Práctica 01

DOCENTE	CARRERA	CURSO
MSc. Vicente Machaca Arceda	Escuela Profesional de Ciencia de la Computación	Estructura de Datos Avanzada

PRÁCTICA	TEMA	DURACIÓN
01	QuadTree	10 horas

1. Competencias del curso

- Conocer e investigar los métodos de acceso multidimensional, métrico y aproximado.
- Analiza, diseña y propone soluciones utilizando estructuras de datos avanzadas.
- Comprende la importancia e impacto de los algoritmos estudiados y las nuevas propuestas.
- Aplica principios matemáticos para la solución de problemas.

2. Competencias de la práctica

- Comprende e implementa la estructura multidimensional *QuadTree*.

3. Equipos y materiales

- Javascript
- Navegador Web
- Cuenta en Github
- IDE de desarrollo

4. Entregables

- Se debe elaborar un informe en Latex donde se responda a cada ejercicio de la Sección 5.
- En el informe se debe agregar un enlace al repositorio Github donde esta el código.
- En el informe se debe agregar el código fuente asi como capturas de pantalla de la ejecución y resultados del mismo.

5. Ejercicios

1. Cree un archivo *main.html*, este llamara 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. En el archivo *quadtree.js* digitemos el siguiente código, además debe completar las funciones *contains* y *intersects* (ambas funciones devuelven 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; //center
    this.y = y;
    this.w = w; //half width
    this.h = h; //half height
}

  // verifica si este objeto contiene un objeto Punto
  contains(point){

  }

  // verifica si este objeto se intersecta con otro objeto Rectangle
  intersects(range){

  }
}
```

3. En el archivo *quadtree.js* digitemos el siguiente código y complete las funciones *subdivide* y *insert*.

```
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(){
    //  Algoritmo
    //  1: Crear 4 hijos: qt_northeast, qt_northwest, qt_southeast, qt_southwest
```

```
    //  2: Asignar los QuadTree creados a cada hijo
    //    this.northeast = qt_northeast;
    //    this.northwest = qt_northwest;
    //    this.southeast = qt_southeast;
    //    this.southwest = qt_southwest;

    //  3.- Hacer: this.divided <- true
  }

  insert(point){
    //  Algoritmo
    //  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.
    //          this.northeast.insert(point);
    //          this.northwest.insert(point);
    //          this.southeast.insert(point);
    //          this.southwest.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.northeast.show();
      this.northwest.show();
      this.southeast.show();
      this.southwest.show();
    }

    for (let p of this.points){
      strokeWeight(4);
      point(p.x, p.y);
    }
  }
}
```

4. Editemos el archivo *sketch.js*. En este archivo estamos creando un QuadTree de tamaño 400x400 con 3 puntos. Ejecute (obentra un resultado similar a la Figura 1) y comente los resultados (muestre capturas de pantalla).

```
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();
}
```

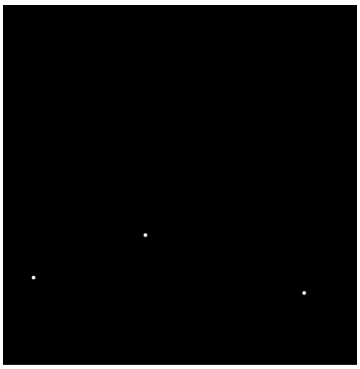

Figura 1: Visualización del QudTree con 3 datos.

5. Abra las opciones de desarrollador (opciones/más herramientas/ opciones de desarrollador) de su navegador para visualizar la *console* (Figura 2). Comente que datos encuentra y muestre una captura de pantalla.

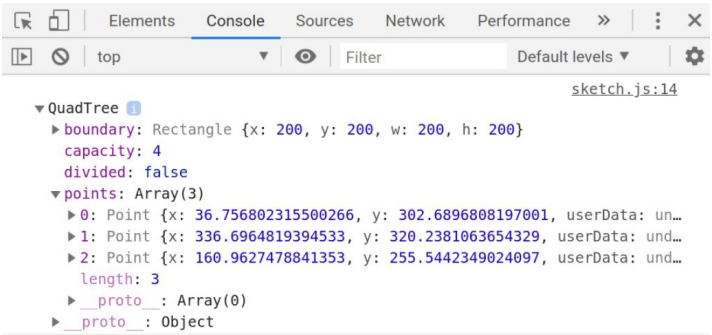

Figura 2: Vista a la *Console* de las opciones de desarrollador del navegador Web.

6. Inserte más puntos y muestre cómo varían sus resultados.

7. Edite el archivo *sketch.js* con el siguiente código. En este caso, nos da la posibilidad de insertar los puntos con el mouse. Muestre sus resultados y comente cómo funciona el código.

```
let qt;
let count = 0;

function setup(){
  createCanvas(400,400);
  let boundary = new Rectangle(200,200,200,200);
  qt = new QuadTree(boundary, 4);
}
```

```
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();
}
```

6. Rúbricas

Rúbrica	**Cumple**	**Cumple con obs.**	**No cumple**
Informe: El informe debe estar en Latex, con un formato limpio, buena presentación y redacción.	5	2.5	0
Implementación: Ha desarrollado todas las actividades solicitadas en la práctica.	10	5	0
Presentación: El alumno demuestra dominio del tema y conoce con exactitud cada parte de su trabajo.	5	2.5	0
Errores ortográficos: Por cada error ortográfico, se le descontará un punto.	-	-	-