

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	Add files via upload
	quatree	Add files via upload
	Avance Informe	Create Avance Informe
	main.html	Primera parte QuadTree
	p5.min.js	Primera parte QuadTree
	quadtree.js	Update quadtree.js
	sketch.js	Update sketch.js

```
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   }
16   // verifica si este objeto contiene un objeto Punto
17   contains(point) {
18     return (
19       point.x >= this.x - this.w &&
20       point.x <= this.x + this.w &&
21       point.y >= this.y - this.h &&
22       point.y <= this.y + this.h
23     );
24   }
25 }
```

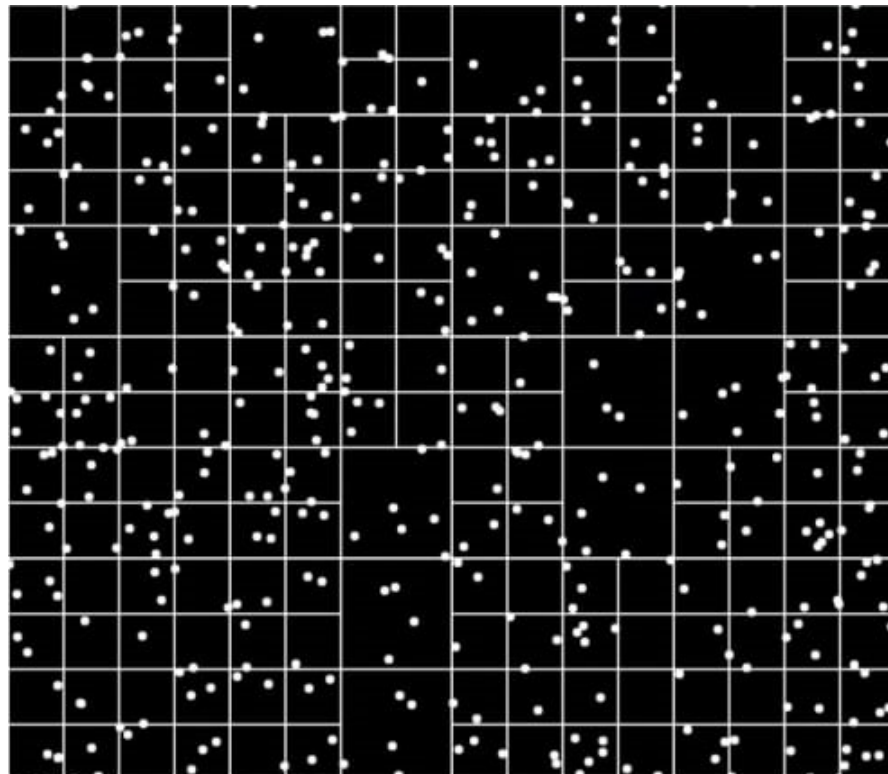
```
25 // verifica si este objeto se interseca con otro objeto Rectangle
26 intersects(range) {
27     return !(
28         range.x - range.w > this.x + this.w ||
29         range.x + range.w < this.x - this.w ||
30         range.y - range.h > this.y + this.h ||
31         range.y + range.h < this.y - this.h
32     );
33 }
34 }
35
36 class QuadTree {
37     constructor(boundary, n) {
38         this.boundary = boundary; // Rectangle
39         this.capacity = n; // capacidad maxima de cada cuadrante
40         this.points = []; // vector , almacena los puntos a almacenar
41         this.divided = false;
42     }
43     // divide el quadtree en 4 quadtrees
44     subdivide() {
```

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.

```
45 // Algoritmo
46 // 1: Crear 4 hijos: qt_northeast , qt_northwest , qt_southeast , qt_southwest
47 // 2: Asignar los QuadTree creados a cada hijo
48 // this.northeast = qt_northeast;
49 // this.northwest = qt_northwest;
50 // this.southeast = qt_southeast;
51 // this.southwest = qt_southwest;
52 // 3.- Hacer: this.divided <- true
53
54 let x = this.boundary.x;
55 let y = this.boundary.y;
56 let w = this.boundary.w;
57 let h = this.boundary.h;
58
59 let qt_northeast = new Rectangle(x + w / 2, y + h / 2, w / 2, h / 2);
60 this.northeast = new QuadTree(qt_northeast, this.capacity);
61 let qt_northwest = new Rectangle(x - w / 2, y + h / 2, w / 2, h / 2);
62 this.northwest = new QuadTree(qt_northwest, this.capacity);
63 let qt_southeast = new Rectangle(x + w / 2, y - h / 2, w / 2, h / 2);
64 this.southeast = new QuadTree(qt_southeast, this.capacity);
65 let qt_southwest = new Rectangle(x - w / 2, y - h / 2, w / 2, h / 2);
66 this.southwest = new QuadTree(qt_southwest, this.capacity);
67
68 this.divided = true;
69 }
```

```
72 // Algoritmo
73 // 1: Si el punto no esta en los limites ( boundary ) del quadtree Return
74 // 2: Si ( this.points.length ) < ( this.capacity ),
75 // 2.1 Insertamos en el vector this.points
76 // Sino
77 // 2.2 Dividimos si aun no ha sido dividido
78 // 2.3 Insertamos recursivamente en los 4 hijos.
79 // this.northeast.insert ( point );
80 // this.northwest.insert ( point );
81 // this.southeast.insert ( point );
82 // this.southwest.insert ( point );
83
84 if (!this.boundary.contains(point)) {
85     return false;
86 }
87
88 if (this.points.length < this.capacity) {
89     this.points.push(point);
90     return true;
91 } else {
92     if (!this.divided) {
93         this.subdivide();
94     }
95 }
96
97     this.northeast.insert(point);
98     this.northwest.insert(point);
99     this.southeast.insert(point);
100    this.southwest.insert(point);
101 }
102
103 show() {
104     stroke(255);
105     strokeWeight(1);
106     noFill();
107     rectMode(CENTER);
108     rect(
109         this.boundary.x,
110         this.boundary.y,
111         this.boundary.w * 2,
112         this.boundary.h * 2
113     );
114     if (this.divided) {
115         this.northeast.show();
116         this.northwest.show();
117         this.southeast.show();
118         this.southwest.show();
119     }
```

```
121     for (let p of this.points) {
122         strokeWeight(4);
123         point(p.x, p.y);
124     }
125 }
126
127 query(range, found) {
128
129     if (!found) {
130         found = [];
131     }
132     if (!this.boundary.intersects(range)) {
133         return;
134     } else {
135         for (let p of this.points) {
136             if (range.contains(p)) {
137                 found.push(p);
138             }
139         }
140     }
141
142     if (this.divided) {
143         this.northwest.query(range, found);
144         this.northeast.query(range, found);
145         this.southwest.query(range, found);
146         this.southeast.query(range, found);
147     }
148     return found;
149 }
150 }
```







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

```
1  let qt;
2  let count = 0;
3
4  function setup() {
5    createCanvas(400, 400);
6    // centre point and half of width and height
7    let boundary = new Rectangle(200, 200, 200, 200);
8    // each leave just could have 4 elements
9    qt = new QuadTree(boundary, 4);
10
11   console.log(qt);
12   for (let i = 0; i < 5; i++) {
13     let p = new Point(Math.random()*100, Math.random()*100);
14     qt.insert(p);
15   }
16
17   background(0);
18   qt.show();
19 }
```

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

```
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>
```

 main.html	Add files via upload
 octree.js	Add files via upload
 p5.min.js	Add files via upload
 sketch.js	Add files via upload

```
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
29 contains(point) {
30     return (
31         point.x >= this.x - this.w &&
32         point.x <= this.x + this.w &&
33         point.y >= this.y - this.h &&
34         point.y <= this.y + this.h &&
35         point.z >= this.z - this.v &&
36         point.z <= this.z + this.v
37     );
38 }
39
40 intersects(range) {
41     return !(
42         range.x - range.w > this.x + this.w ||
43         range.x + range.w < this.x - this.w ||
44         range.y - range.h > this.y + this.h ||
45         range.y + range.h < this.y - this.h ||
46         range.z - range.v > this.z - this.v ||
47         range.z + range.v < this.z + this.v
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     }
149 }
```

```
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 así 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,