UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

ESCUELA PROFESIONAL DE CIENCIA DE LA COMPUTACIÓN

Curso : Estructura de Datos Avanzados
KD - Tree

Trabajo presentado por

HERRERA COOPER, MIGUEL ALEXANDER

Índice

Índice de figuras

1. Antecedentes

- Inventado en 1970 por Jon Bentley.
- El nombre originalmente significaba "árboles 3d, árboles 4d, etc."donde k era el elemento de dimensiones.
- Ahora, la gente dice "árbol kd de dimensión d".
- Idea: cada nivel del árbol se compara con 1 dimensión.
- Tengamos solo dos hijos en cada nodo (en lugar de 2d).

2. KD - Tree

2.1. Definición

Un Árbol KD (abreviatura de árbol k-dimensional) es una estructura de datos de particionado del espacio que organiza los puntos en un Espacio euclídeo de k dimensiones.
Los árboles kd son un caso especial de los árboles BSP.

2.2. Características

- Un árbol kd emplea sólo planos perpendiculares a uno de los ejes del sistema de coordenadas. Esto difiere de los árboles BSP, donde los planos pueden ser arbitrarios.
- Además, todos los nodos de un árbol kd, desde el nodo raíz hasta los nodos hoja, almacenan un punto. Mientras tanto, en los árboles BSP son las hojas los únicos nodos que contienen puntos (u otras primitivas geométricas). Como consecuencia, cada plano debe pasar a través de uno de los puntos del árbol kd.
- Técnicamente, la letra k se refiere al número de dimensiones. Un árbol kd tridimensional podría ser llamado un árbol 3d.
- Las letras k y d se escriben en minúsculas, incluso al principio de una oración. La k se escribe en cursiva, aunque son también comunes las formas "árbol KDz "árbol Kd".

2.3. Construir un Árbol kd

Dado que hay muchas maneras posibles de elegir planos alineados a los ejes, hay muchas maneras de generar árboles kd. El sistema habitual es:

- Conforme se desciende en el árbol, se emplean ciclos a través de los ejes para seleccionar los planos. (Por ejemplo, la raíz puede tener un plano alineado con el eje x, sus descendientes tendrían planos alineados con el y y los nietos de la raíz alineados con el z, y así sucesivamente).
- En cada paso, el punto seleccionado para crear el plano de corte será la mediana de los puntos puestos en el árbol kd, lo que respeta sus coordenadas en el eje que está siendo usado.

Este método lleva a un árbol kd balanceado, donde cada nodo hoja está a la misma distancia de la raíz. De todas formas, los árboles balanceados no son necesariamente óptimos para todas las aplicaciones.

Dada una lista de n puntos, el siguiente **algoritmo** genera un árbol kd balanceado que contiene dichos puntos.

```
function kdtree (list of points pointList, int depth)
{
    if pointList is empty
        return nil;
    else
    {
        // Select axis based on depth so that axis cycles through all valid values
        var int axis := depth mod k;

        // Sort point list and choose median as pivot element
        sort pointList using predicate: point1[axis] < point2[axis];
        choose median from pointList;

        // Create node and construct subtrees
        var tree_node node;
        node.location := median;
        node.leftChild := kdtree(points in pointList before median, depth+1);
        node.rightChild := kdtree(points in pointList after median, depth+1);
        return node;
    }
}
```

3. Código

```
k=2;
class Node{
  constructor(point, axis){
    this.point=point;
    this.left= null;
    this.right = null;
    this.axis = axis;
  }

}

function isEmpty(){
  return this.points.length==0;
}


function getHeight(node){
  var sl=0;
  var sr=0;

  if(node.left!=null){
    sl=getHeight(node.left);
  }

  if(node.right!=null){
    sr= getHeight(node.right);
  }

  if(sl>sr){
    return sl+1;
  }
  else{
    return sr+1;
  }
}


function generate_dot(node){

  if(node==null){return;}

  if(node.left){
    console.log('"'+node.point+'"'+"->"+'"'+node.left.point+'"');
    generate_dot(node.left);
  }
  if(node.right){
    console.log('"'+node.point+'"'+"->"+'"'+node.right.point+'"');
    generate_dot(node.right);
  }

}

function build_kdtree(points,depth = 0){
  var axis=depth%k;
  var median;
  if (points.length==0){
    return null;
  }
  if(points.length==1){
    return new Node(points[0],axis)
```

```
61    }
62    median=Math.floor(points.length/2);
63
64    points.sort(function(a,b){return a[axis]-b[axis]});
65
66    var left=points.slice(0,median);
67    var right=points.slice(median+1);
68    var node = new Node(points[median].slice(0,k),axis);
69    node.left = build_kdtree(left,depth+1);
70    node.right = build_kdtree(right,depth+1);
71    return node;
72    }
```

kdtree.js

Este algoritmo crea el invariante para cualquier nodo. Todos los nodos en el subárbol de la izquierda están en un lado del plano de corte, y todos los nodos del subárbol de la derecha están en el otro lado. El plano de corte de un nodo pasa a través del punto asociado con ese nodo (referenciado en el código por node.location)

4. Pruebas

4.1. Resultado en el Navegador

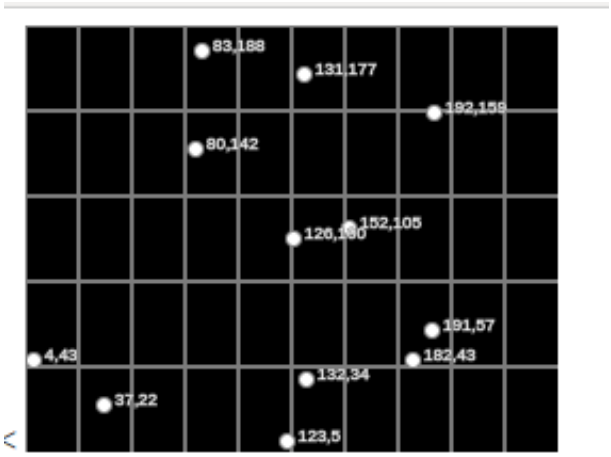

Figura 1: Visualización del kd-Tree en 2 dimensiones

4.2. Resultado en la Consola del Navegador

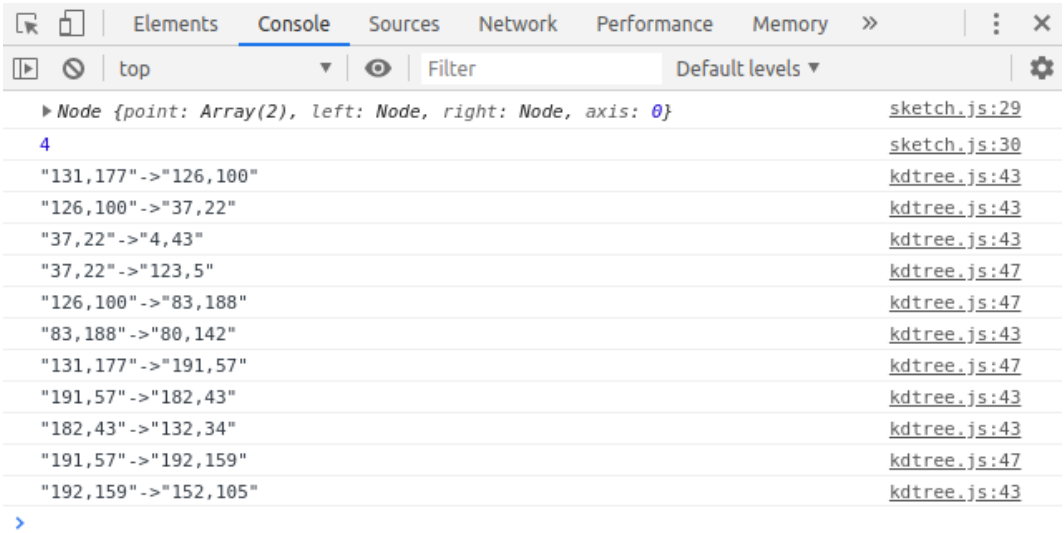

Figura 2: 4 indica la altura del árbol, los datos son los resultados de la estructura del árbol

4.3. Visualización en GraphViz

Para la obtención de la visualización del arbol generado por el kd-Tree con Dimensión k=2, usaremos Web GraphViz.

Primero pasaremos los datos que nos fueron otorgados en la consola:

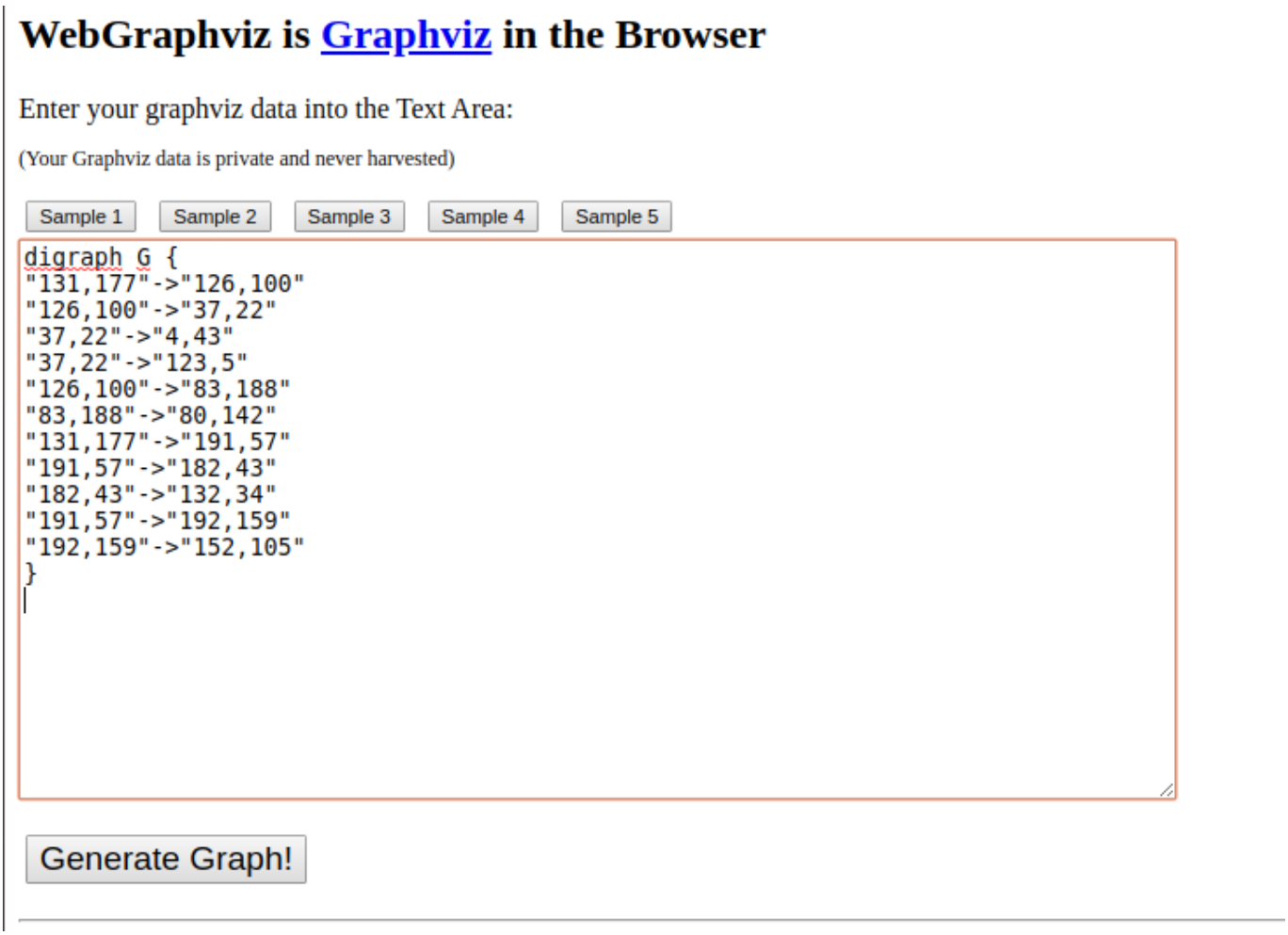

Figura 3: Datos de .dot

Al dar click en el botón Generate, obtenemos nuestro árbol :

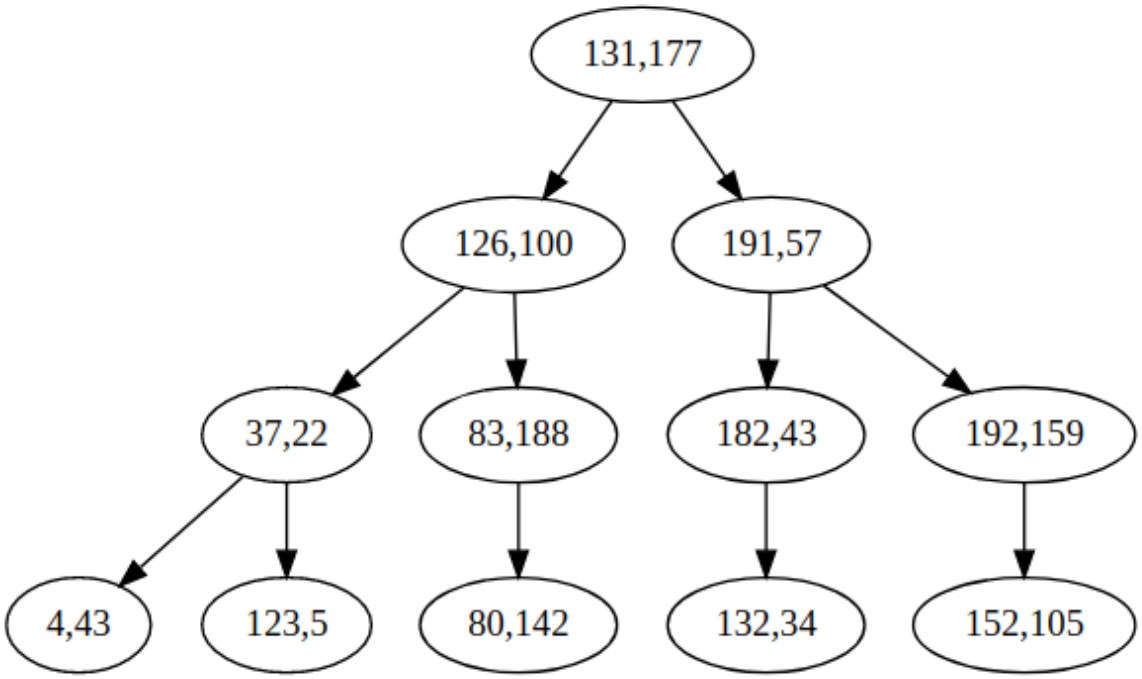

Figura 4: Datos de .dot

5. Referencias

- https://es.wikipedia.org/wiki/
- https://www.geeksforgeeks.org/k-dimensional-tree/