Análisis de Algoritmos

Proyecto Final

Nombres:

Martínez Partida Jair Fabian

Martínez Rodríguez Alejandro

Monteros Cervantes Miguel Angel

Luis Fernando Ramírez Cotonieto

Fecha de entrega:21 de Junio del 2021

Grupo:3CM13

Índice

Proyecto Final

Análisis de Algoritmos

1. Planteamiento del problema

En geometría computacional, el problema del par de puntos más cercano es un problema clásico donde ”Dados n puntos en un espacio métrico, se pide encontrar un par de puntos con la distancia más pequeña entre ellos”. El problema del par de puntos más cercano en el plano euclidiano fue de los primeros problemas tratados en el estudio sistemático de la complejidad computacional de algoritmos geométricos.
Un algoritmo ingenuo para resolver el problema consiste en çalcular las distancias entre todos los pares de puntos del conjunto y seleccionar el mínimo”, que requiere un tiempo O(n^2). Pero resulta que el problema puede ser solucionado en tiempo O(n log n) en un espacio euclídeo. Este tiempo puede incluso ser mejorado: Si asumimos que la función de parte entera (floor) es computable en tiempo constante, el problema puede ser solucionado en tiempo O(n log log n). Si además permitimos utilizar aleatorización, el problema puede ser solucionado en tiempo O(n).

2. Planteamiento del Algoritmo y Solución

2.1. Tecnologías utilizadas

Se utilizó **JavaScript** (abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.
Junto con la librería **Raphael.js** llamado así por el artista Raffaello Sanzio da Urbino, Rapahel.js es una biblioteca javascript diseñada específicamente para artistas y diseñadores gráficos. Es el pincel que puedes utilizar para aplicar imágenes directamente al lienzo del navegador.

2.2. Divide y Conquista

Un enfoque simple es hacer una búsqueda lineal, es decir: Se nos da una matriz de n puntos en el plano, y el problema es averiguar el par de puntos más cercano en la matriz. Este problema surge en una serie de aplicaciones. Por ejemplo, en el control de tráfico aéreo, es posible que desee monitorear los aviones que se acercan demasiado, ya que esto puede indicar una posible colisión. Recuerde la siguiente fórmula para la distancia entre dos puntos p y q.

$$\|pq\| = \sqrt{(px - qx)^2 + (py - qy)^2}$$

La solución de fuerza bruta es O(n^2), calcula la distancia entre cada par y devuelve la más pequeña. Se puede calcular la distancia más pequeña en el tiempo O(nLogn) usando la estrategia Divide y Conquer.
Pasos:

Entrada: Un array de n puntos P[]
Salida: La distancia más pequeña entre dos puntos en el array dado.

Como paso de procesamiento previo, la matriz de entrada se ordena según las coordenadas x.

1. Encontrar el punto medio en el array ordenado, podemos tomar P[n/2] como punto medio.
2. Divida el array dado en dos mitades. El primer subarray contiene puntos de P[0] a P[n/2]. El segundo subarray contiene puntos de P[n/2+1] a P[n-1].
3. Encontrar recursivamente las distancias más pequeñas en ambos subarrays. Dejando que las distancias sean dl y dr. Encontrar el mínimo de dl y dr. Dejando que el mínimo sea d.
4. A partir de los 3 escalones anteriores, tenemos un límite superior d de distancia mínima. Ahora tenemos que considerar los pares de tal manera que un punto en el par sea de la mitad izquierda y el otro sea de la mitad derecha. Considere la línea vertical que pasa por P[n/2] y encuentre todos los puntos cuya coordenada x está más cerca que d de la línea vertical media. Construya una franja de matriz[] de todos esos puntos.
5. Se ordena la franja de matriz[] según las coordenadas y este paso es O(nLogn). Se puede optimizar a O(n) ordenando y fusionando recursivamente.
6. Se encuentra la distancia más pequeña en la tira[]. Esto es complicado. Desde el primer vistazo, parece ser un paso O(n^2), pero en realidad es O(n). Se puede probar geométricamente que para cada punto de la tira, solo necesitamos comprobar como máximo 7 puntos después de ella (tenga en cuenta que la tira está ordenada de acuerdo con la coordenada Y).
7. Finalmente devuelva el mínimo de d y la distancia calculada en el paso anterior. Implementación

3. Representación del Algoritmo

3.1. Tecnología utilizada

Adobe XD es un editor de gráficos vectoriales desarrollado y publicado por Adobe Inc para diseñar y crear un prototipo de la experiencia del usuario para páginas web y aplicaciones móviles. El software está disponible para MacOS y Windows. Adobe XD apoya a los diseño vectoriales y a los sitios web wireframe, y creando prototipos simples e interactivos con un solo click.

3.2. Makups

3.2.1. Pantalla de Inicio

Esta es la visualización al entrar a la página web.

Figura 1: Pantalla Inicial

3.2.2. Colocación de Puntos

Se colocan los puntos en la pantalla azul y se preciona ïniciar".

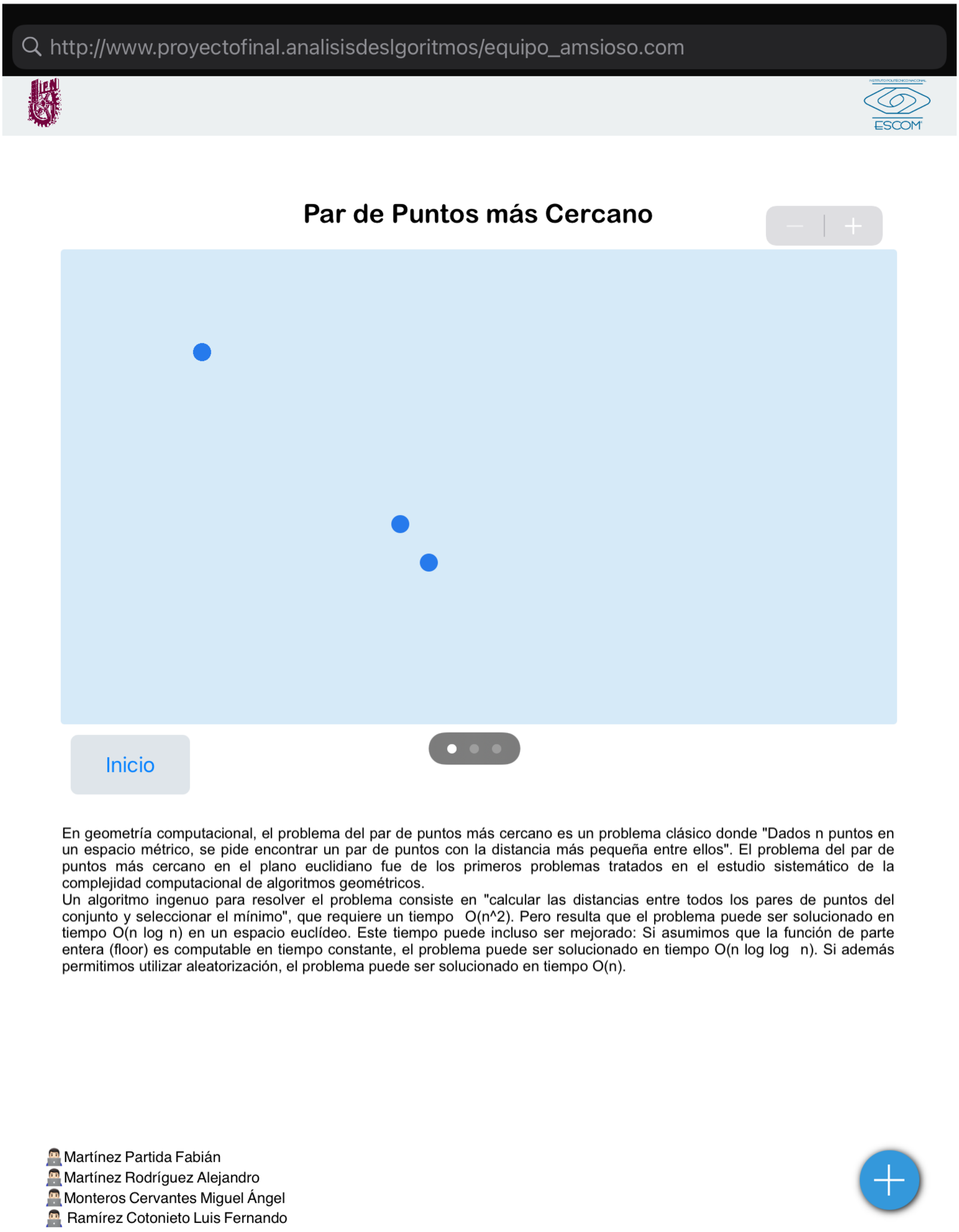

Figura 2: Pantalla Inicial

3.2.3. Inicio del Análisis

El algoritmo comienza a trabajar y analizar los puntos donde es colocado.

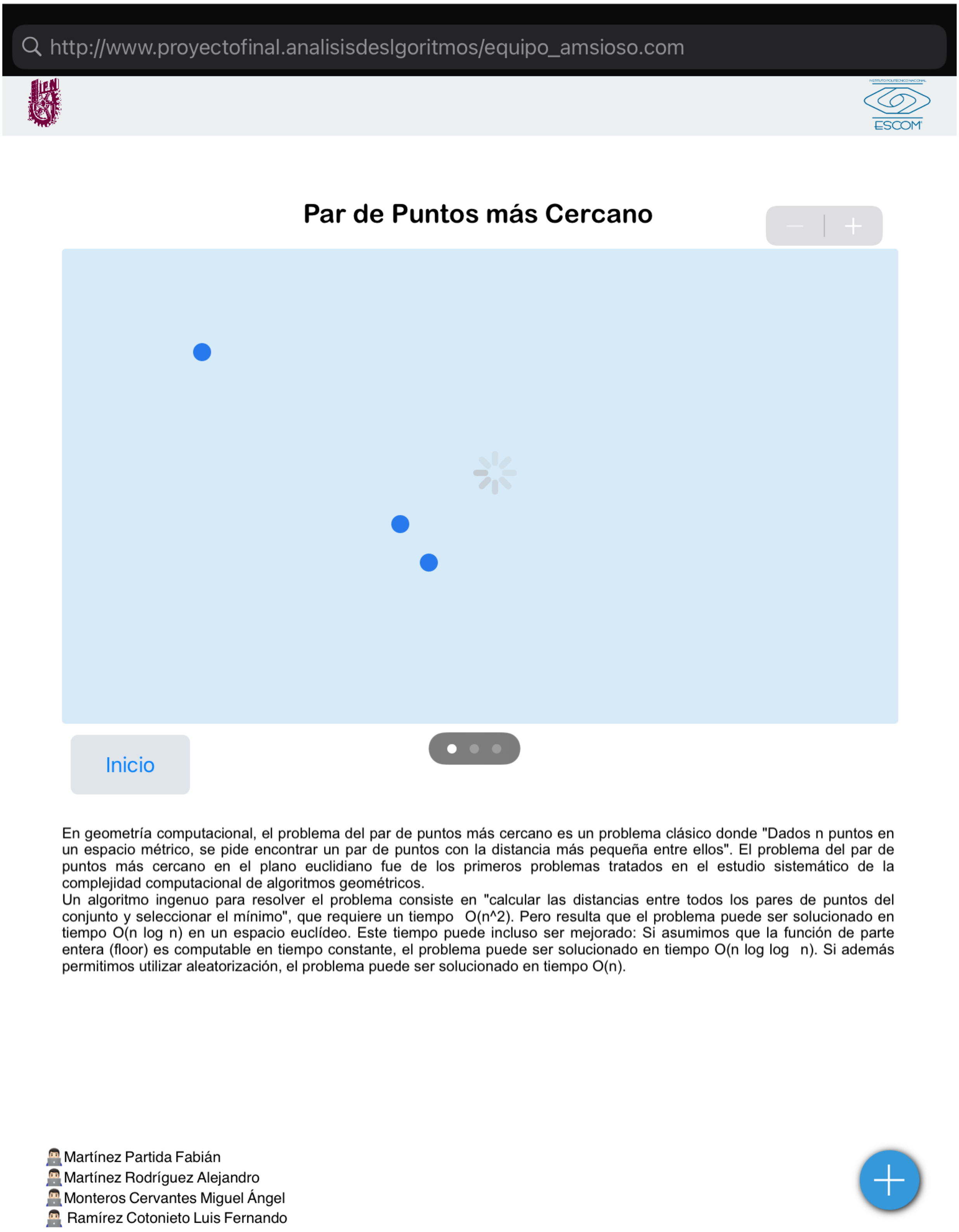

Figura 3: Inicio del análisis

3.2.4. Proceso del Análisis

La líneas que une los puntos empiezan a dibujarse y obtenemos un resultado final.

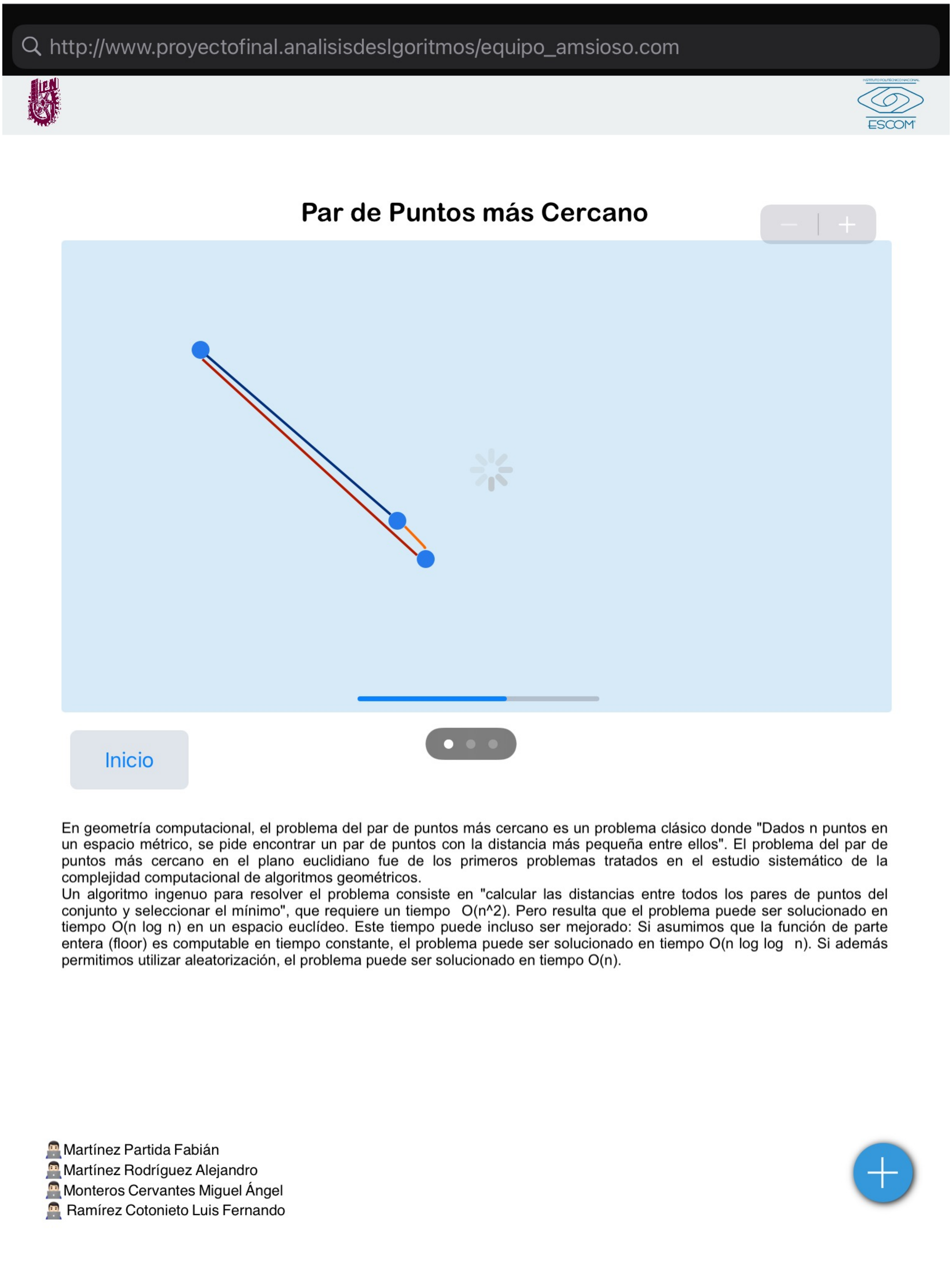

Figura 4: Proceso del análisis

4. Pruebas

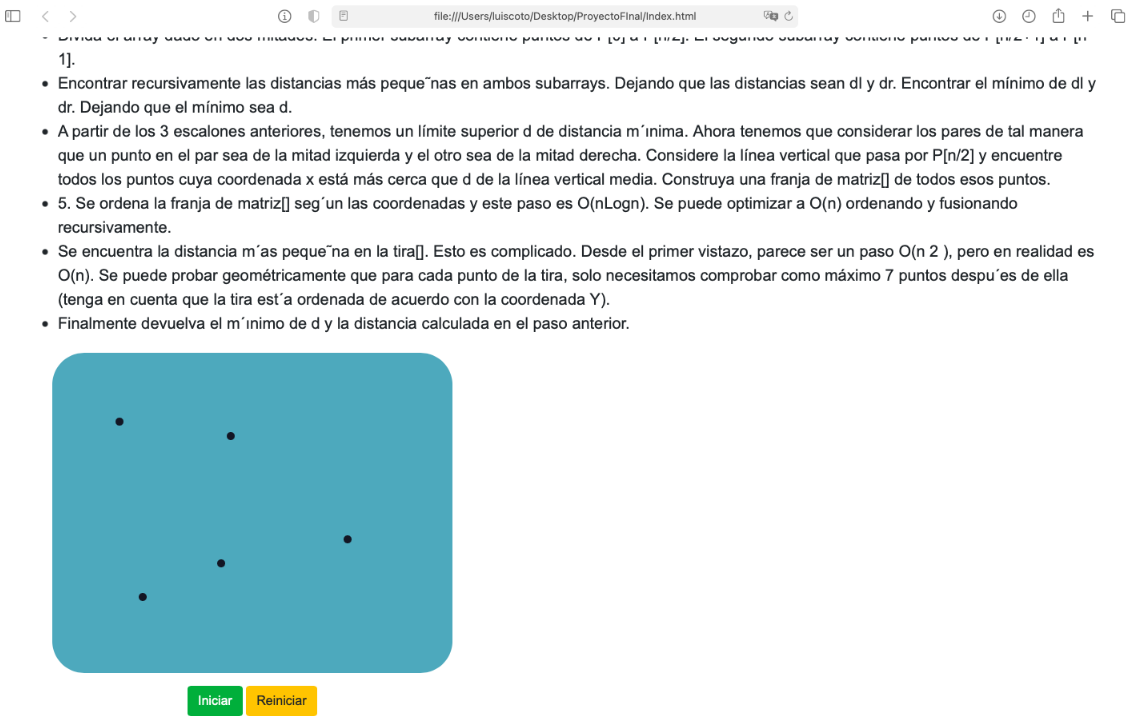

Figura 5: Pagina Inicial

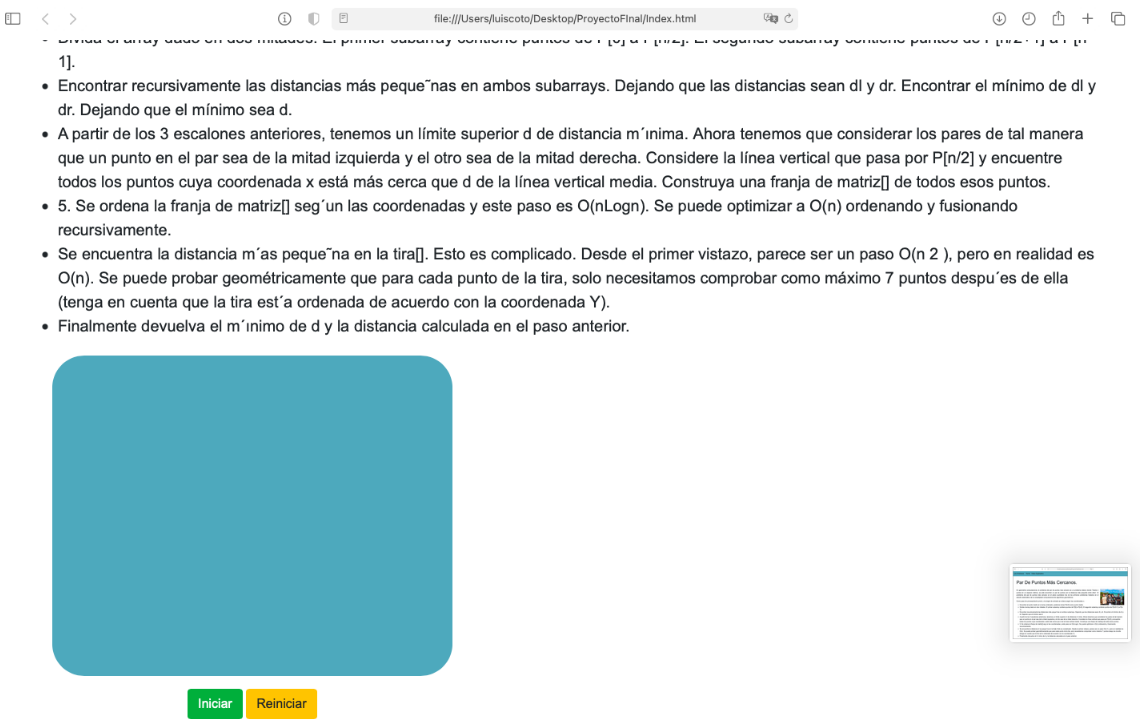

Figura 6: Pagina Inicial

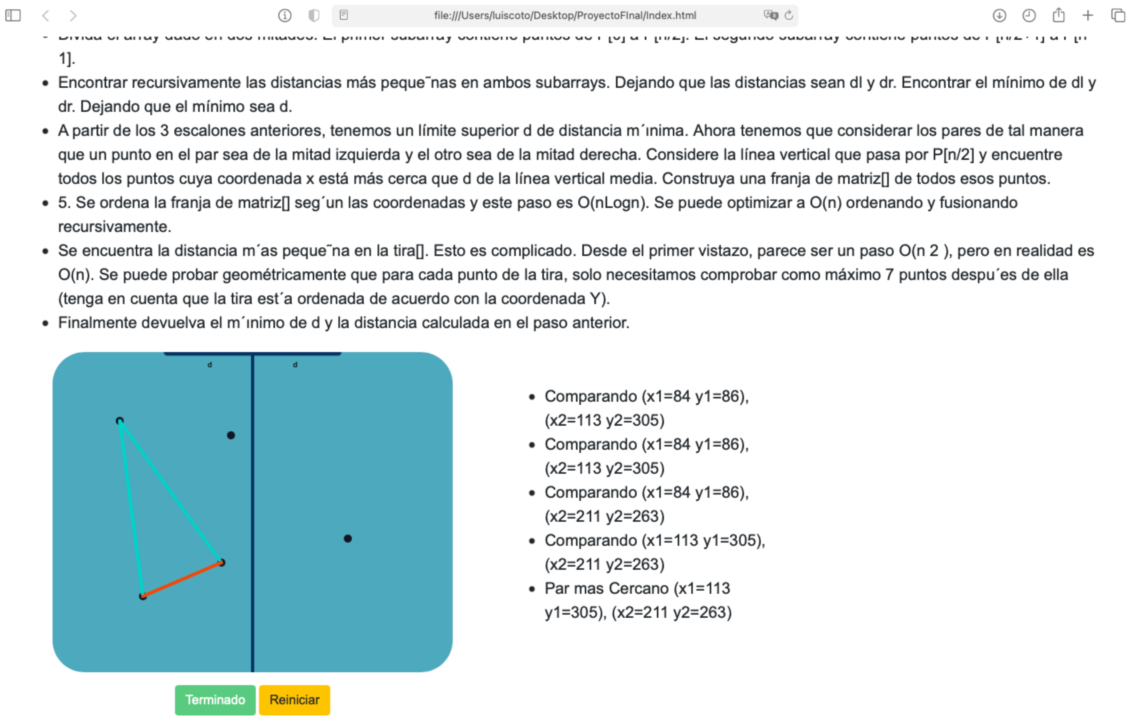

Figura 7: Funcionamiento del algoritmo

5. Anexos

5.1. main.js

```
function Point(x, y, g) {
    this.x = x || 0;
    this.y = y || 0;
    this.graphic = g;
    this.main_line = null;
};

Point.prototype.x = null;
Point.prototype.y = null;
Point.prototype.graphic = null;
Point.prototype.main_line = null;

window.onload = function () {
    Raphael.fn.line = function (startX, startY, endX, endY) {
        return this.path('M' + startX + ' ' + startY + ' L' + endX + ' ' +
            endY);
    };

    var paper = Raphael("canvas", 500, 400);
    var clearbtn = $('#clearbtn');
    var randombtn = $('#randombtn');
    var runbtn = $('#runbtn');
    var points = [];
    var canvas = null;
    var locked = false;
    var convexH = null;
    var auxtimer = null;

    function lineAnim(p, q) {
        line = paper.line(p.x, p.y, p.x, p.y).attr({
            'stroke-linecap': 'round',
```

```
            'stroke-linejoin': 'round',
            'stroke': '#23cec5'
        });
        p.main_line = line.animate({
            'stroke-width': '4',
            'path': 'M' + p.x + ' ' + p.y + ' L' + q.x + ' ' + q.y
        }, 1000);
    }

    function lineAnimFin(p, q) {
        line = paper.line(p.x, p.y, p.x, p.y).attr({
            'stroke-linecap': 'round',
            'stroke-linejoin': 'round',
            'stroke': '#FF5733'
        });
        p.main_line = line.animate({
            'stroke-width': '4',
            'path': 'M' + p.x + ' ' + p.y + ' L' + q.x + ' ' + q.y
        }, 1000);
    }


    function updateColorAnim(p) {
        c = paper.circle(p.x, p.y, 1).animate({
            r: 10,
            fill: '#131723',
            "stroke-width": 0
        }, 2000);
    }

    function deleteLineAnim(p) {
        p.main_line.animate({
            'stroke': 'rgba(255, 0, 0, 0.69)',
        }, 4000);
        p.main_line.animate({
            //'stroke': 'rgba(255, 0, 0, 0.69)',
            'stroke-width': '0'
        }, 1000);
    }


    var auxF = {
        lineAnim: lineAnim,
        lineAnimFin: lineAnimFin,
        deleteLineAnim: deleteLineAnim,
        middleLine: middleLine,
        drLine: drLine,
        dlLine: dlLine,
    };

    function getMousePos(e) {
        var totalOffsetX = 0;
        var totalOffsetY = 0;
        var canvasX = 0;
        var canvasY = 0;
        var currentElement = document.getElementById('canvas');

        do {
            totalOffsetX += currentElement.offsetLeft - currentElement.
                scrollLeft;
```

```
            totalOffsetY += currentElement.offsetTop - currentElement.
                scrollTop;
        }
        while (currentElement = currentElement.offsetParent);

        canvasX = e.pageX - totalOffsetX - document.body.scrollLeft;
        canvasY = e.pageY - totalOffsetY - document.body.scrollTop;

        return new Point(canvasX, canvasY, null, null);
    }

    function addPointAnim(p) {
        if (locked) {
            return;
        }

        if (points.length <= 9) {
            c = paper.circle(p.x, p.y, 1).animate({
                r: 5,
                fill: '#131723',
                "stroke-width": 0
            }, 200);
            p.graphic = c;
            points.push(p);
        } else {
            alert("No se pueden poner mas de 10 puntos")
        }
    }

    function clear() {
        paper.clear();
        canvas = paper.rect(0, 0, 500, 400, 40).attr({
            fill: '#62a8ba',
            stroke: "none"
        });
        points = [];
        unlock();
        running = false;
        convexH = null;
        runbtn.text('Iniciar');
        runbtn.attr('disabled', false);
        $("#group").empty();
        canvas.mouseup(function (e) {
            p = getMousePos(e);
            addPointAnim(p);
        });
    }

    function lock() {
        locked = true;
        if (convexH == null) {
            convexH = new CHAlgorith(points, auxF);
        }
    }

    function unlock() {
        locked = false;
        randombtn.attr('disabled', false);
    }

    function middleLine() {
```

```
        line = paper.line(250, 0, 250, 400).attr({
            'stroke-linecap': 'round',
            'stroke-linejoin': 'round',
            'stroke': '#0F365F'
        });
        p.main_line = line.animate({
            'stroke-width': '4',
            'path': 'M' + 250 + ' ' + 0 + ' L' + 250 + ' ' + 400
        }, 1000);
    }

    function drLine(d) {

        x2 = 250 + d;
        text = 250 + d / 2

        line = paper.line(250, 0, x2, 0).attr({
            'stroke-linecap': 'round',
            'stroke-linejoin': 'round',
            'stroke': '#0F365F'
        });
        p.main_line = line.animate({
            'stroke-width': '9',
            'path': 'M' + 250 + ' ' + 0 + ' L' + x2 + ' ' + 0
        }, 1000);

        paper.text(text, 15, "d")
    }

    function dlLine(d) {
        x2 = 250 - d;
        text = 250 - d / 2

        line = paper.line(250, 0, x2, 0).attr({
            'stroke-linecap': 'round',
            'stroke-linejoin': 'round',
            'stroke': '#0F365F'
        });
        line.animate({
            'stroke-width': '9',
            'path': 'M' + 250 + ' ' + 0 + ' L' + x2 + ' ' + 0
        }, 1000);

        paper.text(text, 15, "d")
    }

    function StartPause() {
        if (running) {
            running = false;
            window.clearInterval(auxtimer);
            runbtn.text('Continuar');
            runbtn.attr('disabled', false);
            clearbtn.attr('disabled', false);
        } else {
            running = true;
            lock();
            if (convexH == null) {
                convexH = new CHAlgorith(points, auxF);
            }
            runbtn.text('Pausa');
            clearbtn.attr('disabled', true);
```

```
            auxtimer = window.setInterval(function () {
                r = convexH.iterate();
                if (!r) {
                    window.clearInterval(auxtimer);
                    runbtn.text('Terminado');
                    runbtn.attr('disabled', true);
                    clearbtn.attr('disabled', false);
                }
            }, 800);
        }
    }

    clearbtn.click(function () {
        clear();
    });

    runbtn.click(function () {
        StartPause();
    });

    //Da click en el boton clear para lanzar la funcion clear()
    clearbtn.click();
};


function CHAlgorith(points, auxF) {

    this.States = {
        SORTING: 's',
        DIVIDE: 'dv',
        DONE: 'd'
    };


    this.points = points;
    this.auxF = auxF;
    this.state = this.States.SORTING;
    this.p = this.q = this.r = null;
    this.min = 0;
    this.d = 0;
    this.p1 = 0;
    this.p2 = 0;
    this.pares = [];

    this.arraySort = function () {
        this.points = this.points.sort(function (a, b) {
            if (a.x - b.x == 0) {
                return b.y - a.y;
            }
            return a.x - b.x;
        });
    }

    const dist = (p1, p2) => {
        return Math.sqrt((p1.x - p2.x) * (p1.x - p2.x) +
            (p1.y - p2.y) * (p1.y - p2.y)
        );
    }



```

```

    const bruteForce = (puntos, n) => {
        this.min = 10000;
        par = [];
        for (let i = 0; i < n; ++i) {
            for (let j = i + 1; j < n; ++j) {
                if (dist(puntos[i], puntos[j]) < this.min) {
                    this.min = dist(puntos[i], puntos[j])
                    this.p1 = puntos[i];
                    this.p2 = puntos[j];
                    par.push(puntos[i]);
                    par.push(puntos[j]);
                    this.pares.push(par);
                    this.par = [];
                    this.d = this.min;
                    this.auxF.lineAnim(this.p1, this.p2);
                    this.addIns("Comparando (x1=" + this.p1.x + " y1=" +
                        this.p1.y + "), (x2=" + this.p2.x + " y2=" + this.p2.
                        y + ")")
                    //this.auxF.deleteLineAnim(this.p1);


                }
            }
        }
        console.log(this.pares)
        return this.min;
    }


    this.closestUtil = (puntos, n) => {


        if (n == 1) {

            return alert("Debes coloar al menos 2 Puntos.")
        }

        if (n <= 3) return bruteForce(puntos, n);

        mid = Math.floor(n / 2);
        midPoint = puntos[mid];



        let dl = this.closestUtil(puntos, mid);
        let dr = this.closestUtil(puntos, n - mid)

        this.d = Math.min(dl, dr);


        let strip = [];
        let j = 0;

        for (const element of puntos) {
            if (Math.abs(element.x - midPoint.x) < this.d) {
                strip[j] = element;
                j++;
            }
        }
        const res = Math.min(this.d, stripClosest(strip, j, this.d));
```

```
        return res;

    }

    const stripClosest = (strip, size, d) => {
        let min = d;
        par = [];

        //sort
        this.points = strip;
        this.arraySort();
        strip = this.points;
        for (let i = 0; i < size; ++i) {
            for (let j = i + 1; j < size && (strip[j].y - strip[i].y) < min;
                ++j) {
                if (dist(strip[i], strip[j]) < min) {
                    min = dist(strip[i], strip[j])
                    this.p1 = (strip[i]);
                    this.p2 = (strip[j]);
                    par.push(strip[i]);
                    par.push(strip[j]);
                    this.pares.push(par);
                    this.par = [];
                    this.auxF.lineAnim(this.p1, this.p2);
                    this.addIns("Comparando (x1=" + this.p1.x + " y1=" +
                        this.p1.y + "), (x2=" + this.p2.x + " y2=" + this.p2.
                        y + ")")
                }
            }
        }
        return min;
    }
    this.addIns = function (text) {
        var ul = document.getElementById("group");
        var li = document.createElement('li');
        li.appendChild(document.createTextNode(text));
        ul.appendChild(li);
    }

    this.iterate = function () {
        switch (this.state) {
            case this.States.DONE:
                return false;

            case this.States.SORTING:

                this.arraySort();
                this.state = this.States.DIVIDE;

                return this.iterate();

            case this.States.DIVIDE:
                auxF.middleLine();
                this.closestUtil(points, points.length)
                this.auxF.drLine(this.d)
                this.auxF.dlLine(this.d)
                this.auxF.lineAnimFin(this.p1, this.p2)
                this.addIns("Par mas Cercano (x1=" + this.p1.x + " y1=" +
                    this.p1.y + "), (x2=" + this.p2.x + " y2=" + this.p2.y +
                    ")")

```

```
                    this.state = this.States.DONE;
                    return this.iterate();

          }
          return true;
     }
}
```

5.2. index.html

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
        integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
    <link rel="stylesheet" href="style.css">
    <title>Par de Puntos mas Cercanos</title>
</head>

<body>

    <!-- navbar -->
    <nav class="navbar navbar-expand-lg navbar-light" style="background-color: #62a8ba;">

        <a class="navbar-brand" href="#">Los Amsiosos</a>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent"
            aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>

        <div class="collapse navbar-collapse" id="navbarSupportedContent">
            <ul class="navbar-nav mr-auto">
                <li class="nav-item active">
                    <a class="nav-link" href="Index.html">Home <span class="sr-only">(current)</span></a>
                </li>
                <li class="nav-item active">
                    <a class="nav-link" href="#">Video Explicativo</a>
                </li>
            </ul>
        </div>
    </nav>

    <!-- Contenido Principal -->
    <div class="container-fluid">
        <div class="jumbotron" style="background-color: white; margin-top: -20px;">
            <h1 class="display-4">Par De Puntos M s Cercanos.</h1>
            <hr class="my-4">
            <img src="img.png" id="equipo" alt="">
            <p>En geometr a computacional, el problema del par de puntos mas cercano es un problema
                cl sico donde  Dados  n
                puntos en un espacio m trico, se pide encontrar un par de puntos con la distancia m s peque a entre
                e l l o s . El
                problema del par de puntos m s cercano en el plano euclidiano fue de los primeros problemas tratados en
                el
```

```
                estudio sistem tico de la complejidad computacional de
                    algoritmos geom tricos.</p>
            <p>
                Como paso de procesamiento previo, el arreglo de entrada se
                    ordena seg n las coordenadas x.
            <ul>
                <li>Encontrar el punto medio en el array ordenado, podemos
                    tomar P[n/2] como punto medio.</li>
                <li>Divida el array dado en dos mitades. El primer subarray
                    contiene puntos de P[0] a P[n/2]. El segundo
                    subarray contiene puntos de P[n/2+1] a P[n-1].</li>
                <li>Encontrar recursivamente las distancias m s peque nas
                    en ambos subarrays. Dejando que las
                    distancias
                    sean dl y dr. Encontrar el m nimo de dl y dr. Dejando
                        que el m nimo sea d.</li>
                <li>A partir de los 3 escalones anteriores, tenemos un
                    l mite superior d de distancia m   nima. Ahora
                    tenemos
                    que considerar los pares de tal manera que un punto en
                        el par sea de la mitad izquierda y el otro
                    sea
                    de la mitad derecha. Considere la l nea vertical que
                        pasa por P[n/2] y encuentre todos los puntos
                    cuya
                    coordenada x est   m s cerca que d de la l nea
                        vertical media. Construya una franja de matriz[] de
                    todos
                    esos puntos.</li>
                <li>
                    5. Se ordena la franja de matriz[] seg un las
                        coordenadas y este paso es O(nLogn). Se puede
                    optimizar a
                    O(n)
                    ordenando y fusionando recursivamente.</li>
                <li>Se encuentra la distancia m as peque na en la tira[].
                    Esto es complicado. Desde el primer vistazo,
                    parece ser
                    un paso O(n
                    2
                    ), pero en realidad es O(n). Se puede probar
                        geom tricamente que para cada punto de la
                    tira, solo necesitamos comprobar como m ximo 7 puntos
                        despu es de ella (tenga en cuenta que la tira
                    est a
                    ordenada de acuerdo con la coordenada Y).</li>
                <li>Finalmente devuelva el m   nimo de d y la distancia
                    calculada en el paso anterior.</li>
            </ul>
            </p>


        </div>

        <div class="row">
            <div class="col-lg-6"">
                <div class=" container-fluid id="main">
                <div id="canvas">
                    <div class="container-fluid" id="botones">
                        <button id="runbtn" class="btn btn-success mt-3"
                            type="button">Iniciar</button>
```

```
                        <button id="clearbtn" class="btn btn-warning mt-3"
                            type="button">Reiniciar</button>
                    </div>
                </div>
            </div>
        </div>
        <div class="col-lg-3" id="list">
            <ul id="group">
            </ul>
        </div>
    </div>

    </div>
    <script src="https://code.jquery.com/jquery-3.6.0.slim.js"
        integrity="sha256-HwWONEZrpuoh951cQD1ov2HUK5zA5DwJ1DNUXaM6FsY="
            crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/
       bootstrap.min.js"
        integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/
            nJGzIxFDsf4x0xIM+B07jRM"
        crossorigin="anonymous"></script>
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
        integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH
            +8abtTE1Pi6jizo"
        crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd
       /popper.min.js"
        integrity="sha384-
            UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
        crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/raphael/2.2.7/
       raphael.js"></script>
    <script src="main.js"></script>
    <script type module src="points.js"></script>
</body>

</html>
```

5.3. style.css

```
body{
  background-color: #fff;
  font-family:Quicksand,Arial;
  }

    h3,h2{
      text-align: center;
      font-family:Quicksand,Arial;
    }
    #leftPanel{

      display: inline-block;
      vertical-align: top;
      margin-left: -100px;
      margin-right: auto;
      height: 150px;
    }
    p{
      text-align: justify;
      font-family:Quicksand,Arial;
      font-size: 20px;

    }

    #canvas {
        height: 400px;
        width: 500px;
        cursor: crosshair;
        margin-top: -90px;
        margin-left: 50px;
    }

    #body button {
        font-size: 1.2em;
        width: 120px;
    }
    #botones{
      text-align: center;
    }
    #p4,#p8,#p6,#p10{
      margin-left: 20px;
    }
    #p5,#p9{
      margin-left: 40px;
    }
    #currentState{
      font-size: large;
    }
    #list{
      margin-left: -80px;
      margin-top: -50px;
    }
     marquee{
       width: 250px;
       height: 100px;
       margin-top: 20px;
     }
     #equipo{
       width: 300px;
```

```
        height: 200px;
        float: right;
        margin-left: 20px;

      }
      #main{
        height: 600px;
        width: 600px;
      }
      li{
        font-size: 20px;
      }
```

6. Bibliografía

- POSNER, Eric A.; SPIER, Kathryn E.; VERMEULE, Adrian. Divide and conquer. Journal of Legal Analysis, 2010, vol. 2, no 2, p. 417-471.
- CARO, Miguel Rodrigo Pincheira. Busqueda del par de puntos mas cercanos sobre conjuntos no indexados. 2012.
- CHANDÍA, Nahuel; FERNANDO, Álvaro. Implementación de un sistema web y móvil para obtener los K-Pares de vecinos más cercanos entre dos conjuntos de puntos espaciales representados en la estructura de datos compacta k2-Tree. 2017.
- PASCUAL, D.; PLA, F.; SÁNCHEZ, S. Algoritmos de agrupamiento. Método Informáticos Avanzados, 2007, p. 164-174.