

## Práctica 03

DOCENTE	CARRERA	CURSO
Vicente Machaca Arceda	Maestría en Ciencia de la Computación	Algoritmos y Estructura de Datos

PRÁCTICA	TEMA	DURACIÓN
03	Quadtrees	3 horas

### 1. Datos de los estudiantes

- Grupo: 07
- Integrantes:
  - Acuña Melo, Edgar
  - López Torres Herrera, Luis Rodrigo
  - Luna Flores, Julio Paolo
  - Vilela Arias, Roger

### 2. Introducción

En el siguiente trabajo se realizó el análisis e implementación de algoritmos eficientes que resuelven problemas computacionales, mediante la estructura de datos multidimensional QuadTree.






Un quadtree es una estructura de datos de árbol en la que cada nodo interno tiene exactamente cuatro hijos. Los Quadrees son el análogo bidimensional de los octrees y se usan con mayor frecuencia para dividir un espacio bidimensional subdividiéndolo recursivamente en cuatro cuadrantes o regiones. Los datos asociados con una celda de la hoja varían según la aplicación, pero la celda de la hoja representa una unidad de información espacial interesante”.

### 3. Desarrollo del Ejercicio

1. Cree un archivo main.html, este invocará 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 la estructura y en el archivo sketch.js estará el código donde haremos pruebas del Quadtree.

```
1  <html>
2    <head>
3      <title>QuadTree</title>
4      <script src="p5.min.js"></script>
5      <script src="quadtree.js"></script>
6      <script src="sketch.js"></script>
7    </head>
8    <body>
9  </body>
10 </html>
```

2. En el archivo quadtree.js digitemos el siguiente código, además debe completar las funciones `contains` y `intersects` (ambas funciones devuelven `true` o `false`).

	octree	2 days ago
	quadtree_vFinal	6 hours ago
	quatree_v1	6 hours ago
	Avance Informe	3 days ago
	Informe Practica 03.pdf	6 hours ago

```
1  class Point {
2    constructor(x, y, userData) {
3      this.x = x;
4      this.y = y;
5      this.userData = userData;
6    }
7  }
8
9  class Rectangle {
10   constructor(x, y, w, h) {
11     this.x = x; //center
12     this.y = y;
13     this.w = w; //half width
14     this.h = h; //half height
15   }
```

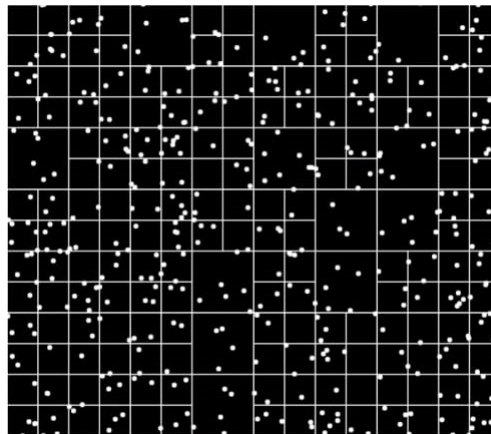
```
17 //TRABAJO
18 // verifica si este objeto contiene un objeto Punto
19 contains(point) {
20     if (point.x >= (this.x - this.w) &&
21         point.x < (this.x + this.w) &&
22         point.y >= (this.y - this.h) &&
23         point.y < (this.y + this.h))
24         return true;
25     return false;
26 }
27
28 //TRABAJO
29 // verifica si este objeto se interseca con otro objeto Rectangle
30 intersects(range) {
31     return !(
32         range.x - range.w > this.x + this.w ||
33         range.x + range.w < this.x - this.w ||
34         range.y - range.h > this.y + this.h ||
35         range.y + range.h < this.y - this.h
36     )
37 }
```

3. En el archivo quadtree.js digitemos el siguiente código y complete las funciones subdivide y insert.
4. Editemos el archivo sketch.js. En este archivo estamos creando un QuadTree de tama no 400x400 con 3 puntos. Ejecute (obentra un resultado similar a la Figura 1).
5. Abra las opciones de desarrollador (opciones/mas herramientas/ opciones de desarrollador) de su navegador para visualizar la console (Figura 2).
6. Edite el archivo sketch.js con el siguiente codigo. En este caso, nos da la posibilidad de insertar los puntos con el mouse.
7. Edite el archivo quadtree.js y complete la funcion query.

```
40 class QuadTree {
41     constructor(boundary, n) {
42         this.boundary = boundary; //Rectangle
43         this.capacity = n; // capacidad maxima de cada cuadrante
44         this.points = []; // vector donde almacena los puntos
45         this.divided = false;
46         this.northeast = null;
47         this.northwest = null;
48         this.southeast = null;
49         this.southwest = null;
50     }
51
52     //TRABAJO
53     // divide el quadtree en 4 quadtrees
54     subdivide() {
55         let x = this.boundary.x;
56         let y = this.boundary.y;
57         let w = this.boundary.w;
58         let h = this.boundary.h;
59     }
60 }
```

```
59
60     let qt_northeast = new Rectangle(x + (w / 2), y + (h / 2), w / 2, h / 2);
61     let qt_northwest = new Rectangle(x - (w / 2), y + (h / 2), w / 2, h / 2);
62     let qt_southeast = new Rectangle(x + (w / 2), y - (h / 2), w / 2, h / 2);
63     let qt_southwest = new Rectangle(x - (w / 2), y - (h / 2), w / 2, h / 2);
64
65     this.northeast = new QuadTree(qt_northeast, this.capacity);
66     this.northwest = new QuadTree(qt_northwest, this.capacity);
67     this.southeast = new QuadTree(qt_southeast, this.capacity);
68     this.southwest = new QuadTree(qt_southwest, this.capacity);
69
70     this.divided = true;
71 }
72
73 //TRABAJO
74 insert(point) {
75     if (!this.boundary.contains(point)) {
76         return false;
77     }
78
79     if (this.points.length < this.capacity && !this.divided) {
80         this.points.push(point);
81         return true;
82     } else {
83         if (!this.divided) {
84             this.subdivide();
85         }
86         this.points.push(point);
87         let points = this.points.slice();
88         for (let p of points) {
89             this.points.shift();
90             if (this.northeast.insert(p)) continue;
91             if (this.northwest.insert(p)) continue;
92             if (this.southeast.insert(p)) continue;
93             if (this.southwest.insert(p)) continue;
94         }
95         return true;
96     }
97 }
98
99 //TRABAJO
100 query(range, found) {
101     if (!found) {
102         found = [];
103     }
104     if (!this.boundary.intersects(range)) {
105         return found;
106     } else {
107         if (!this.divided) {
108             for (let p of this.points) {
109                 if (range.contains(p)) {
110                     found.push(p);
111                 }
112             }
113         } else {
114             found = this.northwest.query(range, found);
115             found = this.northeast.query(range, found);
116             found = this.southwest.query(range, found);
117             found = this.southeast.query(range, found);
118         }
119     }
120 }
```

```
124   show() {
125       stroke(255);
126       strokeWeight(1);
127       noFill();
128       rectMode(CENTER);
129       rect(this.boundary.x, this.boundary.y, this.boundary.w*2, this.boundary.h*2);
130
131       if (this.divided) {
132         this.northeast.show();
133         this.northwest.show();
134         this.southeast.show();
135         this.southwest.show();
136       }
137
138       for (let p of this.points) {
139         strokeWeight(4);
140         point(p.x, p.y);
141       }
142     }
143   }
```



8. Editemos el archivo sketch.js, En este caso haremos consultas con el mouse.

```
1  let qt;
2  let count = 0;
3
4  // PUNTO 04
5  /*
6  function setup() {
7    createCanvas(400, 400);
8    // centre point and half of width and height
9    let boundary = new Rectangle(200, 200, 200, 200);
10
11    // each leaf just could have 4 elements
12    qt = new QuadTree(boundary, 4);
13
14    console.log(qt);
15
16    for (let i=0; i < 3; i++) {
17      let p = new Point(Math.random() * 400, Math.random() * 400);
18      qt.insert(p);
19    }
```

```
1  let qt;
2  let count = 0;
3
4  // PUNTO 04
5  /*
6  function setup() {
7    createCanvas(400, 400);
8    // centre point and half of width and height
9    let boundary = new Rectangle(200, 200, 200, 200);
10
11    // each leave just could have 4 elements
12    qt = new QuadTree(boundary, 4);
13
14    console.log(qt);
15
16    for (let i=0; i < 3; i++) {
17      let p = new Point(Math.random() * 400, Math.random() * 400);
18      qt.insert(p);
19    }
20
21    background(0);
22    qt.show();
23  }
24  */
25
26  // PUNTO 06
27  /*
28  function setup() {
29    createCanvas(400, 400);
30    let boundary = new Rectangle(200, 200, 200, 200);
31    qt = new QuadTree(boundary, 4);
32  }
33
34  function draw() {
35    background(0);
36    if (mouseIsPressed) {
37      for (let i = 0; i < 1; i++) {
38        let m = new Point(mouseX + random(-5 ,5), mouseY + random(-5 ,5));
39        qt.insert(m);
40      }
41    }
42  }
```

```

41   }
42   background(0);
43   qt.show();
44 }
45 */
46
47 // PUNTO 08
48 function setup(){
49   createCanvas(400, 400);
50   let boundary = new Rectangle(200, 200, 200, 200);
51   qt = new QuadTree(boundary, 4);
52
53   console.log(qt);
54   for (let i=0; i < 40; i++) {
55     let p = new Point(Math.random() * 400, Math.random() * 400);
56     qt.insert(p);
57   }
58   background(0);
59   qt.show();
60 }






```

9. Finalmente, debe implementar un Octree y visualizarlo. Puede utilizar cualquier lenguaje de programación.

```

1  <html>
2  <head>
3    <title>Octree</title>
4    <script src=" p5.min.js "></script>
5    <script src="octree.js"></script>
6    <script src=" sketch.js "></script>
7  </head>
8  <body></body>
9  </html>

```

	octree	2 days ago
	quadtree_vFinal	6 hours ago
	quatree_v1	6 hours ago
	Avance Informe	3 days ago
	Informe Practica 03.pdf	6 hours ago

```
1  class Point {
2      constructor(x, y, z) {
3          this.x = x;
4          this.y = y;
5          this.z = z;
6      }
7  }
8
9  class Rectangle {
10     constructor(x, y, w, h) {
11         this.x = x; // center
12         this.y = y;
13         this.w = w; // half width
14         this.h = h; // half height
15     }
16 }
17
18 class Cube {
19     constructor(x, y, z, w, h, v) {
20         this.x = x; // center
21         this.y = y;
22         this.z = z;
23         this.w = w; // half width
24         this.h = h; // half height
25         this.v = v;
26
27
28     contains(point) {
29         return (
30             point.x >= this.x - this.w &&
31             point.x <= this.x + this.w &&
32             point.y >= this.y - this.h &&
33             point.y <= this.y + this.h &&
34             point.z >= this.z - this.v &&
35             point.z <= this.z + this.v
36         );
37     }
38
39     intersects(range) {
40         return !(
41             range.x - range.w > this.x + this.w ||
42             range.x + range.w < this.x - this.w ||
43             range.y - range.h > this.y + this.h ||
44             range.y + range.h < this.y - this.h ||
45             range.z - range.v > this.z - this.v ||
46             range.z + range.v < this.z + this.v
47         );
48     }
49 }
50
```

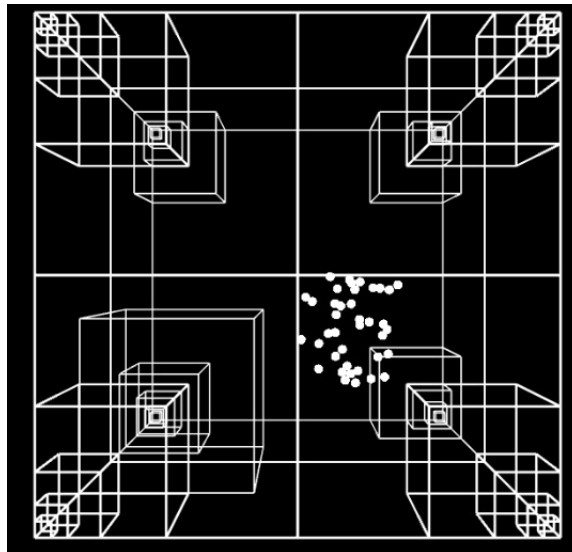


```
51 class Octree {
52     constructor(boundary, n) {
53         this.boundary = boundary; // Cube
54         this.capacity = n; // capacidad maxima de cada cuadrante
55         this.points = []; // vector , almacena los puntos a almacenar
56         this.divided = false;
57     }
58
59     subdivide() {
60         let x = this.boundary.x;
61         let y = this.boundary.y;
62         let z = this.boundary.z;
63         let w = this.boundary.w;
64         let h = this.boundary.h;
65         let v = this.boundary.v;
66
67         let qt_northeastFront = new Cube(x + w / 2, y + h / 2, z + v / 2, w / 2, h / 2, v / 2);
68         this.northeastFront = new Octree(qt_northeastFront, this.capacity);
69         let qt_northwestFront = new Cube(x + w / 2, y - h / 2, z + v / 2, w / 2, h / 2, v / 2);
70         this.northwestFront = new Octree(qt_northwestFront, this.capacity);
71         let qt_southeastFront = new Cube(x + w / 2, y + h / 2, z - v / 2, w / 2, h / 2, v / 2);
72         this.southeastFront = new Octree(qt_southeastFront, this.capacity);
73         let qt_southwestFront = new Cube(x + w / 2, y - h / 2, z - v / 2, w / 2, h / 2, v / 2);
74         this.southwestFront = new Octree(qt_southwestFront, this.capacity);
75
76         let qt_northeastBack = new Cube(x - w / 2, y + h / 2, z + v / 2, w / 2, h / 2, v / 2);
77         this.northeastBack = new Octree(qt_northeastBack, this.capacity);
78         let qt_northwestBack = new Cube(x - w / 2, y - h / 2, z + v / 2, w / 2, h / 2, v / 2);
79         this.northwestBack = new Octree(qt_northwestBack, this.capacity);
80         let qt_southeastBack = new Cube(x - w / 2, y + h / 2, z - v / 2, w / 2, h / 2, v / 2);
81         this.southeastBack = new Octree(qt_southeastBack, this.capacity);
82         let qt_southwestBack = new Cube(x - w / 2, y - h / 2, z - v / 2, w / 2, h / 2, v / 2);
83         this.southwestBack = new Octree(qt_southwestBack, this.capacity);
84     }
85
86     insert(point) {
87         if (!this.boundary.contains(point)) {
88             return false;
89         }
90
91         if (this.points.length < this.capacity) {
92             this.points.push(point);
93             return true;
94         } else {
95             if (!this.divided) {
96                 this.subdivide();
97                 this.divided = true;
98             }
99         }
100     }
101 }
```

```
101     this.northeastFront.insert(point);
102     this.northwestFront.insert(point);
103     this.southeastFront.insert(point);
104     this.southwestFront.insert(point);
105
106     this.northeastBack.insert(point);
107     this.northwestBack.insert(point);
108     this.southeastBack.insert(point);
109     this.southwestBack.insert(point);
110 }
111
112 show() {
113     stroke(255);
114     strokeWeight(1);
115     noFill();
116     box(this.boundary.w, this.boundary.h, this.boundary.v);
117
118     if (this.divided) {
119         this.northeastFront.show();
120         this.northwestFront.show();
121         this.southeastFront.show();
122         this.southwestFront.show();
123
124         this.northeastBack.show();
125         this.northwestBack.show();
126         this.southeastBack.show();
127         this.southwestBack.show();
128     }
129
130     for (let p of this.points) {
131         strokeWeight(5);
132         point(p.x, p.y, p.z);
133     }
134 }
135
136 query(range, found) {
137     if (!found) {
138         found = [];
139     }
140     if (!this.boundary.intersects(range)) {
141         return;
142     } else {
143         for (let p of this.points) {
144             if (range.contains(p)) {
145                 found.push(p);
146             }
147         }
148     }
```

```
150     if (this.divided) {
151         this.northeastFront.query(range, found);
152         this.northwestFront.query(range, found);
153         this.southeastFront.query(range, found);
154         this.southwestFront.query(range, found);
155
156         this.northeastBack.query(range, found);
157         this.northwestBack.query(range, found);
158         this.southeastBack.query(range, found);
159         this.southwestBack.query(range, found);
160     }
161     return found;
162 }
163 }

1  let qt;
2  let count = 0;
3
4  function setup() {
5      createCanvas(400, 400, WEBGL);
6      // centre point and half of width and height
7      let boundary = new Cube(100, 100, 100, 100, 100, 100);
8      // each leave just could have 4 elements
9      qt = new Octree(boundary, 4);
10
11     console.log(qt);
12     for (let i = 0; i < 4; i++) {
13         let p = new Point(Math.random() * 50, Math.random() * 50, Math.random() * 50);
14         qt.insert(p);
15     }
16
17     background(0);
18     qt.show();
19 }
```



## 4. Implementación

El desarrollo de la practica se utilizo un repositorio compartido para la practica, los archivos de codigo fuente del desarrollo de la presente practica, se encuentran alojados en el siguiente enlace: <https://github.com/JulioLunaUNSA/INFORME03.git>

## 5. Conclusiones

- El quadtree es una estructura de datos que representa una partición del espacio en dos dimensiones descomponiendo la región en cuatro cuadrantes iguales esta estructura facilita su utilizacion en aplicaciones donde se requiera buscar longitudes cortas, debido a su metdo de división en subcuadrantes y asi sucesivamente hasta tener regiones que contienen un único elemento. Cada nodo en el ´arbol tiene exactamente cuatro hijos o ninguno en el caso de las hojas.
- La estructura de datos Quadtree ofrece además como ventaja frente a otras estructuras de datos la eficiencia lo que se ve reflejado en un costo menor de tiempo de respuesta ya que es más liviano al consumir menos recursos,