



# 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

<b>1. Planteamiento del problema</b>	<b>3</b>
<b>2. Planteamiento del Algoritmo y Solución</b>	<b>4</b>
2.1. Tecnologías utilizadas . . . . .	4
2.2. Divide y Conquista . . . . .	4
<b>3. Representación del Algoritmo</b>	<b>5</b>
3.1. Tecnología utilizada . . . . .	5
3.2. Makups . . . . .	5
3.2.1. Pantalla de Inicio . . . . .	5
3.2.2. Colocación de Puntos . . . . .	6
3.2.3. Inicio del Análisis . . . . .	7
3.2.4. Proceso del Análisis . . . . .	8
<b>4. Pruebas</b>	<b>9</b>
<b>5. Anexos</b>	<b>10</b>
5.1. main.js . . . . .	10
5.2. index.html . . . . .	18
5.3. style.css . . . . .	21
<b>6. Bibliografía</b>	<b>23</b>

# 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 calcular 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, Raphael.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  $O(n \log n)$  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  $d_l$  y  $d_r$ . Encontrar el mínimo de  $d_l$  y  $d_r$ . 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  $O(n \log n)$ . 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

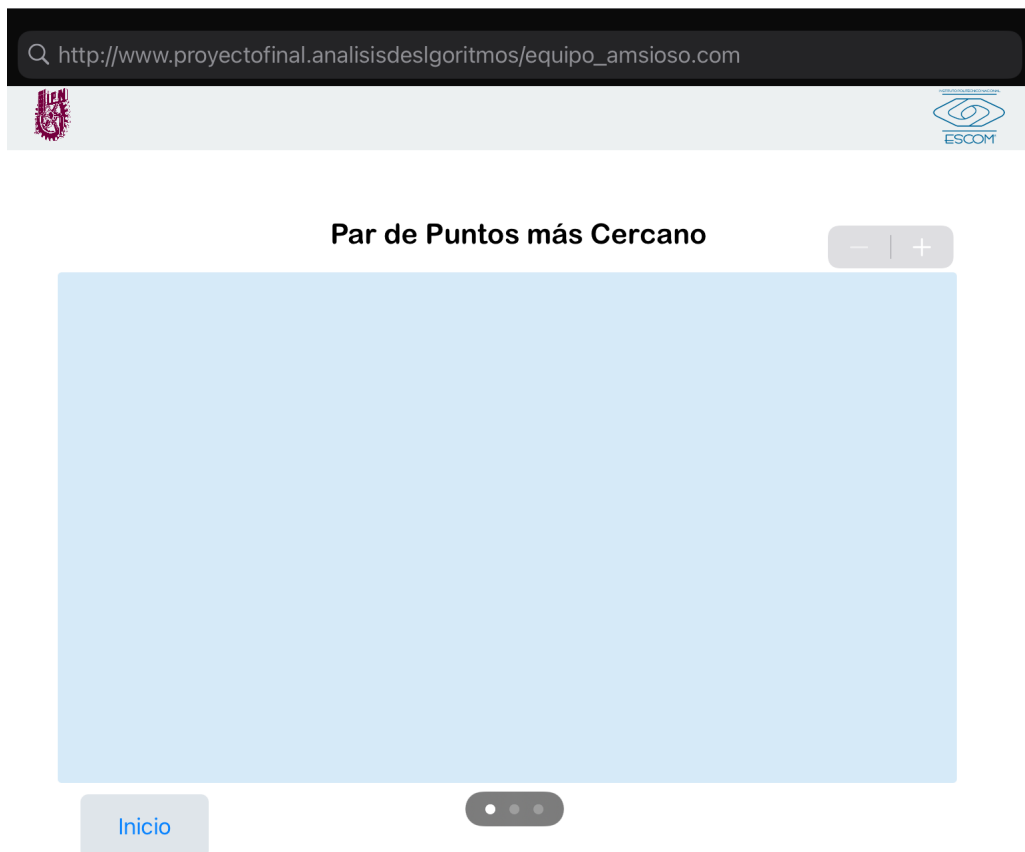
Adobe XD es un editor de gráficos vectoriales desarrollado y publicado por Adobe Inc para diseñar y crear un prototipo de la experiencia del usuario para páginas web y aplicaciones móviles. El software está disponible para MacOS y Windows. Adobe XD apoya a los diseño vectoriales y a los sitios web wireframe,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



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 "calcular 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  $O(n)$ .

Martínez Partida Fabián  
 Martínez Rodríguez Alejandro  
 Monteros Cervantes Miguel Ángel  
 Ramírez Cotonieto Luis Fernando



Figura 1: Pantalla Inicial

### 3.2.2. Colocación de Puntos

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

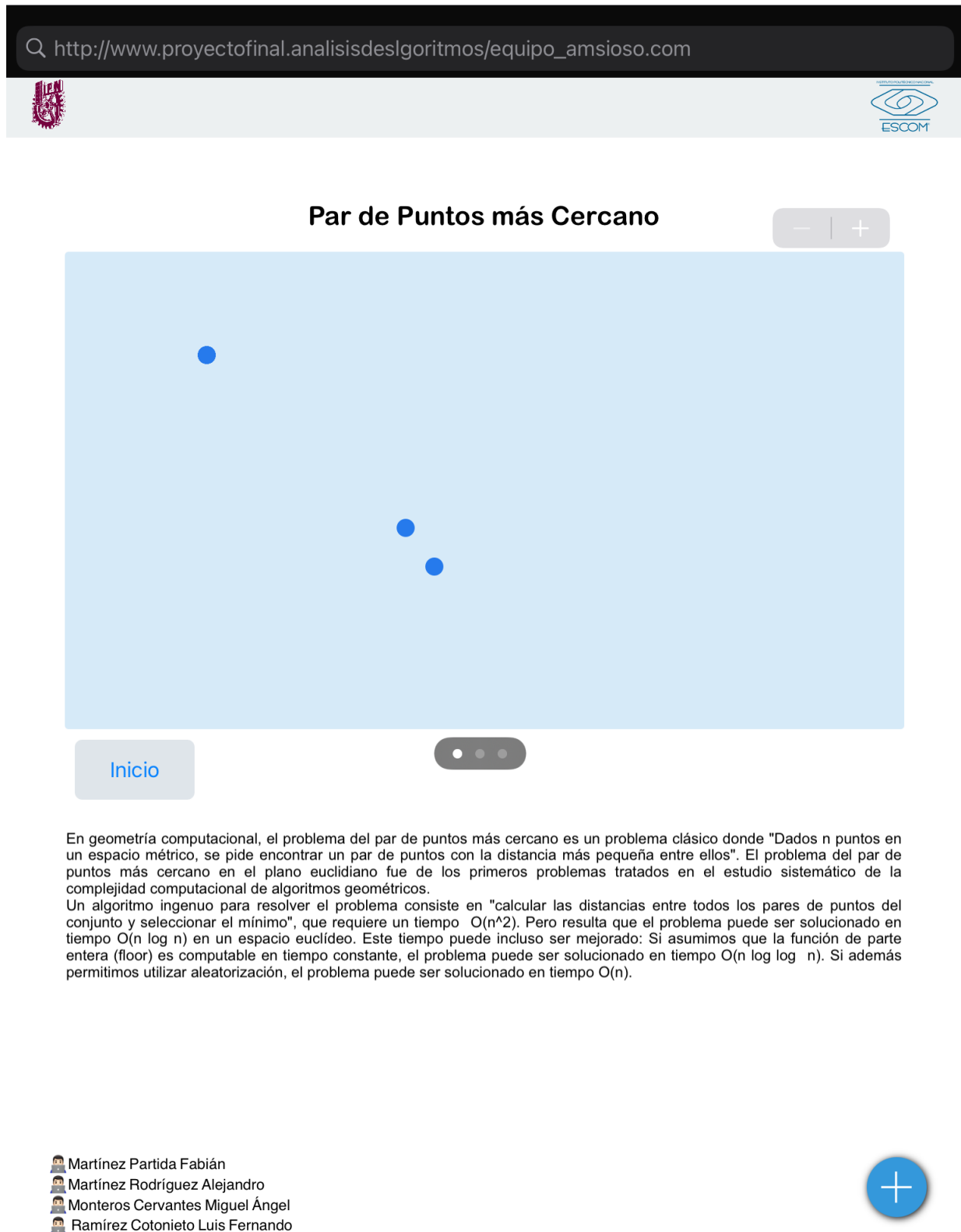




Figura 2: Pantalla Inicial

### 3.2.3. Inicio del Análisis

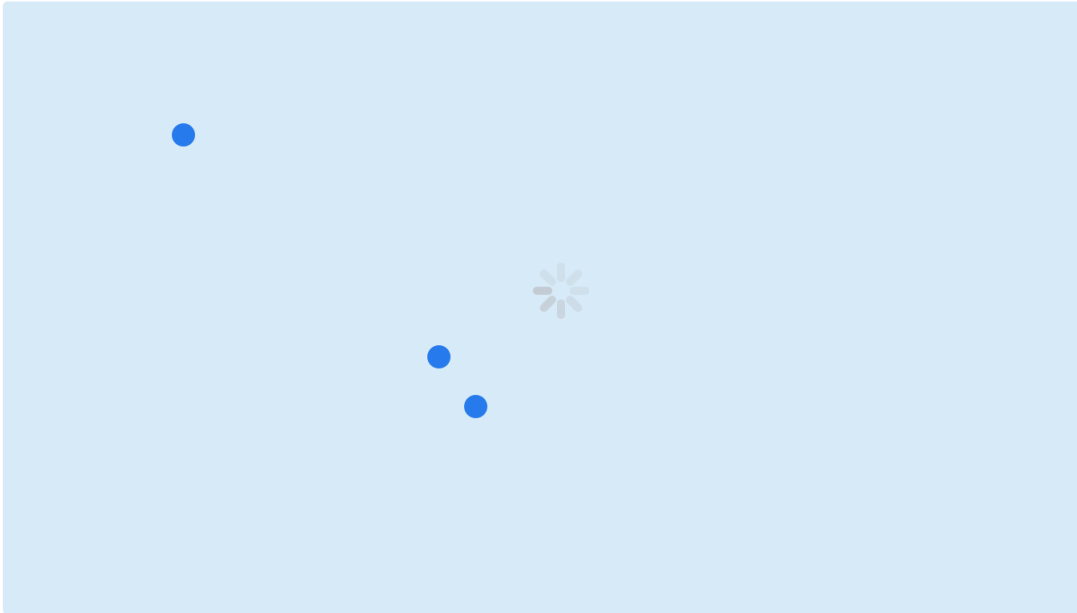
El algoritmo comienza a trabajar y analizar los puntos donde es colocado.

Q [http://www.proyectofinal.analisisdeslgoritmos/equipo\\_amsioso.com](http://www.proyectofinal.analisisdeslgoritmos/equipo_amsioso.com)



## Par de Puntos más Cercano





- | +



Inicio

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 "calcular 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  $O(n)$ .

 Martínez Partida Fabián  
 Martínez Rodríguez Alejandro  
 Monteros Cervantes Miguel Ángel  
 Ramírez Cotonierto Luis Fernando


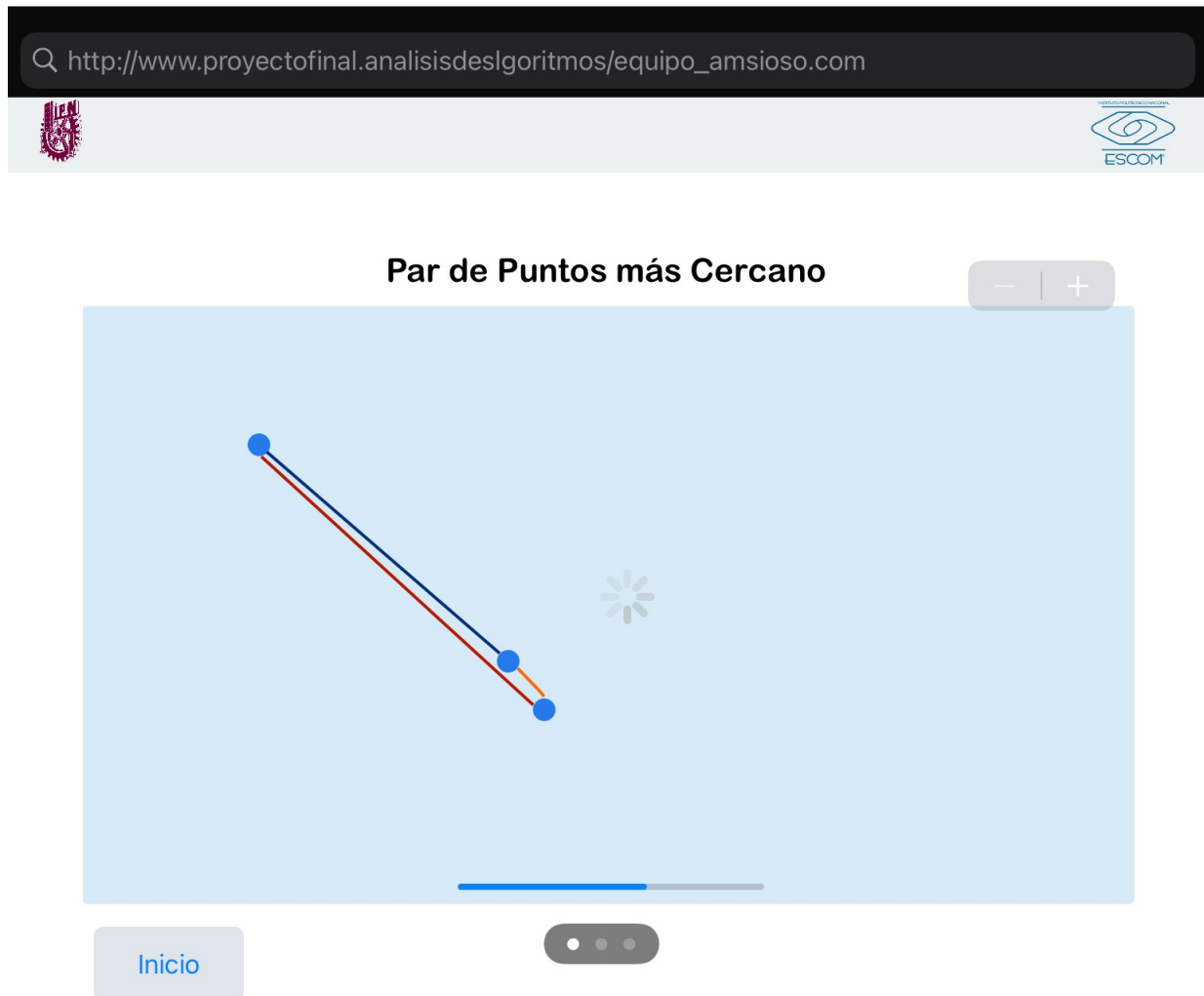


Figura 3: Inicio del análisis



### 3.2.4. Proceso del Análisis

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



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 "calcular 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  $O(n)$ .





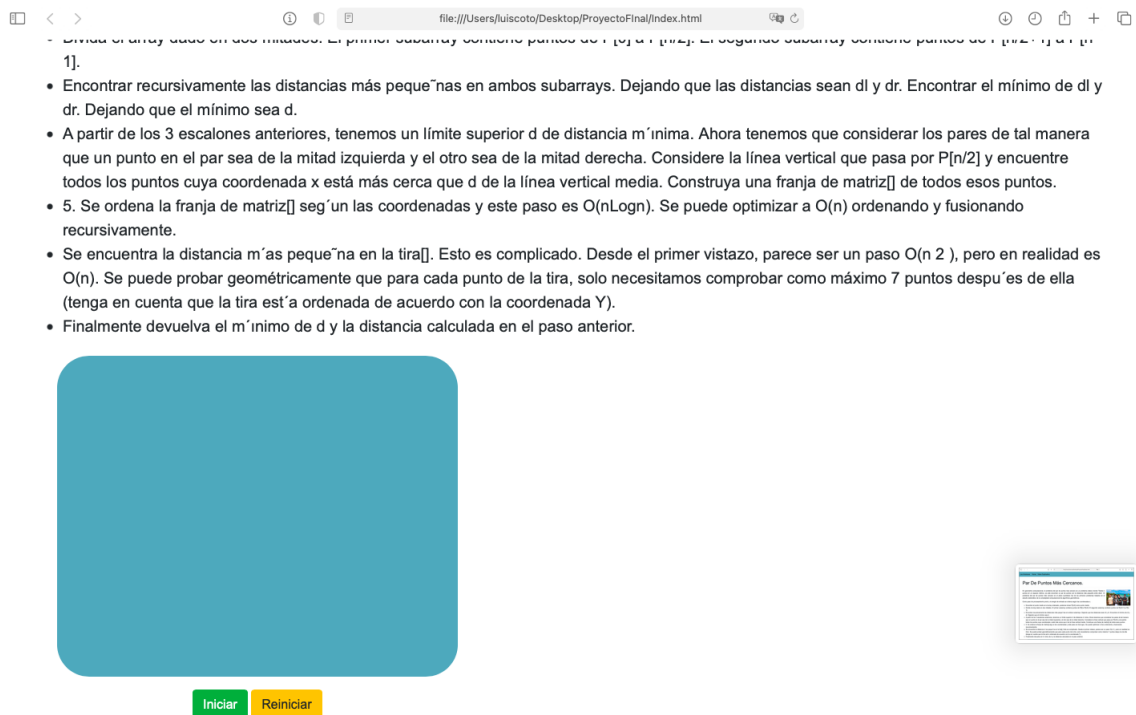
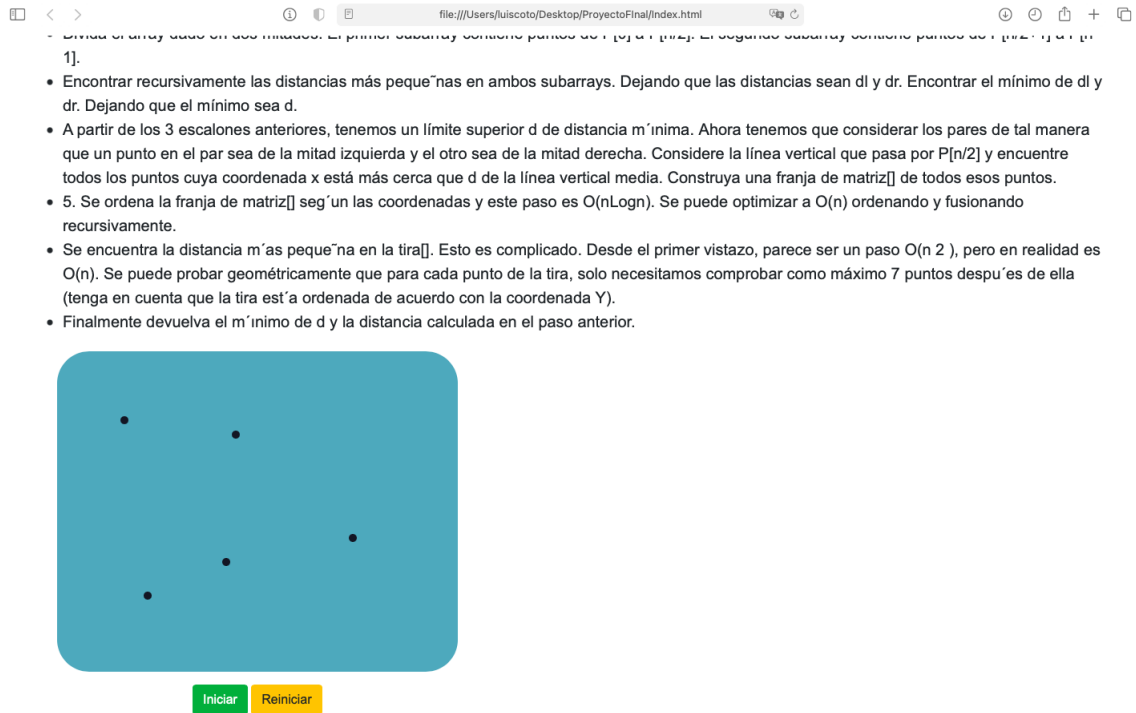
 Martínez Partida Fabián  
 Martínez Rodríguez Alejandro  
 Monteros Cervantes Miguel Ángel  
 Ramírez Cotonieto Luis Fernando



Figura 4: Proceso del análisis



## 4. Pruebas



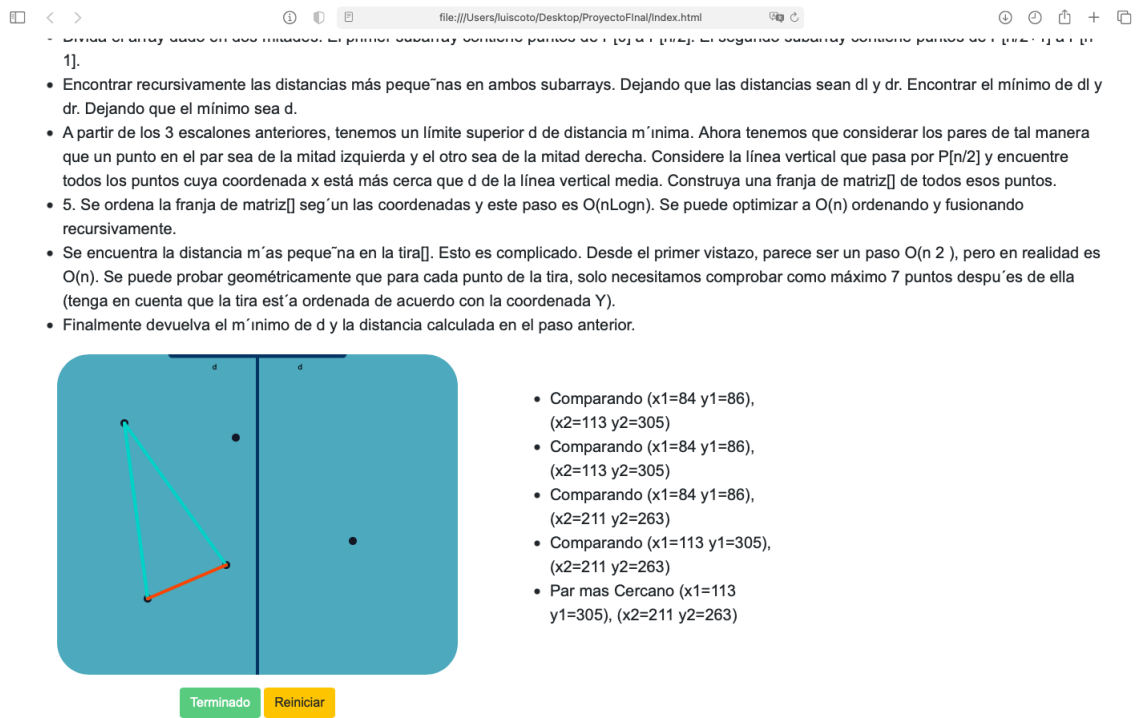


Figura 7: Funcionamiento del algoritmo

## 5. Anexos

### 5.1. main.js

```

1 function Point(x, y, g) {
2     this.x = x || 0;
3     this.y = y || 0;
4     this.graphic = g;
5     this.main_line = null;
6 };
7
8 Point.prototype.x = null;
9 Point.prototype.y = null;
10 Point.prototype.graphic = null;
11 Point.prototype.main_line = null;
12
13 window.onload = function () {
14     Raphael.fn.line = function (startX, startY, endX, endY) {
15         return this.path('M' + startX + ' ' + startY + ' L' + endX + ' ' +
16             endY);
17     };
18
19     var paper = Raphael("canvas", 500, 400);
20     var clearbtn = $('#clearbtn');
21     var randombtn = $('#randombtn');
22     var runbtn = $('#runbtn');
23     var points = [];
24     var canvas = null;
25     var locked = false;
26     var convexH = null;
27     var auxtimer = null;
28
29     function lineAnim(p, q) {
30         line = paper.line(p.x, p.y, p.x, p.y).attr({
31             'stroke-linecap': 'round',

```

```

31         'stroke-linejoin': 'round',
32         'stroke': '#23cec5'
33     });
34     p.main_line = line.animate({
35         'stroke-width': '4',
36         'path': 'M' + p.x + ' ' + p.y + ' L' + q.x + ' ' + q.y
37     }, 1000);
38 }
39
40 function lineAnimFin(p, q) {
41     line = paper.line(p.x, p.y, p.x, p.y).attr({
42         'stroke-linecap': 'round',
43         'stroke-linejoin': 'round',
44         'stroke': '#FF5733'
45     });
46     p.main_line = line.animate({
47         'stroke-width': '4',
48         'path': 'M' + p.x + ' ' + p.y + ' L' + q.x + ' ' + q.y
49     }, 1000);
50 }
51
52
53 function updateColorAnim(p) {
54     c = paper.circle(p.x, p.y, 1).animate({
55         r: 10,
56         fill: '#131723',
57         "stroke-width": 0
58     }, 2000);
59 }
60
61 function deleteLineAnim(p) {
62     p.main_line.animate({
63         'stroke': 'rgba(255, 0, 0, 0.69)',
64     }, 4000);
65     p.main_line.animate({
66         //'stroke': 'rgba(255, 0, 0, 0.69)',
67         'stroke-width': '0'
68     }, 1000);
69 }
70
71
72 var auxF = {
73     lineAnim: lineAnim,
74     lineAnimFin: lineAnimFin,
75     deleteLineAnim: deleteLineAnim,
76     middleLine: middleLine,
77     drLine: drLine,
78     dlLine: dlLine,
79 };
80
81 function getMousePos(e) {
82     var totalOffsetX = 0;
83     var totalOffsetY = 0;
84     var canvasX = 0;
85     var canvasY = 0;
86     var currentElement = document.getElementById('canvas');
87
88     do {
89         totalOffsetX += currentElement.offsetLeft - currentElement.
            scrollLeft;

```

```

90         totalOffsetY += currentElement.offsetTop - currentElement.
           scrollTop;
91     }
92     while (currentElement = currentElement.offsetParent);
93
94     canvasX = e.pageX - totalOffsetX - document.body.scrollLeft;
95     canvasY = e.pageY - totalOffsetY - document.body.scrollTop;
96
97     return new Point(canvasX, canvasY, null, null);
98 }
99
100 function addPointAnim(p) {
101     if (locked) {
102         return;
103     }
104
105     if (points.length <= 9) {
106         c = paper.circle(p.x, p.y, 1).animate({
107             r: 5,
108             fill: '#131723',
109             "stroke-width": 0
110         }, 200);
111         p.graphic = c;
112         points.push(p);
113     } else {
114         alert("No se pueden poner mas de 10 puntos")
115     }
116 }
117
118 function clear() {
119     paper.clear();
120     canvas = paper.rect(0, 0, 500, 400, 40).attr({
121         fill: '#62a8ba',
122         stroke: "none"
123     });
124     points = [];
125     unlock();
126     running = false;
127     convexH = null;
128     runbtn.text('Iniciar');
129     runbtn.attr('disabled', false);
130     $("#group").empty();
131     canvas.mouseup(function (e) {
132         p = getMousePos(e);
133         addPointAnim(p);
134     });
135 }
136
137 function lock() {
138     locked = true;
139     if (convexH == null) {
140         convexH = new CHAlgorith(points, auxF);
141     }
142 }
143
144 function unlock() {
145     locked = false;
146     randombtn.attr('disabled', false);
147 }
148
149 function middleLine() {

```

```

150     line = paper.line(250, 0, 250, 400).attr({
151         'stroke-linecap': 'round',
152         'stroke-linejoin': 'round',
153         'stroke': '#0F365F'
154     });
155     p.main_line = line.animate({
156         'stroke-width': '4',
157         'path': 'M' + 250 + ' ' + 0 + ' L' + 250 + ' ' + 400
158     }, 1000);
159 }
160
161 function drLine(d) {
162
163     x2 = 250 + d;
164     text = 250 + d / 2
165
166     line = paper.line(250, 0, x2, 0).attr({
167         'stroke-linecap': 'round',
168         'stroke-linejoin': 'round',
169         'stroke': '#0F365F'
170     });
171     p.main_line = line.animate({
172         'stroke-width': '9',
173         'path': 'M' + 250 + ' ' + 0 + ' L' + x2 + ' ' + 0
174     }, 1000);
175
176     paper.text(text, 15, "d")
177 }
178
179 function dlLine(d) {
180     x2 = 250 - d;
181     text = 250 - d / 2
182
183     line = paper.line(250, 0, x2, 0).attr({
184         'stroke-linecap': 'round',
185         'stroke-linejoin': 'round',
186         'stroke': '#0F365F'
187     });
188     line.animate({
189         'stroke-width': '9',
190         'path': 'M' + 250 + ' ' + 0 + ' L' + x2 + ' ' + 0
191     }, 1000);
192
193     paper.text(text, 15, "d")
194 }
195
196 function StartPause() {
197     if (running) {
198         running = false;
199         window.clearInterval(auxtimer);
200         runbtn.text('Continuar');
201         runbtn.attr('disabled', false);
202         clearbtn.attr('disabled', false);
203     } else {
204         running = true;
205         lock();
206         if (convexH == null) {
207             convexH = new CHAlgorith(points, auxF);
208         }
209         runbtn.text('Pausa');
210         clearbtn.attr('disabled', true);

```

```

211         auxtimer = window.setInterval(function () {
212             r = convexH.iterate();
213             if (!r) {
214                 window.clearInterval(auxtimer);
215                 runbtn.text('Terminado');
216                 runbtn.attr('disabled', true);
217                 clearbtn.attr('disabled', false);
218             }
219         }, 800);
220     }
221 }
222
223 clearbtn.click(function () {
224     clear();
225 });
226
227 runbtn.click(function () {
228     StartPause();
229 });
230
231 //Da click en el boton clear para lanzar la funcion clear()
232 clearbtn.click();
233 };
234
235
236 function CHAlgorith(points, auxF) {
237
238     this.States = {
239         SORTING: 's',
240         DIVIDE: 'dv',
241         DONE: 'd'
242     };
243
244
245     this.points = points;
246     this.auxF = auxF;
247     this.state = this.States.SORTING;
248     this.p = this.q = this.r = null;
249     this.min = 0;
250     this.d = 0;
251     this.p1 = 0;
252     this.p2 = 0;
253     this.pares = [];
254
255     this.arraySort = function () {
256         this.points = this.points.sort(function (a, b) {
257             if (a.x - b.x == 0) {
258                 return b.y - a.y;
259             }
260             return a.x - b.x;
261         });
262     }
263
264     const dist = (p1, p2) => {
265         return Math.sqrt((p1.x - p2.x) * (p1.x - p2.x) +
266             (p1.y - p2.y) * (p1.y - p2.y)
267         );
268     }
269
270
271

```

```

272
273     const bruteForce = (puntos, n) => {
274         this.min = 10000;
275         par = [];
276         for (let i = 0; i < n; ++i) {
277             for (let j = i + 1; j < n; ++j) {
278                 if (dist(puntos[i], puntos[j]) < this.min) {
279                     this.min = dist(puntos[i], puntos[j])
280                     this.p1 = puntos[i];
281                     this.p2 = puntos[j];
282                     par.push(puntos[i]);
283                     par.push(puntos[j]);
284                     this.pares.push(par);
285                     this.par = [];
286                     this.d = this.min;
287                     this.auxF.lineAnim(this.p1, this.p2);
288                     this.addIns("Comparando (x1=" + this.p1.x + " y1=" +
289                                 this.p1.y + " ), (x2=" + this.p2.x + " y2=" + this.p2.
290                                 y + ")")
291                     //this.auxF.deleteLineAnim(this.p1);
292                 }
293             }
294         }
295         console.log(this.pares)
296         return this.min;
297     }
298
299
300     this.closestUtil = (puntos, n) => {
301
302
303         if (n == 1) {
304
305             return alert("Debes colocar al menos 2 Puntos.")
306         }
307
308         if (n <= 3) return bruteForce(puntos, n);
309
310         mid = Math.floor(n / 2);
311         midPoint = puntos[mid];
312
313
314
315         let dl = this.closestUtil(puntos, mid);
316         let dr = this.closestUtil(puntos, n - mid)
317
318         this.d = Math.min(dl, dr);
319
320
321         let strip = [];
322         let j = 0;
323
324         for (const element of puntos) {
325             if (Math.abs(element.x - midPoint.x) < this.d) {
326                 strip[j] = element;
327                 j++;
328             }
329         }
330         const res = Math.min(this.d, stripClosest(strip, j, this.d));

```



```

331         return res;
332
333     }
334
335     const stripClosest = (strip, size, d) => {
336         let min = d;
337         par = [];
338
339         //sort
340         this.points = strip;
341         this.arraySort();
342         strip = this.points;
343         for (let i = 0; i < size; ++i) {
344             for (let j = i + 1; j < size && (strip[j].y - strip[i].y) < min;
345                 ++j) {
346                 if (dist(strip[i], strip[j]) < min) {
347                     min = dist(strip[i], strip[j]);
348                     this.p1 = (strip[i]);
349                     this.p2 = (strip[j]);
350                     par.push(strip[i]);
351                     par.push(strip[j]);
352                     this.pares.push(par);
353                     this.par = [];
354                     this.auxF.lineAnim(this.p1, this.p2);
355                     this.addIns("Comparando (x1=" + this.p1.x + " y1=" +
356                                 this.p1.y + ")", (x2=" + this.p2.x + " y2=" + this.p2.
357                                     y + ")")
358                 }
359             }
360             return min;
361         }
362
363         this.addIns = function (text) {
364             var ul = document.getElementById("group");
365             var li = document.createElement('li');
366             li.appendChild(document.createTextNode(text));
367             ul.appendChild(li);
368         }
369
370         this.iterate = function () {
371             switch (this.state) {
372                 case this.States.DONE:
373                     return false;
374
375                 case this.States.SORTING:
376
377                     this.arraySort();
378                     this.state = this.States.DIVIDE;
379
380                     return this.iterate();
381
382                 case this.States.DIVIDE:
383                     auxF.middleLine();
384                     this.closestUtil(points, points.length)
385                     this.auxF.drLine(this.d)
386                     this.auxF.dlLine(this.d)
387                     this.auxF.lineAnimFin(this.p1, this.p2)
388                     this.addIns("Par mas Cercano (x1=" + this.p1.x + " y1=" +
389                                 this.p1.y + ")", (x2=" + this.p2.x + " y2=" + this.p2.y +
390                                     ")")

```

```
387         this.state = this.States.DONE;
388         return this.iterate();
389     }
390     return true;
391 }
392 }
393 }
```

## 5.2. index.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5     <meta charset="UTF-8">
6     <meta http-equiv="X-UA-Compatible" content="IE=edge">
7     <meta name="viewport" content="width=device-width, initial-scale=1.0">
8     <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/
9         bootstrap/4.3.1/css/bootstrap.min.css"
10         integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/
11         iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
12     <link rel="stylesheet" href="style.css">
13     <title>Par de Puntos mas Cercanos</title>
14 </head>
15
16 <body>
17     <!-- navbar -->
18     <nav class="navbar navbar-expand-lg navbar-light" style="background-
19         color: #62a8ba;">
20
21         <a class="navbar-brand" href="#">Los Amsiosos</a>
22         <button class="navbar-toggler" type="button" data-toggle="collapse"
23             data-target="#navbarSupportedContent"
24             aria-controls="navbarSupportedContent" aria-expanded="false"
25             aria-label="Toggle navigation">
26             <span class="navbar-toggler-icon"></span>
27         </button>
28
29         <div class="collapse navbar-collapse" id="navbarSupportedContent">
30             <ul class="navbar-nav mr-auto">
31                 <li class="nav-item active">
32                     <a class="nav-link" href="Index.html">Home <span class="
33                         sr-only">(current)</span></a>
34                 </li>
35                 <li class="nav-item active">
36                     <a class="nav-link" href="#">Video Explicativo</a>
37                 </li>
38             </ul>
39         </div>
40     </nav>
41
42     <!-- Contenido Principal -->
43     <div class="container-fluid">
44         <div class="jumbotron" style="background-color: white; margin-top:
45             -20px;">
46             <h1 class="display-4">Par De Puntos M s Cercanos.</h1>
47             <hr class="my-4">
48             
49             <p>En geometr a computacional, el problema del par de puntos
50                 mas cercano es un problema
51                 cl sico donde Dados n
52                 puntos en un espacio m trico , se pide encontrar un par de
53                 puntos con la distancia m s peque a entre
54                 ellos . El
55                 problema del par de puntos m s cercano en el plano
56                 euclidiano fue de los primeros problemas tratados en
57                 el
```

```

49         estudio sistem tico de la complejidad computacional de
50         algoritmos geom tricos.</p>
51     <p>
52         Como paso de procesamiento previo, el arreglo de entrada se
53         ordena seg n las coordenadas x.
54     <ul>
55         <li>Encontrar el punto medio en el array ordenado, podemos
56         tomar  $P[n/2]$  como punto medio.</li>
57         <li>Divida el array dado en dos mitades. El primer subarray
58         contiene puntos de  $P[0]$  a  $P[n/2]$ . El segundo
59         subarray contiene puntos de  $P[n/2+1]$  a  $P[n-1]$ .</li>
60         <li>Encontrar recursivamente las distancias m s peque nas
61         en ambos subarrays. Dejando que las
62         distancias
63         sean  $d_l$  y  $d_r$ . Encontrar el m nimo de  $d_l$  y  $d_r$ . Dejando
64         que el m nimo sea  $d$ .</li>
65         <li>A partir de los 3 escalones anteriores, tenemos un
66         l mite superior  $d$  de distancia m nima. Ahora
67         tenemos
68         que considerar los pares de tal manera que un punto en
69         el par sea de la mitad izquierda y el otro
70         sea
71         de la mitad derecha. Considere la l nea vertical que
72         pasa por  $P[n/2]$  y encuentre todos los puntos
73         cuya
74         coordenada x est m s cerca que  $d$  de la l nea
75         vertical media. Construya una franja de matriz[] de
76         todos
77         esos puntos.</li>
78         <li>
79             5. Se ordena la franja de matriz[] seg un las
80             coordenadas y este paso es  $O(n \log n)$ . Se puede
81             optimizar a
82              $O(n)$ 
83             ordenando y fusionando recursivamente.</li>
84         <li>Se encuentra la distancia m as peque na en la tira[].
85         Esto es complicado. Desde el primer vistazo,
86         parece ser
87         un paso  $O(n^2)$ 
88         ), pero en realidad es  $O(n)$ . Se puede probar
89         geom tricamente que para cada punto de la
90         tira, solo necesitamos comprobar como m ximo 7 puntos
91         despu es de ella (tenga en cuenta que la tira
92         est a
93         ordenada de acuerdo con la coordenada Y).</li>
94         <li>Finalmente devuelva el m nimo de  $d$  y la distancia
95         calculada en el paso anterior.</li>
96     </ul>
97 </p>
98 </div>
99
100 <div class="row">
101     <div class="col-lg-6">
102         <div class=" container-fluid id="main">
103             <div id="canvas">
104                 <div class="container-fluid" id="botones">
105                     <button id="runbtn" class="btn btn-success mt-3"
106                         type="button">Iniciar</button>

```

```
94         <button id="clearbtn" class="btn btn-warning mt-3"
95             type="button">Reiniciar</button>
96     </div>
97 </div>
98 </div>
99 <div class="col-lg-3" id="list">
100     <ul id="group">
101     </ul>
102 </div>
103 </div>
104
105 </div>
106 <script src="https://code.jquery.com/jquery-3.6.0.slim.js"
107     integrity="sha256-HwWONEZrpuoh951cQD1ov2HUK5zA5DwJ1DNUXaM6FsY="
108     crossorigin="anonymous"></script>
109 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/
110     bootstrap.min.js"
111     integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy60rQ6VrjIEaFf/
112     nJGzIxFDsf4x0xIM+B07jRM"
113     crossorigin="anonymous"></script>
114 <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd
115     /popper.min.js"
116     integrity="sha384-
117     U02eT0CpHqdSJQ6hJty5KVphtPhzWj9W01clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
118     crossorigin="anonymous"></script>
119 <script src="https://cdnjs.cloudflare.com/ajax/libs/raphael/2.2.7/
120     raphael.js"></script>
121 <script src="main.js"></script>
122 <script type module src="points.js"></script>
123 </body>
124 </html>
```

### 5.3. style.css

```
1 body{
2   background-color: #fff;
3   font-family: Quicksand, Arial;
4 }
5
6   h3,h2{
7     text-align: center;
8     font-family: Quicksand, Arial;
9   }
10  #leftPanel{
11
12    display: inline-block;
13    vertical-align: top;
14    margin-left: -100px;
15    margin-right: auto;
16    height: 150px;
17  }
18  p{
19    text-align: justify;
20    font-family: Quicksand, Arial;
21    font-size: 20px;
22
23  }
24
25  #canvas {
26    height: 400px;
27    width: 500px;
28    cursor: crosshair;
29    margin-top: -90px;
30    margin-left: 50px;
31  }
32
33  #body button {
34    font-size: 1.2em;
35    width: 120px;
36  }
37  #botones{
38    text-align: center;
39  }
40  #p4,#p8,#p6,#p10{
41    margin-left: 20px;
42  }
43  #p5,#p9{
44    margin-left: 40px;
45  }
46  #currentState{
47    font-size: large;
48  }
49  #list{
50    margin-left: -80px;
51    margin-top: -50px;
52  }
53  marquee{
54    width: 250px;
55    height: 100px;
56    margin-top: 20px;
57  }
58  #equipo{
59    width: 300px;
```

```
60     height: 200px;
61     float: right;
62     margin-left: 20px;
63
64 }
65 #main{
66     height: 600px;
67     width: 600px;
68 }
69 li{
70     font-size: 20px;
71 }
```



## 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.