**ESTRUCTURA DE DATOS AVANZADOS**

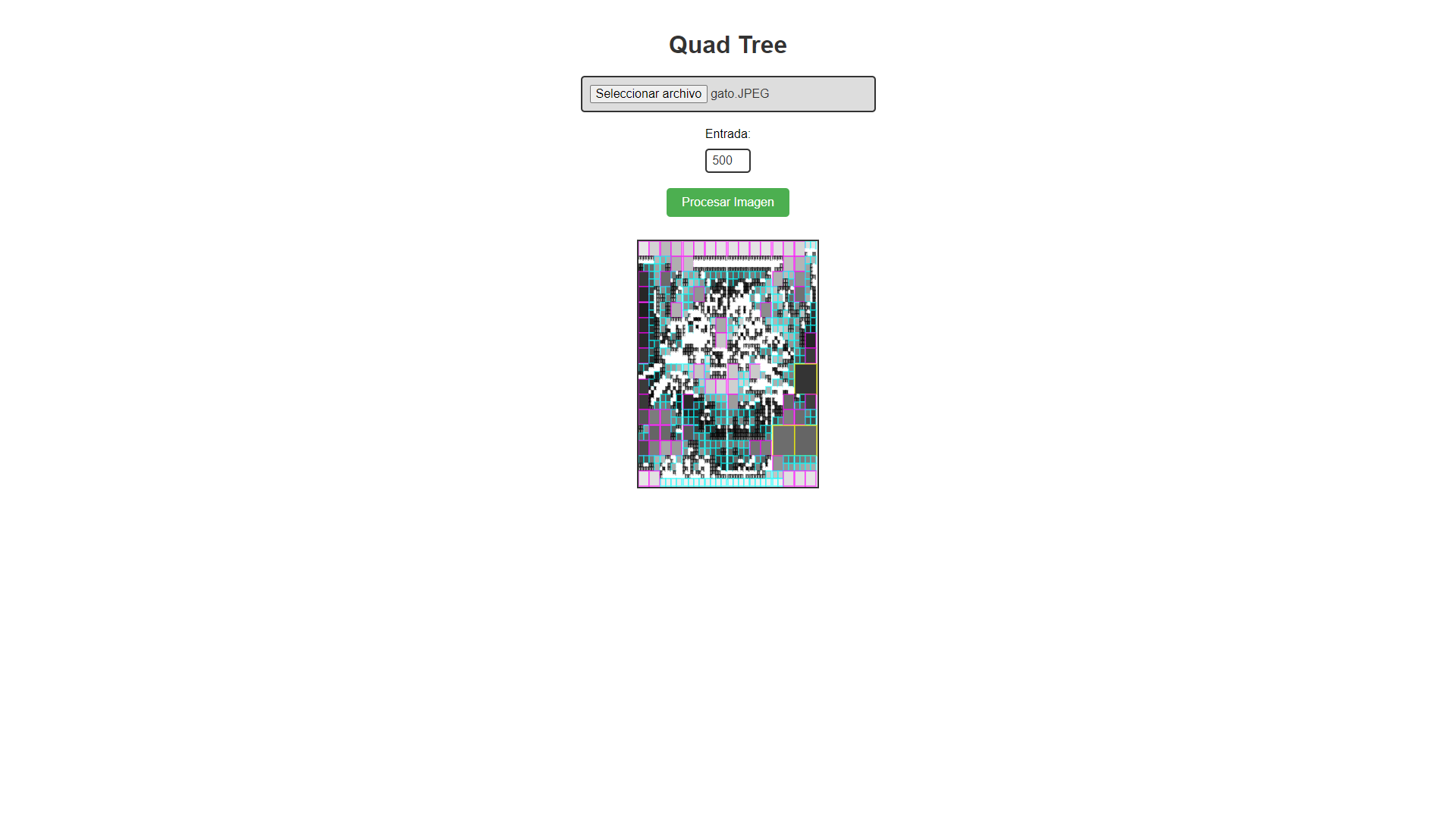
**PRESENTADO POR: BELTRAN EDWIN MAMANI MAMANI**

**Ejercicio 1: Descomposición de Imágenes con Quad Trees**

**1. Descripción del Problema**

Dada una imagen en escala de grises, divide la imagen en cuadrantes recursivamente hasta que cada región sea homogénea según un umbral. Implementa una función que construya el Quad Tree correspondiente y otra función que permita visualizar la estructura del Quad Tree.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  dieciséis  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  sesenta y cinco  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199 | <html lang= "es" >  <head>  <meta charset= "UTF-8" >  <meta name= "viewport" content= "width=device-width, inicial-scale=1.0" >  <title> Visualización de Quad Árbol </title>  <style>  body {  **font-family** : Arial , **sans-serif** ;  **color de fondo** : '#ff0000' ;  **margen** : **0** ;  **relleno** : **0** ;  }  **#contenedor** {  **text-align** : **center** ;  **relleno** : **20px** ;  }  h1 {  **color** : **#333** ;  }  entrada [ tipo = "archivo" ] {  **pantalla** : **bloque** ;  **margen** : **20px**  **automático** ;  **relleno** : **10px** ;  **borde** : **2px**  **sólido**  **#333** ;  radio de **borde** : **5px** ;  **cursor** : **puntero** ;  **tamaño de fuente** : **16px** ;  **color de fondo** : **#ddd** ;  **color** : **#333** ;  }  entrada [ tipo = "archivo" ] **:hover** {  **color de fondo** : **#ccc** ;  }  etiqueta , entrada [ tipo = "número" ] {  **tamaño de fuente** : **16px** ;  **margen** : **10px** ;  **bloqueo** de **pantalla** ;  **alineación de texto** : **centro** ;  }  entrada [ tipo = "número" ] {  **relleno** : **5px** ;  **borde** : **2px**  **sólido**  **#333** ;  radio de **borde** : **5px** ;  **alineación de texto** : **centro** ;  **ancho** : **60 píxeles** ;  **margen** : **10px automático** ;  **color de fondo** : **#fff** ;  **color** : **#333** ;  }  botón {  **color de fondo** : **#4CAF50** ;  **color** blanco ;  ​ **relleno** : **10px**  **20px** ;  **borde** : **ninguno** ;  radio de **borde** : **5px** ;  **cursor** : **puntero** ;  **tamaño de fuente** : **16px** ;  **margen** : **10px** ;  }  botón **: desplazar el cursor** {  **color de fondo** : **#45a049** ;  }  lienzo {  **borde** : **2px**  **sólido**  **#333** ;  **bloqueo** de **pantalla** ;  **margen** : **20px automático** ;  **color de fondo** : **#fff** ;  }  </style>  </head>  <body>  <div id= "container" >  <h1> Quad Tree </h1>  <input type= "file" id= "upload" Accept= "image/\*" >  <label for= "umbral" > Entrada: </label>  < tipo de entrada= "número" id= "umbral" valor= "50" min= "0" max= "255" >  <botón onclick= "processImage()" > Procesar Imagen </button>  <canvas id= "canvas" width= "512" height= "512" </canvas>  </div>  <script>  **class** QuadTree {  constructor(x, y, ancho, alto, umbral, profundidad = **0** ) {  **this** .x = x;  **este** .y = y;  **este** .ancho = ancho;  **este** .altura = altura;  **este** .umbral = umbral;  **este** .profundidad = profundidad;  **esto** .isLeaf = **false** ;  **este** .niños = [];  **esto** .mean = **nulo** ;  }  es homogéneo (datos de imagen) {  **sea** ​​total = **0** ;  **deja** contar = **0** ;  **for** ( **sea** i = **this** .y; i < **this** .y + **this** .height; i ++ ) {  **for** ( **sea** j = **this** .x; j < **this** .x + **this** .width; j ++ ) {  **let** index = (i \* datosimagen.ancho + j) \* **4** ;  **let** intensidad = imageData.data[index];  total += intensidad;  contar ++ ;  }  }  **let** significa = total / contar;  **esta** .media = media;  **sea** ​​varianza = **0** ;  **for** ( **sea** i = **this** .y; i < **this** .y + **this** .height; i ++ ) {  **for** ( **sea** j = **this** .x; j < **this** .x + **this** .width; j ++ ) {  **let** index = (i \* datosimagen.ancho + j) \* **4** ;  **let** intensidad = imageData.data[index];  varianza += Math .pow(intensidad - media, **2** );  }  }  varianza /= recuento;  variación **de retorno** <= **este** .umbral;  }  subdividir (datos de imagen) {  **if** ( **este** .ancho <= **1**  || **este** .alto <= **1**  || **este** .isHomogeneous(imageData)) {  **este** .isLeaf = **true** ;  **devolver** ;  }  **let** halfWidth = Math .floor( **this** .width / **2** );  **let** halfHeight = Math .floor( **this** .height / **2** );  **este** .children.push( **new** QuadTree( **este** .x, **este** .y, medio ancho, medio alto, **este** .umbral, **este** .profundidad + **1** ));  **this** .children.push( **new** QuadTree( **this** .x + halfWidth, **this** .y, halfWidth, halfHeight, **this** .threshold, **this** .profundidad + **1** ));  **este** .children.push( **new** QuadTree( **este** .x, **este** .y + media altura, media anchura, media altura, **este** .umbral, **este** .profundidad + **1** ));  **this** .children.push( **new** QuadTree( **this** .x + halfWidth, **this** .y + halfHeight, halfWidth, halfHeight, **this** .threshold, **this** .profundidad + **1** ));  **for** ( **dejar** hijo de **este** .children) {  child.subdivide(imageData);  }  }  dibujar (contexto) {  **si** ( **este** .isLeaf) {  contexto.fillStyle = ` rgb(${ **este** .significado}, ${ **este** .significado}, ${ **este** .significado}) ` ;  context.fillRect ( **este** .x, **este** .y, **este** .ancho, **este** .alto);  contexto.strokeStyle = **esto** .getColorByDepth(); // Color de la línea de división  context.strokeRect( **this** .x, **this** .y, **this** .width, **this** .height);  } **else** {  **para** ( **dejar** hijo de **este** .children) {  niño.draw(contexto);  }  }  }  getColorByDepth() {  colores **const** = [  '#ff0000' , // rojo  '#00ff00' , // verde  '#0000ff' , // azul  '#ffff00' , // amarillo  '#ff00ff' , // magenta  '#00ffff' , / / cian  '#000000' , // negro  '#ffffff' // blanco  ];  **devolver** colores [ **este** .profundidad % colores.longitud];  }  }  **función** ProcessImage() {  **const** fileInput = documento .getElementById( 'cargar' );  **const** umbralInput = documento .getElementById( 'umbral' );  umbral **constante** = parseInt (umbralInput.valor, **10** );  **const** lienzo = documento .getElementById ( 'lienzo' );  contexto **constante** = lienzo.getContext( '2d' );  **if** (fileInput.files && fileInput.files[ **0** ]) {  **const** lector = **nuevo** FileReader();  lector.onload = **función** (e) {  **const** img = **nueva** imagen();  img.onload = **function** () {  // Ajustar el lienzo al tamaño de la imagen  canvas.width = img.width;  lienzo.altura = img.altura;  // Dibujar la imagen en el lienzo  context.drawImage(img, **0** , **0** , img.width, img.height);  **const** imageData = contexto.getImageData ( **0** , **0** , lienzo.ancho, lienzo.alto);  **const** quadTree = **nuevo** QuadTree ( **0** , **0** , lienzo.ancho, lienzo.alto, umbral);  quadTree.subdivide(imageData);  contexto.clearRect( **0** , **0** , lienzo.ancho, lienzo.alto);  quadTree.draw(contexto);  };  img.src = e.objetivo.resultado;  };  lector.readAsDataURL(fileInput.files[ **0** ]);  }  }  </script>  </body>  </html> |



**2. Descripción del Problema**

Dada una imagen en escala de grises, queremos representar la imagen utilizando un Quad Tree para reducir el tamaño de almacenamiento. Después, se debe reconstruir la imagen a partir del Quad Tree y comparar la imagen original con la imagen reconstruida.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  dieciséis  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  sesenta y cinco  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165 | <html lang= "es" >  <head>  <meta charset= "UTF-8" >  <meta name= "viewport" content= "ancho=ancho del dispositivo, escala inicial=1.0" >  <título> Imagen de árbol cuádruple </title>  <style>  body {  **font-family** : Arial , **sans-serif** ;  **alineación de texto** : **centro** ;  **color de fondo** : **#f0f0f0** ;  }  **.contenedor** {  **pantalla** : flex;  **justificar** - **contenido** : espacio - alrededor;  alinear elementos : **centrar** ;  ​ **margen superior** : **20px** ;  }  **.canvas-container** {  **text-align** : **center** ;  }  lienzo {  **borde** : **1px**  negro **sólido** ;  **margen superior** : **10px** ;  }  **.controles** {  **margen superior** : **20px** ;  }  botón {  **relleno** : **10px**  **20px** ;  **tamaño de fuente** : **16px** ;  **cursor** : **puntero** ;  **color de fondo** : **#4CAF50** ;  **color** blanco ;  ​ **borde** : **ninguno** ;  radio de **borde** : **4px** ;  transición : **color de fondo**  **0,3 s** ;  }  botón **: desplazar el cursor** {  **color de fondo** : **#45a049** ;  }  </style>  </head>  <body>  <h1> Imagen Quad Tree </h1>  <div class= "container" >  <div class= "canvas-container" >  <h2> Imagen Original </h2>  <canvas id= "imagen-original" ancho= "500" alto= "500" >>canvas  > </div>  <div class= "canvas-container" >  <h2> Imagen Reconstruida </h2>  <canvas id= " reconstructed-image" width= "500" height= "500" >>canvas  > </div>  </div>  <div class= "controls" >  <button id= "rebuildButton" > Reconstruir Imagen </button>  < /div>  <script>  // Función para dividir la imagen en cuadrantes recursivamente  **function** divideImage(imageData, x, y, w, h, umbral) {  **if** (w <= **1**  || h <= **1** ) {  **return** [[x, y, w,h]];  }  **let** regiones = [];  **let** halfWidth = Math .floor(w / **2** );  **let** halfHeight = Math .floor(h / **2** );  // Dividir la imagen en cuatro cuadrantes  regiones.push(divideImage(imageData, x, y, halfWidth, halfHeight, umbral));  regiones.push(divideImage(imageData, x + halfWidth, y, halfWidth, halfHeight, umbral));  regiones.push(divideImage(imageData, x, y + halfHeight, halfWidth, halfHeight, umbral));  regiones.push(divideImage(imageData, x + halfWidth, y + halfHeight, halfWidth, halfHeight, umbral));  // Filtrar las regiones que no sean homogéneas  regiones = regiones.filter(region => {  **let** pixelCount = **0** ;  **let** sum = **0** ;  **for** ( **let** i = **0** ; i < region.length; i ++ ) {  **let** pixel = imageData[(region[i][ **1** ] \* imageData.width + region[i][ **0** ]) \* **4** ] // Se multiplica por 4 para acceder a los canales RGBA  pixelCount ++ ;  suma += píxel;  }  **let** promedio = suma / pixelCount;  **return**  Math .abs(promedio - imageData[(región[ **0** ][ **1** ] \* imageData.width + región[ **0** ][ **0** ]) \* **4** ]) <= umbral;  });  **regiones de retorno** ;  }    **function** buildQuadTree(imageData, regiones) {  **let** quadTree = [];  **for** ( **sea** i = **0** ; i < regiones.longitud; i ++ ) {  **sea** región = regiones[i];  **dejar** nodo = {  x : región[ **0** ],  y : región[ **1** ],  w : región[ **2** ],  h : región[ **3** ],  color : [  imageData[(región[ **1** ] \* imageData.width + región[ **0** ]) \* **4** ], // Rojo  imageData[(región[ **1** ] \* imageData.width + región[ **0** ]) \* **4**  + **1** ], // Verde  imageData[(región[ **1** ] \* imageData.width + región[ **0** ]) \* **4**  + **2** ] // Azul  ]  };  quadTree.push(nodo);  }  **devolver** árbol cuádruple;  }  // Función para reconstruir la imagen a partir del Quad Tree  **function** reconstructImage(quadTree, canvas) {  **let** ctx = canvas.getContext( '2d' );  **let** imageData = ctx.getImageData( **0** , **0** , lienzo.ancho, lienzo.alto);  **for** ( **let** i = **0** ; i < quadTree.length; i ++ ) {  **let** node = quadTree[i];  **for** ( **sea** y = nodo.y; y < nodo.y + nodo.h; y ++ ) {  **for** ( **sea** x = nodo.x; x < nodo.x + nodo.w; x ++ ) {  **sea** índice = (y \* datosimagen.ancho + x) \* **4** ;  imageData.data[índice] = nodo.color[ **0** ];  imageData.data[índice + **1** ] = nodo.color[ **1** ];  imageData.data[índice + **2** ] = nodo.color[ **2** ];  imageData.data[índice + **3** ] = **255** ;  }  }  }  ctx.putImageData(imageData, **0** , **0** );  }  // Cargar la imagen y construir el Quad Tree al cargar la página  **let** originalCanvas = document .getElementById( 'original-image' );  **let** reconstructedCanvas = document .getElementById ( 'imagen reconstruida' );  **let** ctx = originalCanvas.getContext( '2d' );  **let** imagen = **nueva** Imagen();  imagen.onload = **función** () {  ctx.drawImage(imagen, **0** , **0** , originalCanvas.width, originalCanvas.height);  **let** imageData = ctx.getImageData( **0** , **0** , originalCanvas.width, originalCanvas.height).data;  **let** regiones = divideImagen(imageData, **0** , **0** , originalCanvas.width, originalCanvas.height, **10** );  **let** quadTree = buildQuadTree(imageData, regiones);  reconstructImage(quadTree, reconstructedCanvas);  };    imagen.src = 'gaton.jpg' ;  // Botón para reconstruir la imagen cuando se haga clic  **let** reconstruirButton = document .getElementById( 'rebuildButton' );  reconstruirButton.addEventListener ( 'hacer clic' , () => {  ctx.clearRect( **0** , **0** , originalCanvas.width, originalCanvas.height);  reconstructedCanvas.getContext( '2d' ).clearRect( **0** , **0** , reconstructedCanvas.width, reconstructedCanvas.height);  ctx.drawImage(imagen, **0** , **0** , originalCanvas.width, originalCanvas.height);  **let** imageData = ctx.getImageData( **0** , **0** , originalCanvas.width, originalCanvas.height).data;  **let** regiones = divideImagen(imageData, **0** , **0** , originalCanvas.width, originalCanvas.height, **10** );  **let** quadTree = buildQuadTree(imageData, regiones);  reconstructImage(quadTree, reconstructedCanvas);  });  </script>  </body>  </html> |