quadtree.js** Código para la estructura QuadTree.
- **sketch.js** Código para pruebas de la estructura.

La captura de pantalla del **Código 1** se encuentra en la sección Capturas de pantalla como **Figura 6**.

Código 1: main.html

```
<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>
```

Ejercicio 02

Implementación en el archivo quadtree.js de las funciones **contains** y **intersects** dentro de la clase Rectangle. La captura de pantalla del **Código 2** se encuentra en la sección Capturas de pantalla como **Figura 7**.

Código 2: Implementación de contains y intersects en Rectangle

```
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
```

```
14      }
15      // verifica  si  este objeto contiene un objeto Punto
16      contains ( point ){
17          return (point.x >= this.x - this.w &&
18              point.x <= this.x + this.w &&
19              point.y >= this.y - this.h &&
20              point.y <= this.y + this.h);
21      }
22      // verifica  si  este objeto se intersecta con otro objeto Rectangle
23      intersects ( range ){
24          return !(range.x - range.w > this.x + this.w ||
25              range.x + range.w < this.x - this.w ||
26              range.y - range.h > this.y + this.h ||
27              range.y + range.h < this.y - this.h);
28      }
29  }
```

Ejercicio 03

Implementación en el archivo quadtree.js de las funciones **subdivide** y **insert** dentro de la clase QuadTree. Siguiendo los algoritmos para cada funcion.

- **subdivide:** Divide el quadtree en 4 quadtrees
  - 1: Crear 4 hijos : qt_northeast , qt_northwest , qt_southeast , qt_southwest
  - 2: Asignar los QuadTree creados a cada hijo.
  - 3. - Hacer : this . divided <- true
- **insert:**
  - 1: Si el punto no esta en los limites (boundary) del quadtree Return
  - 2: Si (this . points . length) < (this . capacity)
  - 2.1 Insertamos en el vector this . points
  - Sino
  - 2.2 Dividimos si aun no ha sido dividido
  - 2.3 Insertamos recursivamente en los 4 hijos .

La captura de pantalla del **Código 3** se encuentra en la sección Capturas de pantalla como **Figura 8** y **Figura 9**.

Código 3: Implementación de subdivide y insert en QuadTree

```
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 / 2;
        let h = this.boundary.h / 2;

        let ne = new Rectangle(x + w, y - h, w, h);
        this.northeast = new QuadTree(ne, this.capacity);
        let nw = new Rectangle(x - w, y - h, w, h);
        this.northwest = new QuadTree(nw, this.capacity);
        let se = new Rectangle(x + w, y + h, w, h);
        this.southeast = new QuadTree(se, this.capacity);
        let sw = new Rectangle(x - w, y + h, w, h);
        this.southwest = new QuadTree(sw, 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.northeast.insert(point);
         this.northwest.insert(point);
         this.southeast.insert(point);
         this.southwest.insert(point);
      }
    }

    show () {
        stroke (255) ;
        strokeWeight (1) ;
        noFill () ;
        rectMode ( CENTER );
```

```
52          rect ( this . boundary .x, this . boundary .y, this . boundary .w*2 , this . boundary .h
    ↪ *2) ;
53          if( this . divided ){
54              this.northeast.show();
55              this.northwest.show();
56              this.southeast.show();
57              this.southwest.show();
58          }
59          for (let p of this . points ){
60              strokeWeight (4) ;
61              point (p.x, p.y);
62          }
63      }
64  }
```

Ejercicio 04

Se edito el archivo sketch.js creando un QuadTree de 400x400 con 3 puntos aleatorios. La captura de pantalla del **Código 4** se encuentra en la sección Capturas de pantalla como **Figura 10**.

Código 4: Implementación de QuadTree con 3 puntos aleatorios

```
1  let qt;
2  let count = 0;
3  function setup () {
4      createCanvas (400 ,400) ;
5      // centre point and half of width and height
6      let boundary = new Rectangle (200 ,200 ,200 ,200) ;
7      // each leave just could have 4 elements
8      qt = new QuadTree ( boundary , 4) ;
9      console .log (qt);
10
11     for (let i =0; i < 3; i ++) {
12         let p = new Point ( Math.random() * 400 , Math.random() * 400) ;
13         qt.insert(p);
14     }
15     background (0) ;
16     qt.show() ;
17 }
```

Los resultados de la ejecución del QuadTree de 3 puntos aleatorios son los que se presentan en la **Figura 2**.

- Se puede ver que la estructura recibió 3 puntos en su rango 400, 400. Sin embargo esta estructura necesita mas de 3 puntos para separarse por sus segmentos.
- Por lo tanto, no importa cuantas veces decidamos ejecutar el mismo sketch.js nunca se separar por segmentos, debido a que siempre serán 3 puntos y que su posición no importara.

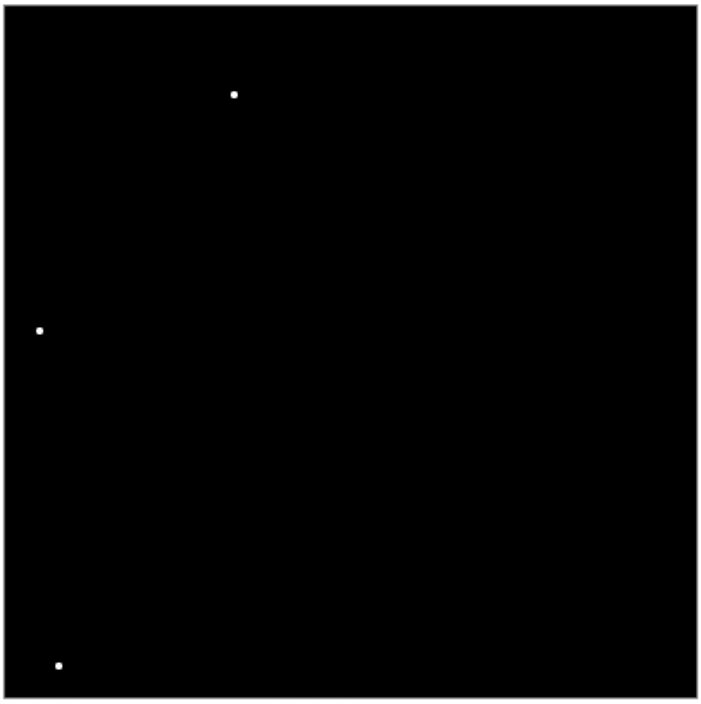


Figura 2: Visualización de QuadTree con tres puntos aleatorios

Ejercicio 05

Dentro de las opciones de desarrollador del navegador se obtuvo los siguientes resultados que se ve en la **Figura 3**.

- Se muestra el array de 3 donde se encuentran los 3 puntos. Cada uno con sus respectivas coordenadas en x y y.
- Justamente debido a su tamaño del array no se crean nuevos segmentos ya que este no ha sobrepasado su capacidad.

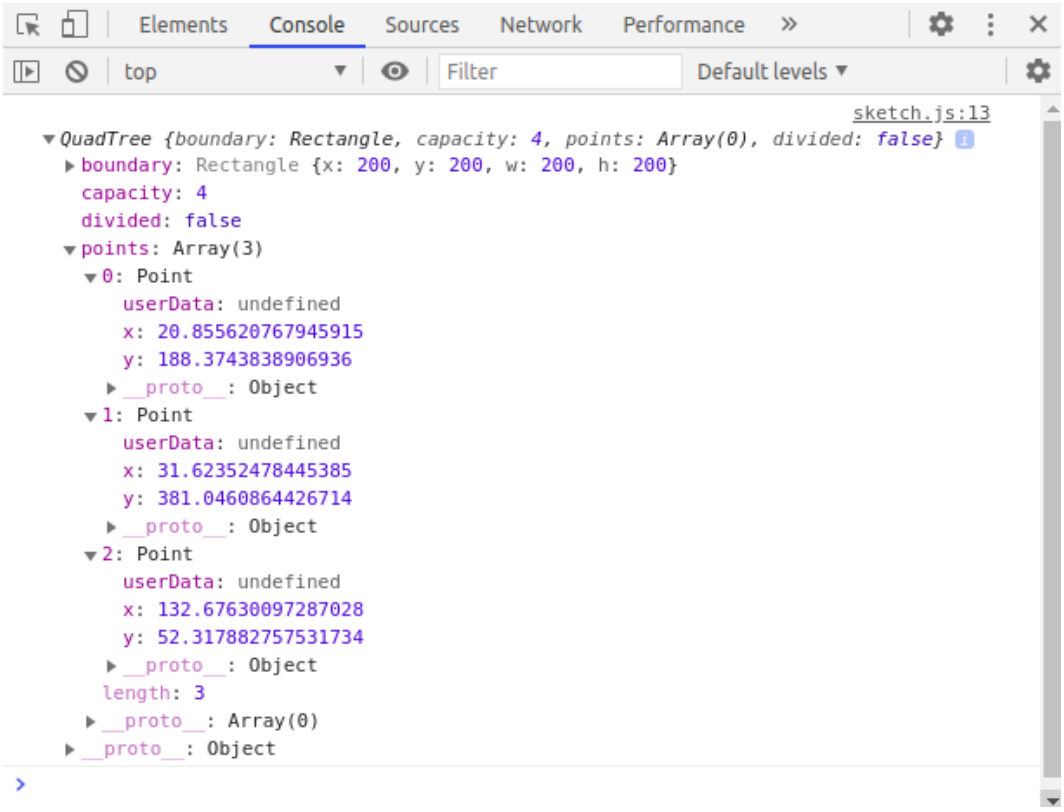
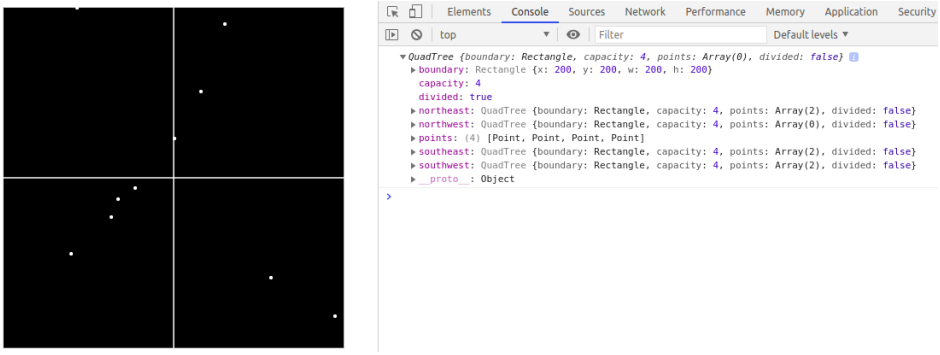


Figura 3: Vista a la Console de las opciones de desarrollador del navegador Web

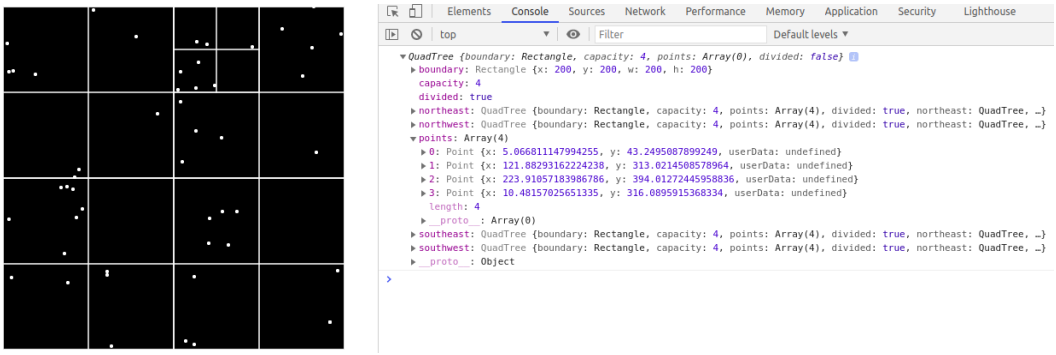
Ejercicio 06

Para insertar mas puntos se creo un nuevo archivo sketch1.js donde se modificara la cantidad de puntos. Este archivo es igual a skecth.js solo se modificara en el bucle for. Se probara con 10, 50, 100, 200. Los resultados que se ve en la **Figura 4**.

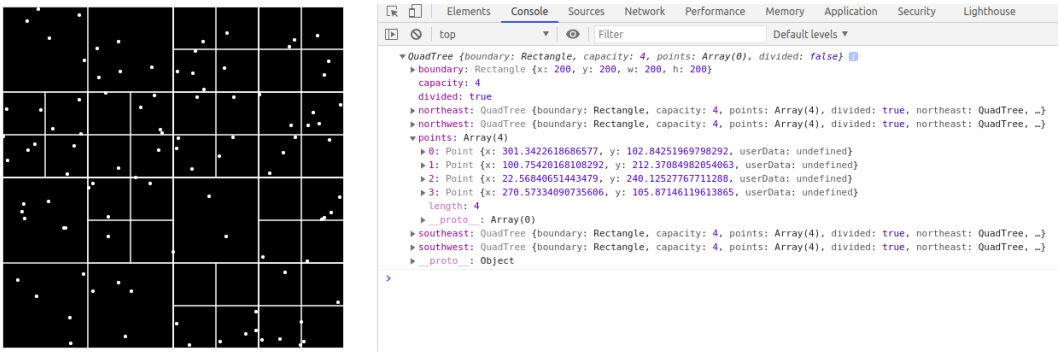
- En la figura podemos ver como aumenta la distribución de cada nodo hijo se va creando conforme la capacidad este completa.
- Sin embargo, esto se podría solucionar aumentando la capacidad, así se reducirían el numero de hijos.



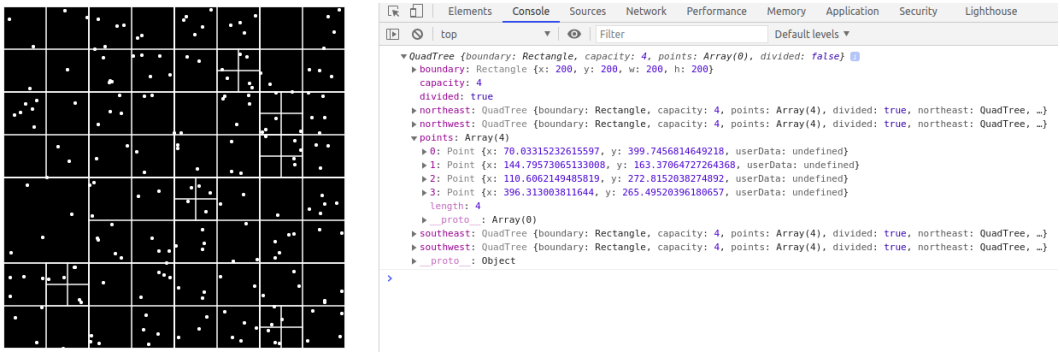
(a) Con 10 puntos



(b) Con 50 puntos



(c) Con 100 puntos



(d) Con 200 puntos

Figura 4: Resultados con mas puntos en QuadTree

Ejercicio 07

Se creo un archivo sketch2.js donde el código nos da la posibilidad de insertar los puntos con el mouse. La captura de pantalla del **Código 5** se encuentra en la sección Capturas de pantalla como **Figura 11**.

Código 5: Implementación de QuadTree con puntos con mouse

```
let qt;
let count = 0;

// Insertar puntos con el mouse
function setup () {
    createCanvas (400 ,400) ;
    let boundary = new Rectangle (200 ,200 ,200 ,200) ;
    qt = new QuadTree ( boundary , 4) ;
}

function draw () {
    background (0) ;
    // para agregar puntos si el mouse esta precionado
    if ( mouseIsPressed ) {
        for (let i = 0; i < 1; i ++) {
            // posision aleatoria para los puntos e insercion
            let m = new Point ( mouseX + random ( -5 ,5) , mouseY + random ( -5 ,5) );
            qt.insert(m)
        }
    }
    background (0);
    qt.show();
}
```

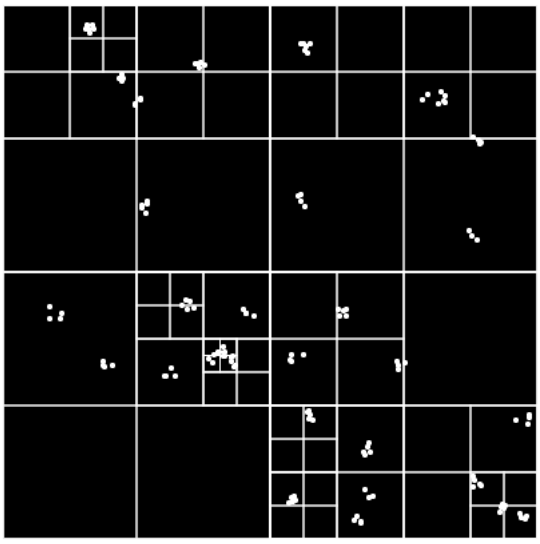


Figura 5: Visualización de QuadTree de puntos con mouse

3. Capturas de pantalla

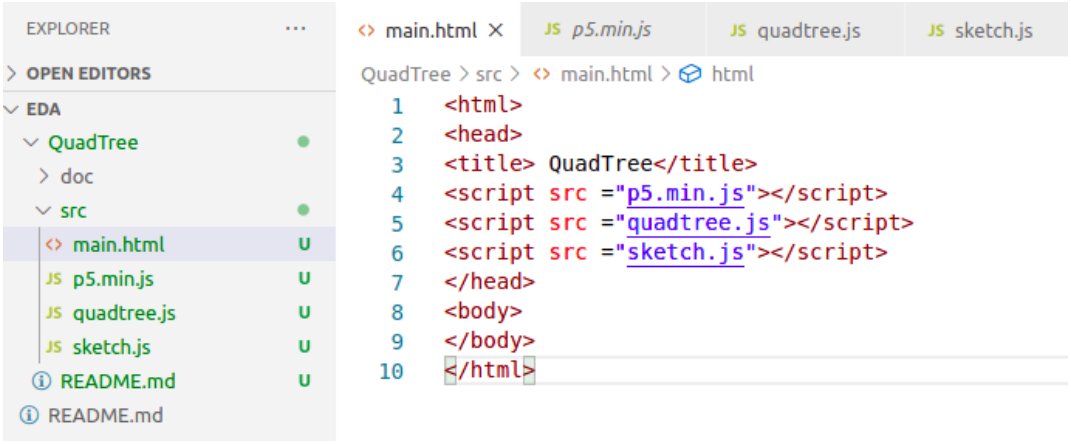


Figura 6: Captura de pantalla de main.html

QuadTree > src > quadtree.js > ...

```
1   class Point {
2       constructor (x, y, userData ){
3           this.x = x;
4           this.y = y;
5           this.userData = userData ;
6       }
7   }
8   class Rectangle {
9       constructor (x, y, w, h){
10          this.x = x; // center
11          this.y = y;
12          this.w = w; // half width
13          this.h = h; // half height
14      }
15      // verifica si este objeto contiene un objeto Punto
16      contains ( point ){
17          return (point.x >= this.x - this.w &&
18              point.x <= this.x + this.w &&
19              point.y >= this.y - this.h &&
20              point.y <= this.y + this.h);
21      }
22      // verifica si este objeto se intersecta con otro objeto Rectangle
23      intersects ( range ){
24          return !(range.x - range.w > this.x + this.w ||
25              range.x + range.w < this.x - this.w ||
26              range.y - range.h > this.y + this.h ||
27              range.y + range.h < this.y - this.h);
28      }
29  }
```

Figura 7: Captura de pantalla de la implementación de contains y intersects

QuadTree > src > JS quadtree.js > QuadTree > show

```
33
34  class QuadTree {
35      constructor ( boundary , n){
36          this.boundary = boundary ; // Rectangle
37          this.capacity = n; // capacidad maxima de cada cuadrante
38          this.points = []; // vector , almacena los puntos a almacenar
39          this.divided = false ;
40      }
41
42      // divide el quadtree en 4 quadtrees
43      subdivide () {
44          let x = this.boundary.x;
45          let y = this.boundary.y;
46          let w = this.boundary.w / 2;
47          let h = this.boundary.h / 2;
48
49          let ne = new Rectangle(x + w, y - h, w, h);
50          this.northeast = new QuadTree(ne, this.capacity);
51          let nw = new Rectangle(x - w, y - h, w, h);
52          this.northwest = new QuadTree(nw, this.capacity);
53          let se = new Rectangle(x + w, y + h, w, h);
54          this.southeast = new QuadTree(se, this.capacity);
55          let sw = new Rectangle(x - w, y + h, w, h);
56          this.southwest = new QuadTree(sw, this.capacity);
57
58          this.divided = true;
59      }
60
61      insert ( point ){
62          if (!this.boundary.contains(point)) {
63              return false;
64          }
65          if (this.points.length < this.capacity) {
66              this.points.push(point);
67              return true;
68          }
```

Figura 8: Captura de pantalla de la implementación de QuadTree - Parte 1

QuadTree > src > JS quadtree.js > QuadTree > show

```
65          if (this.points.length < this.capacity) {
66              this.points.push(point);
67              return true;
68          }
69          else{
70              if(!this.divided){
71                  this.subdivide();
72              }
73              this.northeast.insert(point);
74              this.northwest.insert(point);
75              this.southeast.insert(point);
76              this.southwest.insert(point);
77          }
78      }
79
80      show () {
81          stroke (255) ;
82          strokeWeight (1) ;
83          noFill () ;
84          rectMode ( CENTER );
85          rect ( this . boundary .x, this . boundary .y, this . boundary .w*2 ,
86              this . boundary .h *2) ;
87          if( this . divided ){
88              this.northeast.show();
89              this.northwest.show();
90              this.southeast.show();
91              this.southwest.show();
92          }
93          for (let p of this . points ){
94              strokeWeight (4) ;
95              point (p.x, p.y);
96          }
97      }
98  }
```

Figura 9: Captura de pantalla de la implementación de QuadTree - Parte 2

QuadTree > src > JS sketch.js > ...

```
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() ;
}
```

Figura 10: Captura de pantalla de sketch1.js para 3 puntos aleatorios

QuadTree > src > JS sketch2.js > ...

```
let qt;
let count = 0;

// Insertar puntos con el mouse
function setup () {
    createCanvas (400 ,400) ;
    let boundary = new Rectangle (200 ,200 ,200 ,200) ;
    qt = new QuadTree ( boundary , 4) ;
}

function draw () {
    background (0) ;
    // para agregar puntos si el mouse esta precionado
    if ( mouseIsPressed ) {
        for (let i = 0; i < 1; i ++) {
            // posision aleatoria para los puntos e insercion
            let m = new Point ( mouseX + random ( -5 ,5) , mouseY + random ( -5 ,5) );
            qt.insert(m)
        }
    }
    background (0);
    qt.show();
}
```

Figura 11: Captura de pantalla de sketch2.js para inserción de puntos con mouse

4. Repositorio

La practica 02 se encuentra en el siguiente Repositorio https://github.com/Leslym03/EDA/tree/master/QuadTree dentro de el se encuentra la carpeta **src** donde se esta el código fuente de esta practica y la carpeta **doc** donde se encuentra este documento.