

## Recuperación de información

- Proporcionar una estructura de datos
- Debe estar justificada
-

## Datos multidimensionales

los e.d nos sirven para recuperar data de manera rápida y eficiente

- Árboles
  - Matrices
  - listas
  - arrays
  - Tabla de Hash
- 

### Challenges

- Preprocessing cost
- Storage cost
- Access cost
- Visualization

- Coordenadas espaciales

### Data type

#### Categoricos

Nominal

Ordinal

#### Numericos

Interval

Ratio

## La maldición de la dimensionalidad

- Bellman (1920 - 1984)

describe los problemas causados por causa exponencial del volumen de los datos asociado

---

### Sampling

$$\epsilon(P(x_i, \in \text{ bin}_k)) \sim \frac{N}{k^M}$$

Solución Agg mas puntos.

---

### Searching

- Nearest Neighbors (NN) Search
  - Range search
-

P <sub>1</sub>	=	3   4   5   6   7   3   9   2   1   3
P <sub>2</sub>	=	5   3   3   3   4   2   9   8   1   5
P <sub>3</sub>	=	6   2   5   4   1   6   5   3   4   3
P <sub>4</sub>	=	7   4   4   9   1   5   7   5   6   5
P <sub>5</sub>	=	9   7   5   6   3   4   9   7   8   2
P <sub>6</sub>	=	6   8   6   5   4   3   8   9   1   8
P <sub>7</sub>	=	3   6   8   2   5   2   6   8   0   9
P <sub>8</sub>	=	1   7   2   3   6   1   4   0   6   1

$$\begin{aligned}
 & (3-5)^2 + (4-7)^2 + (5-1)^2 + (6-1)^2 + (7-4)^2 + (8-1)^2 + (9-9)^2 + (2-8)^2 + (1-1)^2 + (3-1)^2 \\
 & (-2)^2 + (1)^2 + (2)^2 + (7)^2 + (3)^2 + (4)^2 + (0)^2 + (-6)^2 + (-6)^2 + (2)^2 \\
 & 4 + 1 + 4 + 9 + 9 + 16 + 0 + 36 + 0 + 4 = \sqrt{85} = 9,11
 \end{aligned}$$

$$\frac{7}{6} + \frac{5}{4}$$

$\sqrt{85}$

$60^\circ + 39^\circ \sim 71.6^\circ$

7  
2  
1

## Spatial Data Structures

Expo → Día lunes 07 de abril : 9:30 am

- Árbol K-d (k-dimensional Tree)
- Quad Tree
- Octree
- R-tree
- BSP Tree
- Grid Spatial Index

A medida que aumentamos las dimensiones, aparece un espacio cada vez más grande porque los datos se hacen más esparsos.

Consisten en datos espaciales, líneas, regiones, puntos, rectángulos

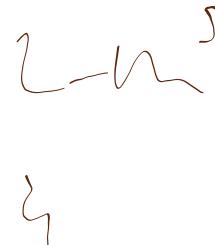
los cuales potencialmente podrían incluir una asociación del tiempo

### B-Tree

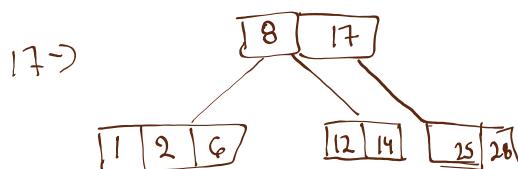
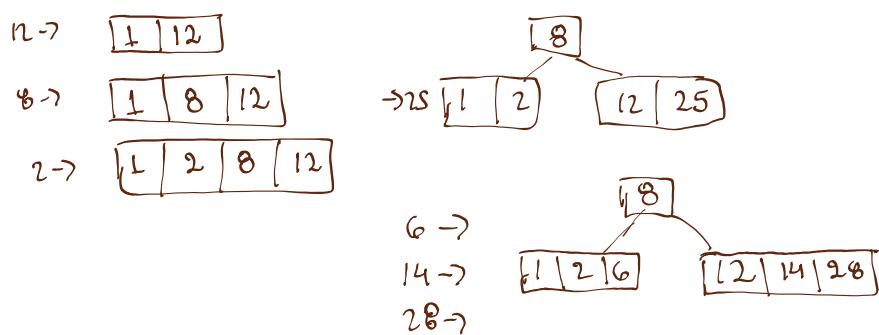
Es una E.D que cambia altura x anchura

aumentar los hijos

- cada nodo debe tener hasta  $m$  hijos
- todas las hojas están en el mismo nivel
- todos los nodos no hoja tienen techo menor raíz
- La raíz puede ser también un nodo hoja de 2 hasta  $m$  hijos
- Raíz → nodo - hoja
- Un nodo hoja contiene  $m-1$  datos
- El número  $m$  debería ser impar
- Cant de datos : orden -1



[1]



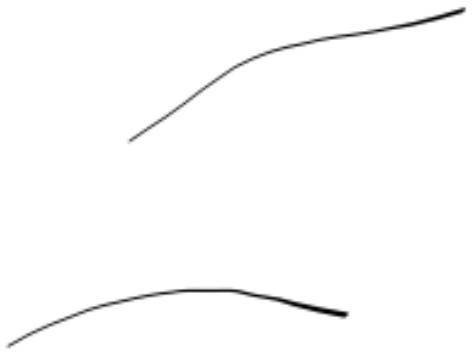
### Proyecto final :

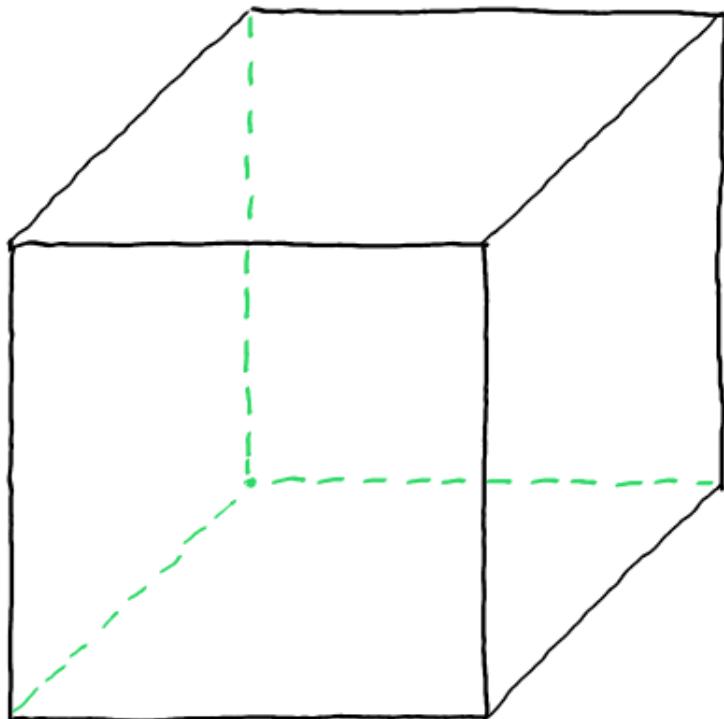
- Open streetmap
- o Latitud - longitud
- o Hacer consultas anidadas tiempo -

### E.D

- Hashed cubes
- Nano cubes ✓
- Cantidad de casos (Devolución)
- o

- Quad tree
  - Point Quad Tree
- NO han quadrants que  
resolvieren
- P R Quadtree





$$S = 17 \text{ ph}$$

$$(S^3 - 1)/7$$

$$(17^3 - 1)/7$$

$$(4913 - 1)/7$$

$$4912 / 7 = 701.714$$

Las EO son incrementales

Una EO es incremental cuando no necesita rehacer la estructura

- Los árboles binarios son incrementales
- QuadTree es incremental

### Region Quad Tree

- Criterio de parada: Que los datos sean iguales

### Point Quad Tree

### Kd - Tree

- Almacena o intenta escalar a K dimensiones
- Alterna las dimensiones
- Distancia euclídea (dist min)

#### Version 2D

- Vecino mas cercano
  - Busque dentro rango
  - Si está o no está un dato
- Son Bentley, 75

Formar para hacer un KD-tree

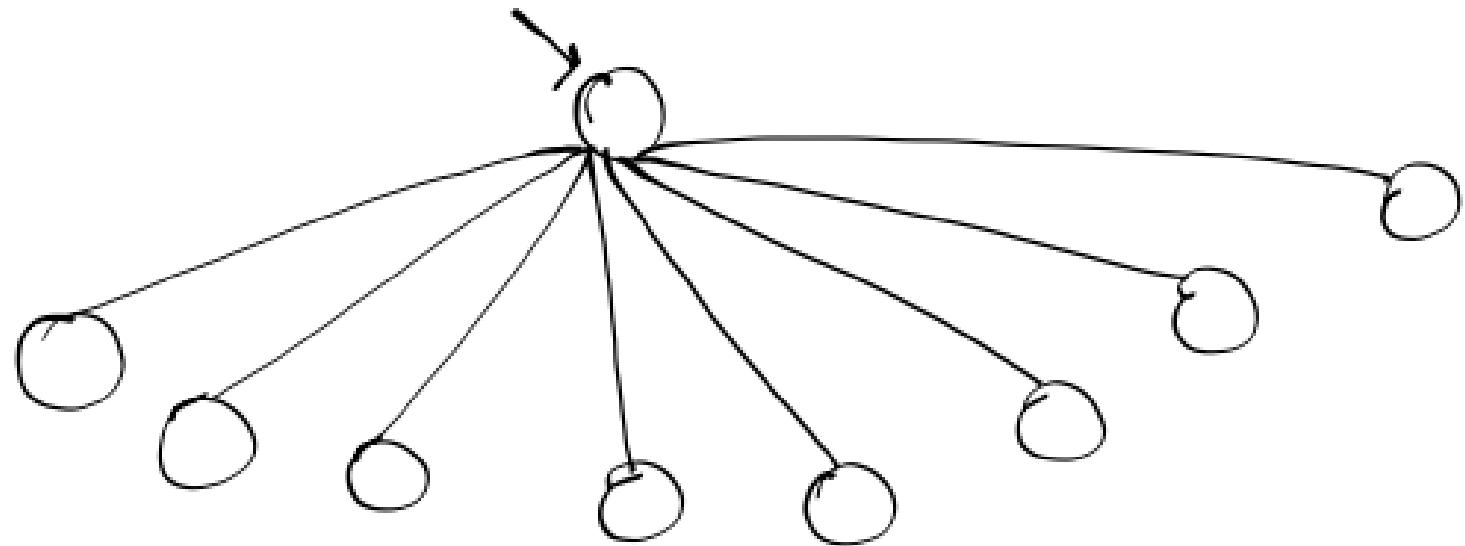
- Escoger la mediana y definir una linea horizontal o vertical
- Recursividad en ambos lados.

### PR Quadtree

- Es una adaptación del Region Quadtree para puntos asociados
- Los nodos hojas están vacíos ó contienen pts y sus coordenadas
- Un cuadrante contiene algo mucho un punto

### Bs-tree

- Se expandía en anchura que en altura
- Cant de datos:  $m-1$  (hijos -1)



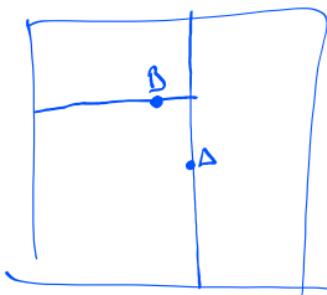
## Kd-tree

- Las hojas representan un hiperplano que divide el espacio en dos partes
- Alternamos los cortes mediante las dimensiones

---

### Estrategias para el Kd-tree

- Dividir basado en el orden (vemos la discriminante) de la inserción de los pts
- Dividir en contrando la mediana
- Dividir perpendicularmente con el mayor exarcamiento



- media ponderada

porque en esta estrategia elegimos la dimensión con mayor dispersión (el eje)

-- split perpendicular

B - K - D - tree

- el xpo IS de abril

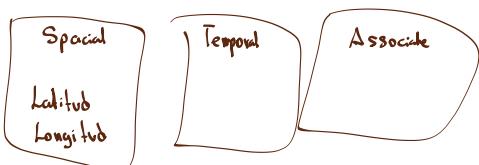
KD B-tree multidimensional variations  
es un arbol para subdividir  
a K-dimensiones el espacio

- Region pages : Es una colección de pares (región, hijo)  
Regiones

- Point pages : Colección (punto, locación)  
data, puntos

- R-gutman
  - R\*
  - R-tree
- Voy leyendo  
Dibujar
- r-tree gutman
- Algoritmo de inserción R-tree

Tu eliges puntos aplicando diferentes criterios



Knn → similitud  
de datos

Víctor Reyna

vcsp tesis repositorio Víctor Reyna computer science

## Proyecto Final

- Trabajo Personal
- Datos geocodificados y multidimensionales

Latitud - longitud

geocodificada	datos multidimensionales asociados	→ numéricos	
Latitud	Longitud	type	attrn
→			
→			
→			
→			
→			

- Estructura de datos estática

Recomendación

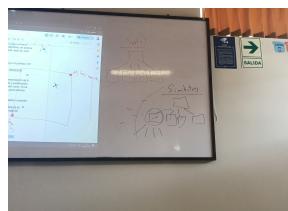
- Formato Binario mas rápido (Convertir)

Similitud: Aquellos datos que tienen características semejantes o parecidas distancias menores.

Consultas:

- Similitud de datos (dato) retorna los puntos más parecidos
- ¿Cuántos grupos similares hay en una región determinada?
- ¿Cuáles son los puntos más parecidos en Jose en esta región?

- Preprocesar los datos como sea conveniente



- Métodos de cluster y preprocesamiento
- Mapa 2D

struct Point {

```
int id
double latitud, longitud;
vector<int> atributos; //igen: Vector Xd atributos
int id_inicial_cluster;
```

- Preprocesamiento

## Dynamica : R-Tree

Aprender insertion R-tree

### Propiedades

- Todos los nodos hoja contienen m y M records a no ser que sea raiz
- Id, int contiene nBR

- Priorizaremos velocidad de consulta.
  - Datos no dinámicas, datos de aprox 1000000.
- 

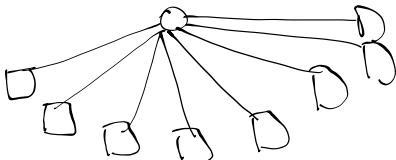
Estructuras Pensadas por mi:

- Quad - Tree (parte latitud y longitud)
  - KD-tree (parte asociativo atributos)
  - Octree - Rtree (parte geográfica)
- 

¿Que funciones necesitamos?

- Insertar\_punto ( struct punto )
- Encontrar\_puntos\_similares\_por\_asociación ( punto, latitud\_min , latitud\_max , long\_min , long\_max )
- Encontrar\_puntos\_similares\_en\_rango\_delimitado ( latitud\_min , latitud\_max , long\_min , long\_max )

a<sub>0</sub>, a<sub>1</sub>, a<sub>2</sub>, a<sub>3</sub>, a<sub>4</sub>, a<sub>5</sub>, a<sub>6</sub>, a<sub>7</sub>, a<sub>8</sub>, a<sub>9</sub>



## Introducción

### Arboles:

- Kd tree
- quad tree
- range tree
- polygon tree
- quinary tree
- multiple attribute tree (MAT)

Cuando los elementos tienen pesos asociados los elementos se llaman elementos ponderados.

Si es así entonces el árbol de estos elementos se llama árbol ponderado de K-dimensiones

### ¿Qué es una Estructura de Datos Multidimensional?

Se dice que un dato es unidimensional si se caracteriza por un único atributo o valor clave.

### E.D. Unidimensionales

- Binary-tree equilibrado
- Árboles B
- AVLs
- Tabla Hash
- Árboles Multidireccionales

Un dato es multidimensional si se caracteriza por una serie de atributos.

### Arboles:

- Kd tree
- quad tree
- multiple attribute tree (MAT)
- multidimensional B tree
- multidimensional and weighted tree

El rendimiento de una estructura de datos multidimensional se mide en términos de estas tres cantidades

- N puntos - k atributos
- P(N) Coste de preprocessamiento, que es el tiempo para construir la estructura de datos.
- S(N) Coste de almacenamiento, que es la cantidad de memoria necesaria para almacenar la estructura de datos
- Q(N) Coste de acceso, que es el tiempo necesario para acceder a la e.d según las necesidades. Puede ser el coste de consulta al responder una consulta sobre los datos. En general coste de inserción, eliminación, consultas o cualquier otra modificación

## R-tree

### Choose Leaf:

$I$  es el BMR de  $F$

- Selecciona a un nodo hoja  $L$  en el cuál colocamos la nueva entrada  $E$

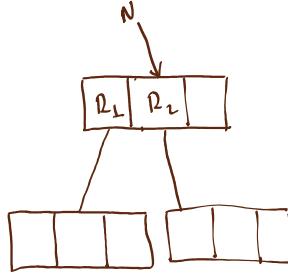
CL1) Se selecciona  $N$ , como nodo raiz

CL2) Si  $N$  es una hoja, retorna  $N$

CL3) Si  $N$  no es hoja, sea  $F$  la nueva entrada de  $N$  cuyo rectángulo  $F.I$  necesita la menor ampliación para incluir  $E.I$

CL4) Descendemos hasta encontrar una hoja.

Vamos a setear  $N$  como el nodo hijo al que apunta  $F.p$  y repite desde CL2



### Empate

#### Criterio

- 1) Crecimiento
- 2) Área real

$E.I \rightarrow MBR$

### Split Node

- Quadratic Split: Cuando se llega a un nodo y hay  $n+1$  entradas, dividimos en dos grupos

QS1: Aplicamos el algoritmo Pick Seeds

#### Pick Seeds:

PS1: Para cada par de entradas  $E_1$  y  $E_2$ , componer un rectángulo  $J$  que incluya  $E_1.I$  y  $E_2.I$ . Calcular:  $d = \text{área}(J) - \text{área}(E_1.I) - \text{área}(E_2.I)$ .

PS2: Elija el par con la  $d$  mayor.

QS2: Si todas las entradas han sido asignadas. Si es que no, si un grupo tiene pocas entradas todo el resto tiene que ser asignado al nodo que tienen pocas entradas

QS3: Llamamos al algoritmo Pick Next para elegir la sgte entrada al ser asignada.

#### Pick Next:

PN1: Para cada entrada  $E$ , que no está en un grupo 1, calculamos  $d_1 = \text{el área que se incrementa en el rectángulo del grupo 2 incluido } E.I$ . Calcula  $d_2$  similar para el grupo 2

PN2: Encontrar la entrada con la max diferencia entre  $d_2$  y  $d_1$

#### Linear Split:

LS1: Linear Pick Seeds:

LPS1:

### Nodos intermedios

- tupla < puntero al hijo, MBR>

### Nodos hijos

- tupla < puntero

I4: Si la propagación del nodo split causa que la raíz se divide creamos una nueva raíz cuyos hijos son los dos nodos nuevos resultantes

### AdjustTree

Subimos desde un nodo hoja  $L$  hasta la raíz ajustando los rectángulos de cobertura y las divisiones de nodos de propagación.

AT1: Establecer  $N=L$ , si  $L$  se dividió previamente, establecer  $NN$  como el segundo nodo resultante

AT2: Si  $N$  es la raíz, paramos.

AT3: Sea  $P$  el nodo padre de  $N$ , y sea  $E_N$  la entrada de  $N$  en  $P$ . Ajustamos  $E_N.I$  para que englobe todos los rectángulos de entrada en  $N$ .

AT4: Si  $N$  tiene una pareja  $NN$  resultante de una división anterior, crea una nueva entrada  $E_{NN}$  con  $E_{NN}.P$  apuntando a  $NN$  y  $E_{NN}.I$  encierra todos los rectángulos en  $NN$ . Agrega  $E_{NN}$  a  $P$  si hay espacio, de lo contrario llamamos a SplitNode para crear  $P$  y  $PP$  que contengon  $E_{NN}$  y todas las entradas antiguas  $P$ 's.

AT5: Establecemos  $N=P$  y  $NN=PP$  si se produjo una división. Repetimos desde AT2

## R-tree Análisis

RTree.h :

- struct Rect
- struct Branch
- struct Node
- struct List Node
- struct PartitionVars
- Class RTree
  - Rtree () ;
  - Rtree (const RTree & other) ;
  - virtual ~RTree () ;
  - vector <vector <pair <int, int>>> mObjs;

## Propiedades

- Todas las hojas están en el mismo nivel

## Node Splitting

### Método

- 1) Encuentrar una partición para dividir el nodo
- 2) Crear dos nuevos nodos si es necesario
- 3) Propagar los cambios hacia arriba y hacia abajo (nodos intermedios)

## SNI (Find a Partition)

- x Usamos la Partition , del Pack algoritmo
- x Partition (  $p_1$  Red) let  $s_1$  y  $s_2$

## Insert

### Node splitting

Pack  
Partition

20 atributos y

2 (long y lat)