

El árbol R*: Un acceso eficiente y robusto

Método

para puntos y rectángulos

Norbert Beckmann, Hans-Peter Begel
Ralf Schneider, Bernhard Seeger

Presentado por Snehal Thakkar

Contenido

Terminología

Resumen

Introducción

Variantes del árbol R

Árbol R*

Experimentos

Conclusiones

Terminología

Cuadro delimitador

también conocido como Rectángulo Mínimo Delimitador (MBR)

rectángulo más pequeño paralelo al eje que encierra el valor SDT

Rectángulo de directorio

geométricamente el MBR de los rectángulos subyacentes

Margen

la suma de las longitudes de los bordes de un rectángulo

PAM (Método de acceso por punto)

Una estructura de datos y algoritmos asociados principalmente para buscar puntos definidos en el espacio multidimensional.

SAM (Método de acceso espacial)

Una estructura de datos para buscar líneas, polígonos, etc.

Abstracto

Árbol R

basado en la optimización heurística del área del rectángulo circundante en cada nodo interno

heurística: un algoritmo que normalmente, pero no siempre, funciona o que da casi la respuesta correcta

Árbol R*

incorpora una optimización combinada del área, el margen y la superposición de cada rectángulo circundante en el directorio

superan las variantes existentes del árbol R

Admitir eficientemente datos puntuales y espaciales al mismo tiempo

El costo de implementación es solo ligeramente más alto que el de otros árboles R

Introducción

Métodos de acceso espacial (SAM)

aproximar un objeto espacial complejo mediante MBR de ejes paralelos
un objeto complejo está representado por un número limitado de bytes

Aunque se pierde mucha información, la MBR de objetos espaciales
ya conserva las propiedades geométricas más esenciales • la
ubicación del objeto • la
extensión del objeto en cada eje

El SAM más popular para almacenar rectángulos: R-tree

Árboles R

Basado en el árbol PAM B+ utilizando regiones superpuestas

Criterio de optimización: minimizar el área de cada rectángulo envolvente en los nodos internos

M: número máximo de entradas en un nodo

m: número mínimo de entradas en un nodo ($2 \leq m \leq M/2$)

Satisfacer lo siguiente:

la raíz tiene al menos dos hijos a menos que sea una hoja

cada nodo que no sea hoja tiene entre m y M hijos a menos que es la raíz

cada nodo hoja contiene entre m y M entradas a menos que sea la raíz

Todas las hojas aparecen en el mismo nivel

Árboles R

Un árbol R (R^* -árbol) es completamente dinámico

Las inserciones y eliminaciones se pueden combinar con consultas.

no se requiere una reorganización global periódica

La estructura debe permitir rectángulos de directorio superpuestos

no se puede garantizar que solo se requiera una ruta de búsqueda para una consulta de coincidencia exacta

Este artículo muestra que la técnica de regiones superpuestas no implica un mal desempeño de recuperación promedio.

Problema principal en los árboles R

Los parámetros conocidos de un buen rendimiento de recuperación se afectan entre sí de una manera muy compleja

Es imposible optimizar uno de ellos sin influir en los demás.

Los rectángulos de datos tienen diferentes tamaños/formas y los rectángulos de directorio crecen y se reducen dinámicamente.

El éxito de los métodos que optimizarán un parámetro es muy incierto.

Se aplica un enfoque heurístico

basado en muchos experimentos diferentes realizados en un marco sistemático

Criterios de optimización

(O1) Minimizar el área cubierta por un rectángulo de directorio

minimizar el espacio muerto

mejorar el rendimiento ya que las decisiones sobre qué caminos deben recorrerse se pueden tomar en niveles superiores

(O2) Minimizar la superposición entre rectángulos de directorio

disminuir el número de caminos a recorrer

(O3) Minimizar el margen de un rectángulo de directorio

El rectángulo del directorio tendrá una forma más cuadrática

Los objetos cuadráticos se pueden empaquetar más fácilmente

rectángulos de directorio más pequeños en el nivel superior

(O4) Optimizar la utilización del almacenamiento

La altura del árbol se mantendrá baja.

reducir el costo del usuario

Mejoramiento

(+) Minimizar área

menor cobertura del espacio de datos, por lo que se puede reducir la superposición

(+) Más cuadrático

Mejor embalaje, por lo tanto, es más fácil mantener una alta utilización del almacenamiento

(-) Minimizar el área y la superposición

requieren más libertad en la cantidad de rectángulos almacenados en un nodo menor

utilización de almacenamiento

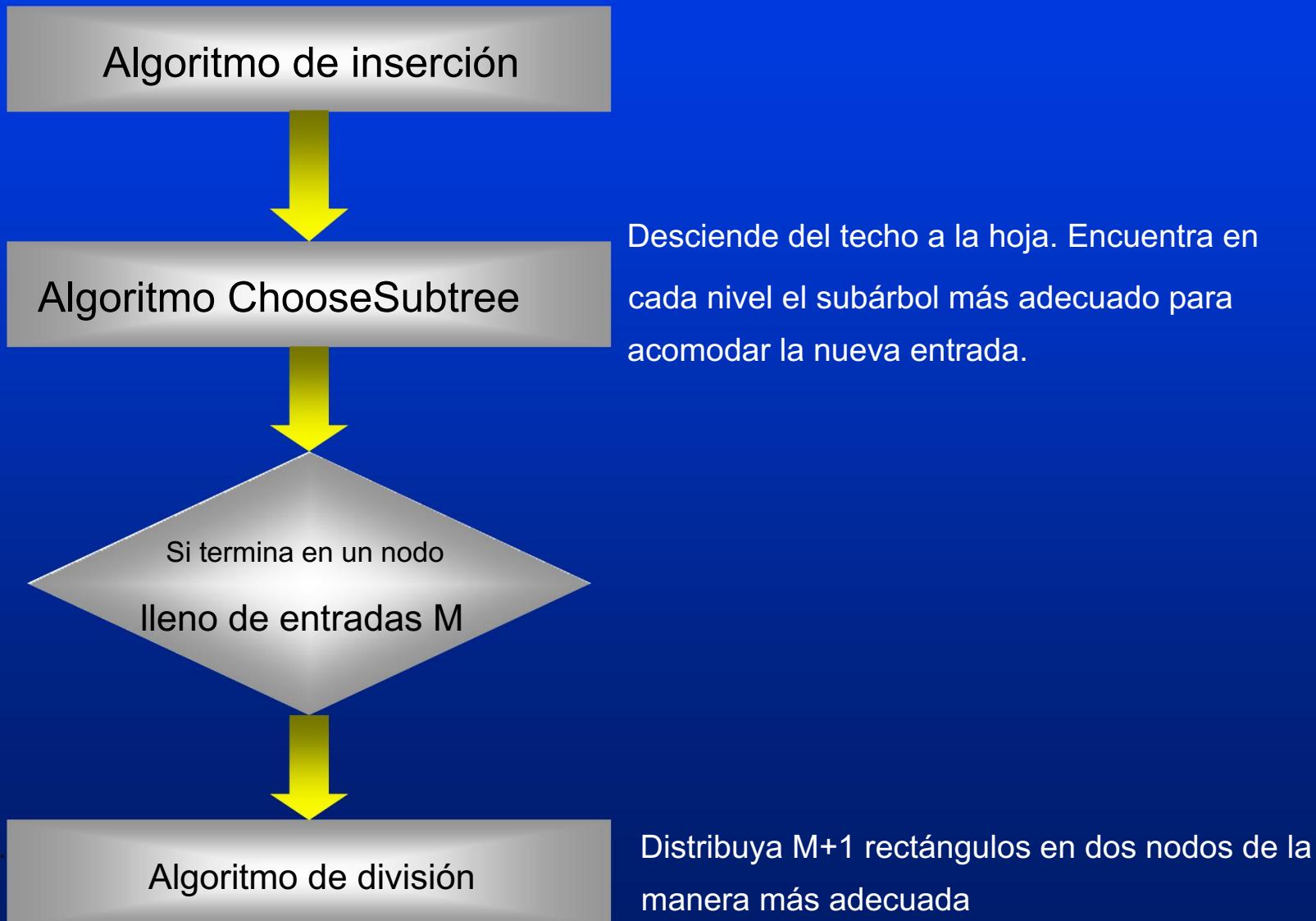
rectángulos “menos cuadráticos”

(-) Minimizar el margen

reducir la utilización del almacenamiento

Variantes del árbol R

Optimización de recuperación durante la inserción



Árbol R original (de Guttman)

Método de optimización

minimizar el área cubierta por un rectángulo de directorio

también puede reducir la superposición

El costo de la CPU será relativamente bajo

Árbol R original (de Guttman)

Algoritmo ChooseSubtree

CS1 [Iniciar] Establezca N como el nodo raíz

CS2 [Comprobación de hojas]

Si N es una hoja,

devolver N

demás

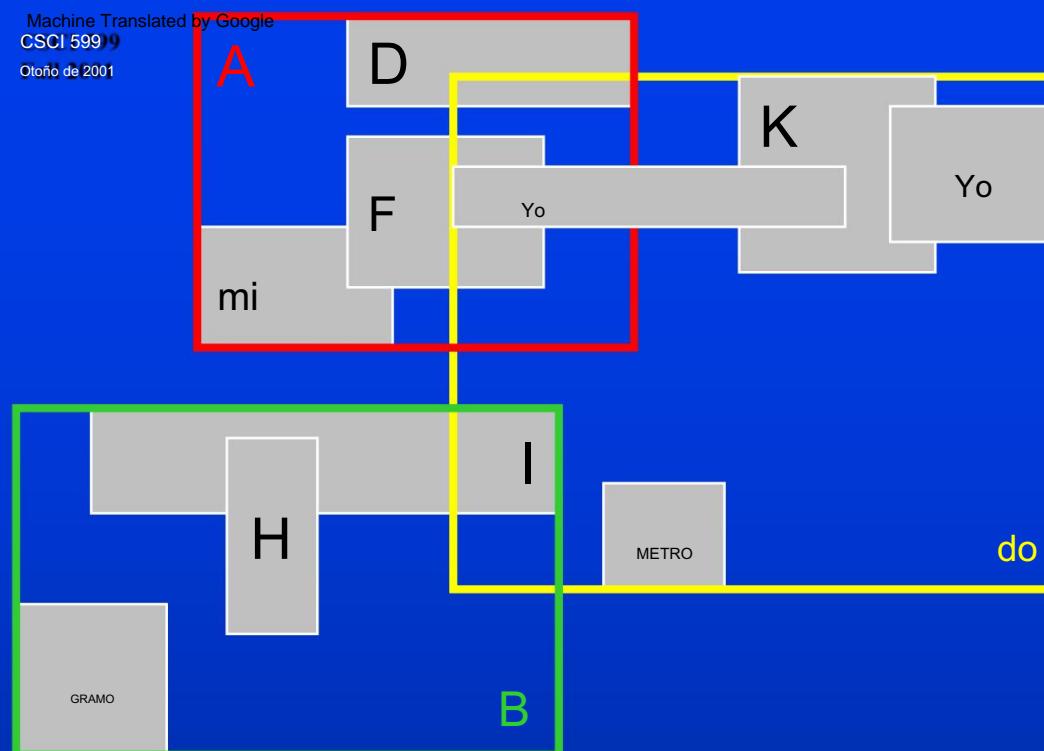
[Elegir subárbol]

Seleccione la entrada en N cuyo rectángulo requiera la menor ampliación de área para incluir los nuevos datos. Resuelva los empates seleccionando la entrada con el rectángulo de menor área.

fin

CS3 [Descender hasta llegar a una hoja]

Establezca N como el nodo secundario al que apunta el puntero secundario de la entrada seleccionada. Repetir desde CS2.

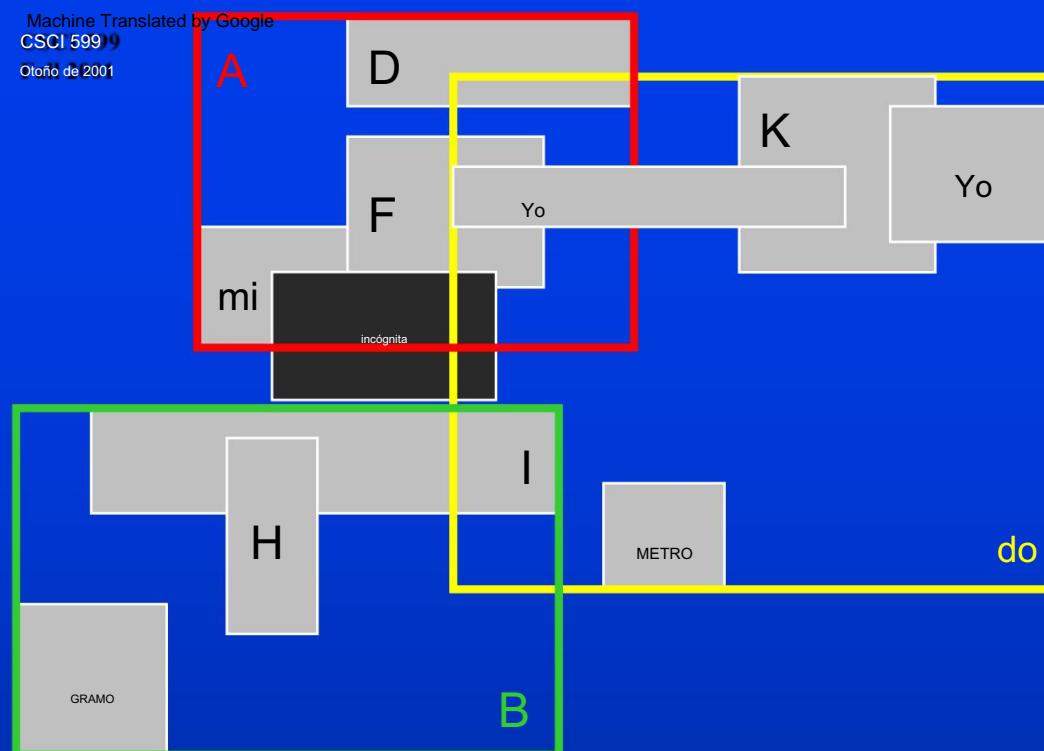


abecedario

DEF

GHI

JKLM

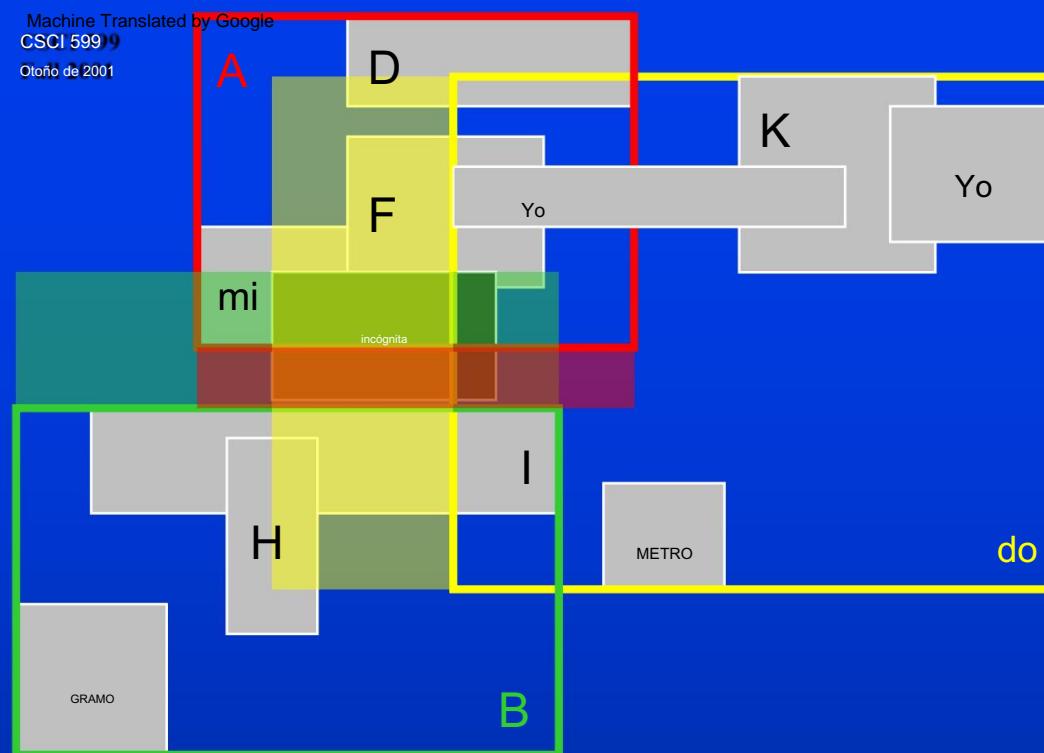


abecedario

DEF

GHI

JKLM



Raíz

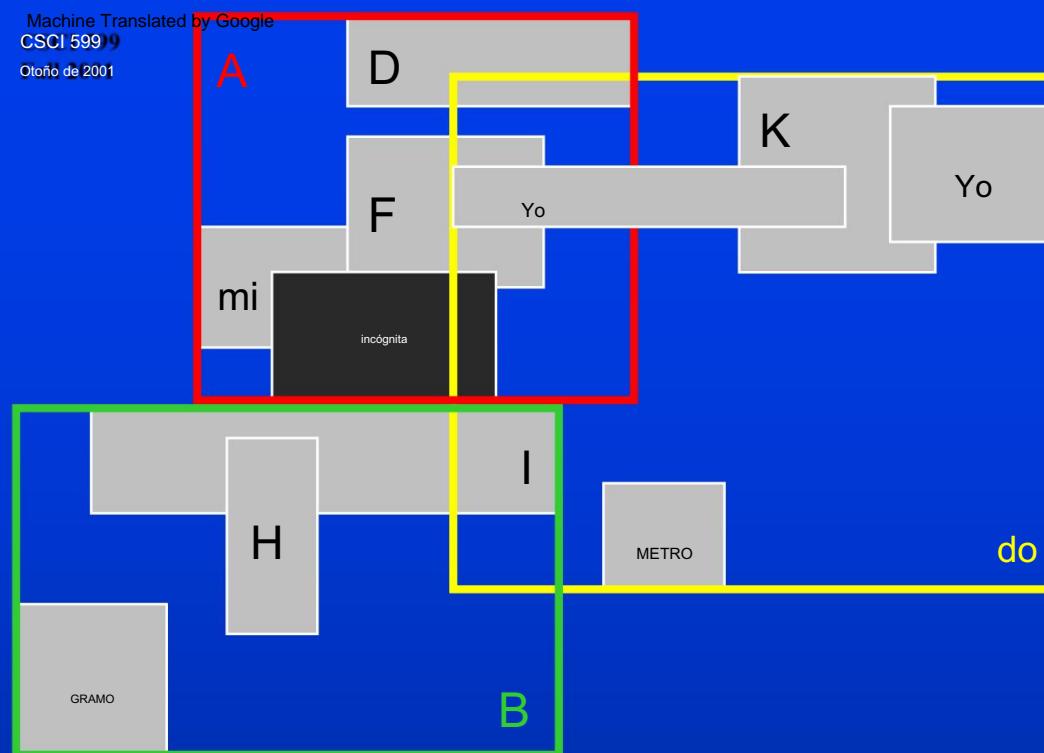
norte

abecedario

DEF

GHI

JKLM



Raíz

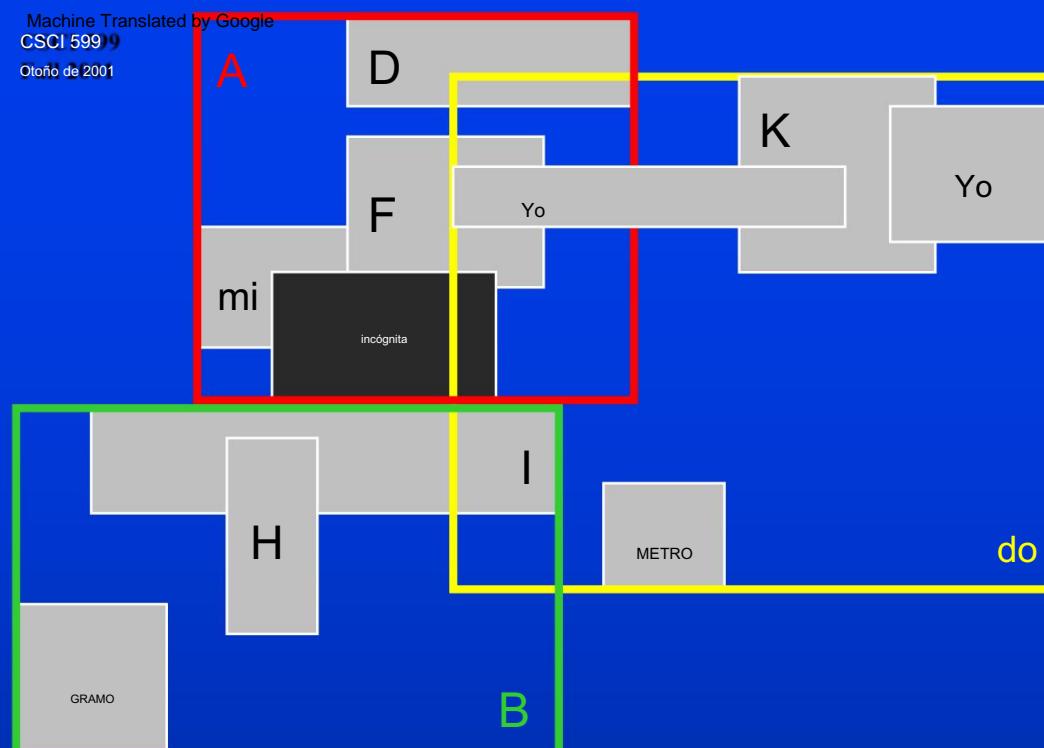
norte

abecedario

DEF

GHI

JKLM



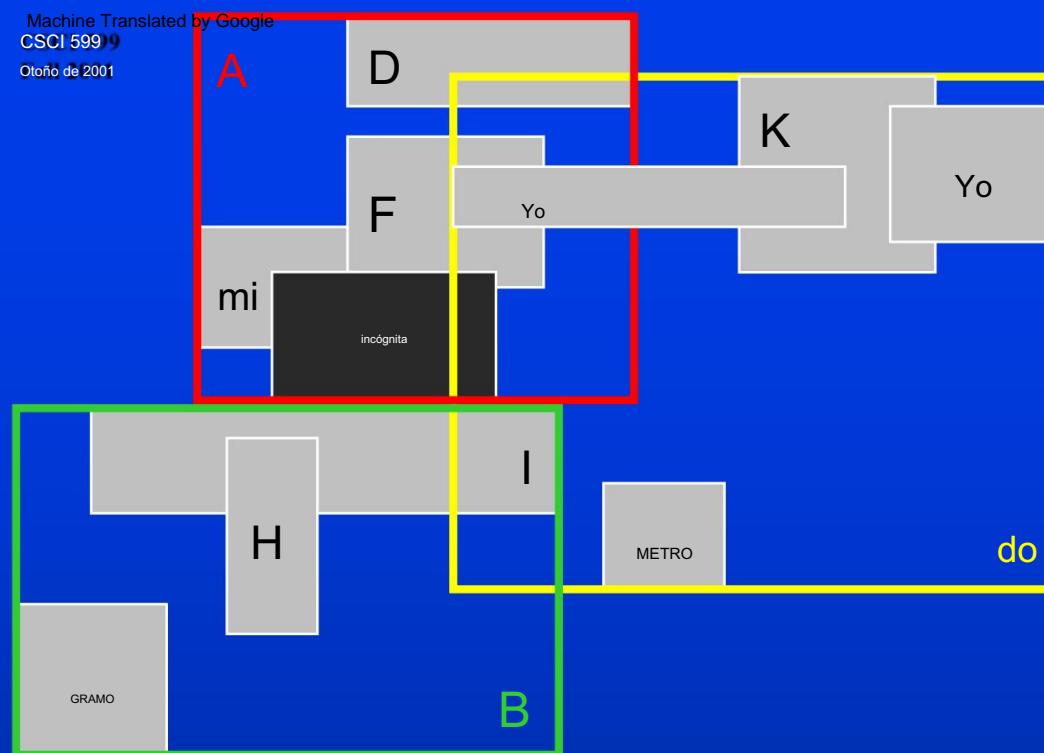
abecedario



N DEF

GHI

JKLM



abecedario

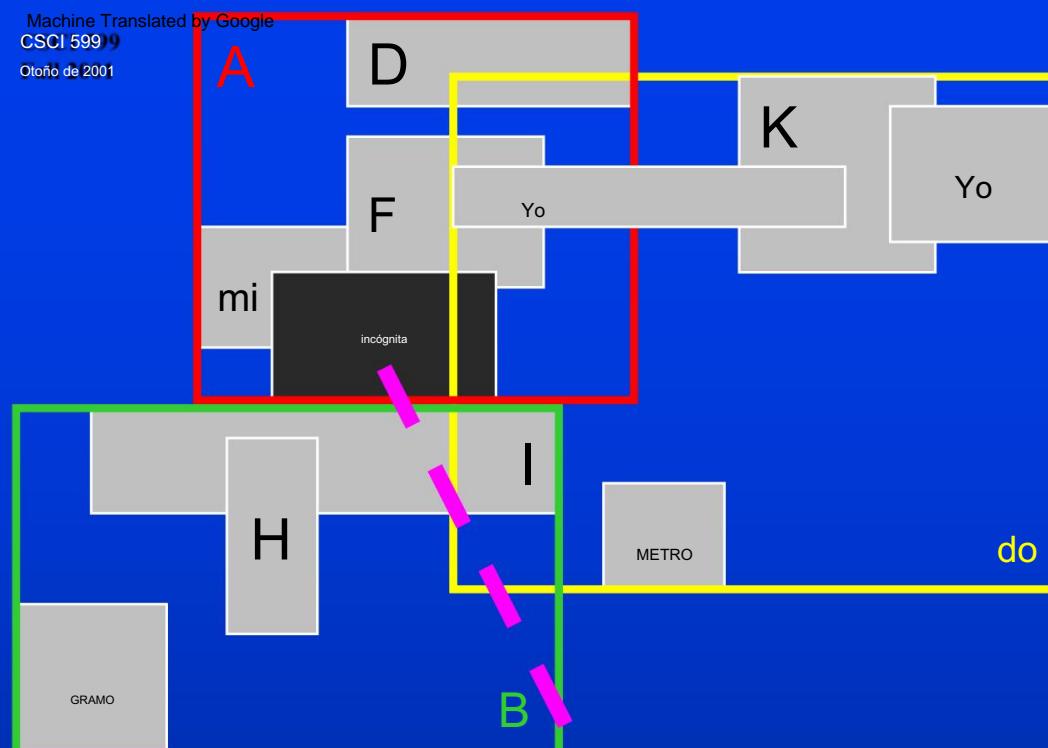
Dado que N = Hoja,
parada y retorno N

DEF

GHI

JKLM

Insertar



abecedario

DEF X GHI

JKLM

Algoritmos de división

Tres versiones

Todos están diseñados para minimizar el área cubierta por dos rectángulos resultantes de la división.

Exponencial

encuentre el área con mínimo global El costo de CPU es demasiado alto

Cuadrático y Lineal

encontrar aproximación

[Guttman] rendimiento de recuperación casi igual [este artículo] el modelo cuadrático tiene un rendimiento mucho mejor que el lineal

Algoritmo QuadraticSplit

[Dividir un conjunto de entradas de índice M+1 en dos grupos]

QS1 [Seleccione la primera entrada para cada grupo]

Invoque PickSeeds para elegir dos entradas, cada una de las cuales será la primera entrada de cada grupo

QS2 [Marcar si está hecho]

Repetir

DistribuirEntrada

hasta

Se distribuyen todas las entradas o uno de los dos grupos tiene $\tilde{M}m+1$ entradas (para que el otro grupo tenga m entradas)

QS3 [Seleccione la entrada para asignar]

Si quedan entradas, asígnelas al otro grupo para que tenga el número mínimo m requerido

Algorithm PickSeeds

[Elija dos entradas para que sean las primeras entradas de los grupos]

PS1 [Calcular la ineficiencia de agrupar entradas]

Para cada par de entradas E1 y E2, componga un rectángulo R que incluya el rectángulo E1 y el rectángulo E2

Calcular $d = \text{área}(R) - \text{área}(E1 \text{ rectángulo}) - \text{área}(E2 \text{ rectángulo})$

PS2 [Elige el par más desperdiciado]

Elige el par con la d más grande

[las semillas tenderán a ser pequeñas, si los rectángulos son de tamaño muy diferente (y) o la superposición entre ellos es alta]

Algoritmo DistributeEntry

[Asignar las entradas restantes según el criterio de área mínima]

DE1 Invoca PickNext para elegir la siguiente entrada que se asignará

DE2 Agregarlo al grupo cuyo rectángulo de cobertura tendrá que

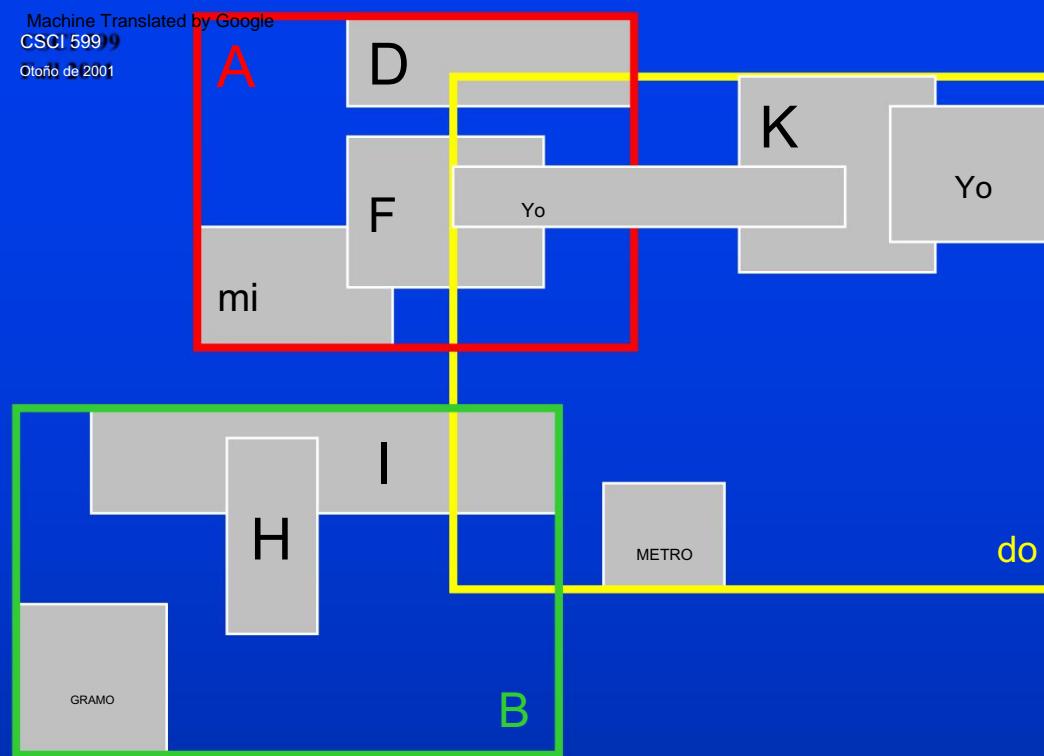
Se ampliará al menos para acomodarlo. Resuelva los empates agregando la entrada al grupo con el área más pequeña, luego al que tiene menos entradas, luego a cualquiera de los dos.

Algoritmo PickNext

[elige la entrada con el mejor valor de bondad de área en cada situación]

DE1 Para cada entrada E que aún no esté en un grupo, calcule $d_1 =$ el aumento de área requerido en el rectángulo de cobertura del Grupo 1 para incluir el Rectángulo E. Calcule d_2 de forma análoga para el Grupo 2.

DE2 Elija la entrada con la máxima diferencia entre d_1 y d_2 .

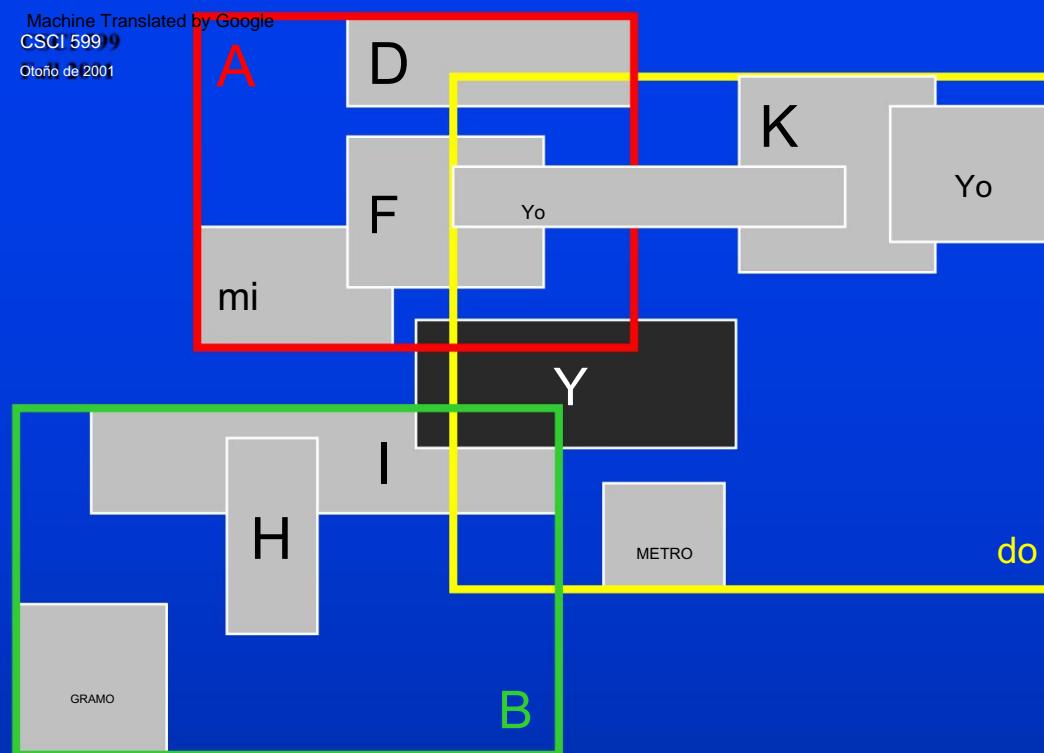


abecedario

DEF

GHI

JKLM

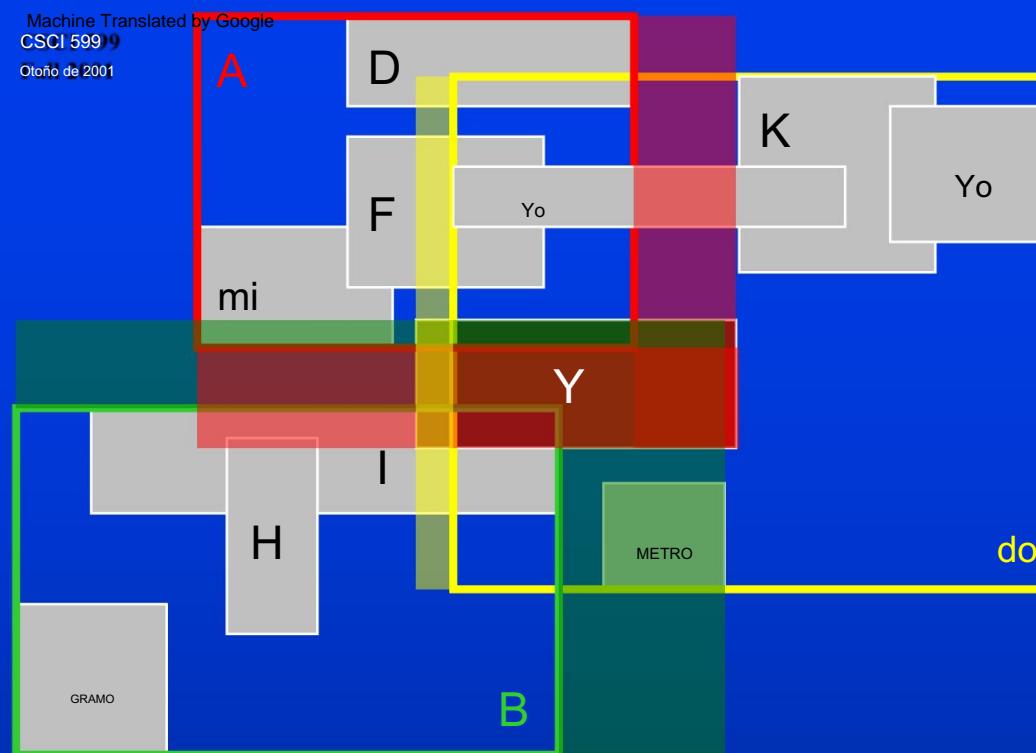


abecedario

DEF

GHI

JKLM



Raíz

abecedario

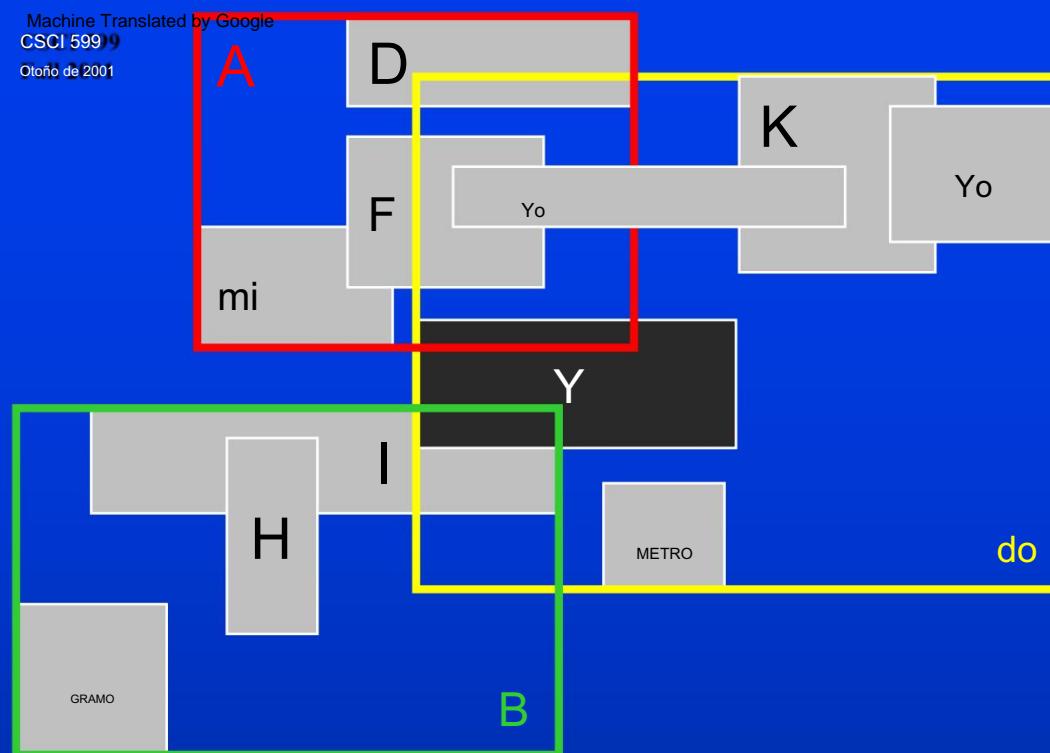


norte

DEF

GHI

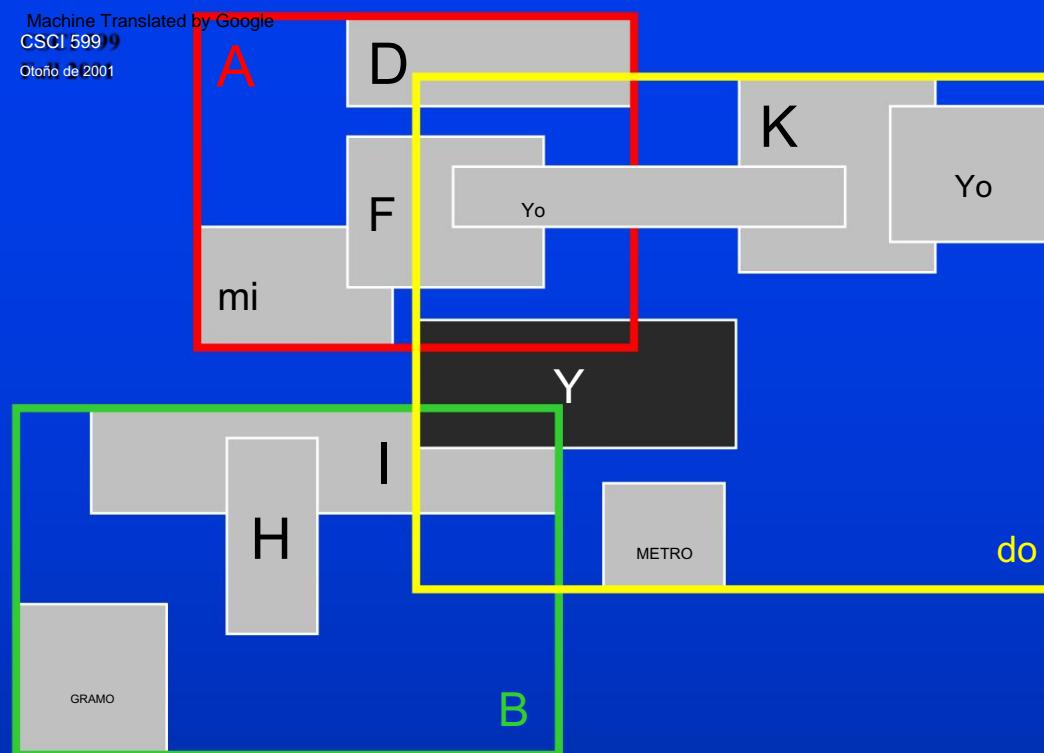
JKLM



DEF

GHI

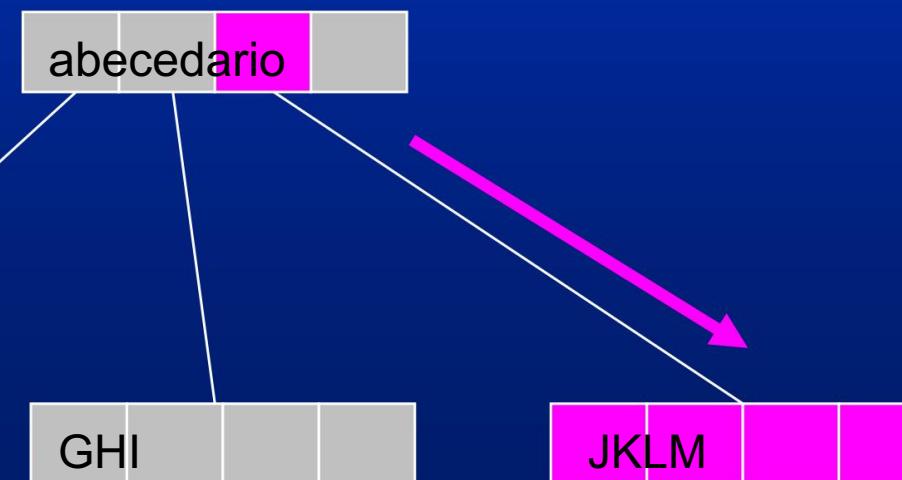
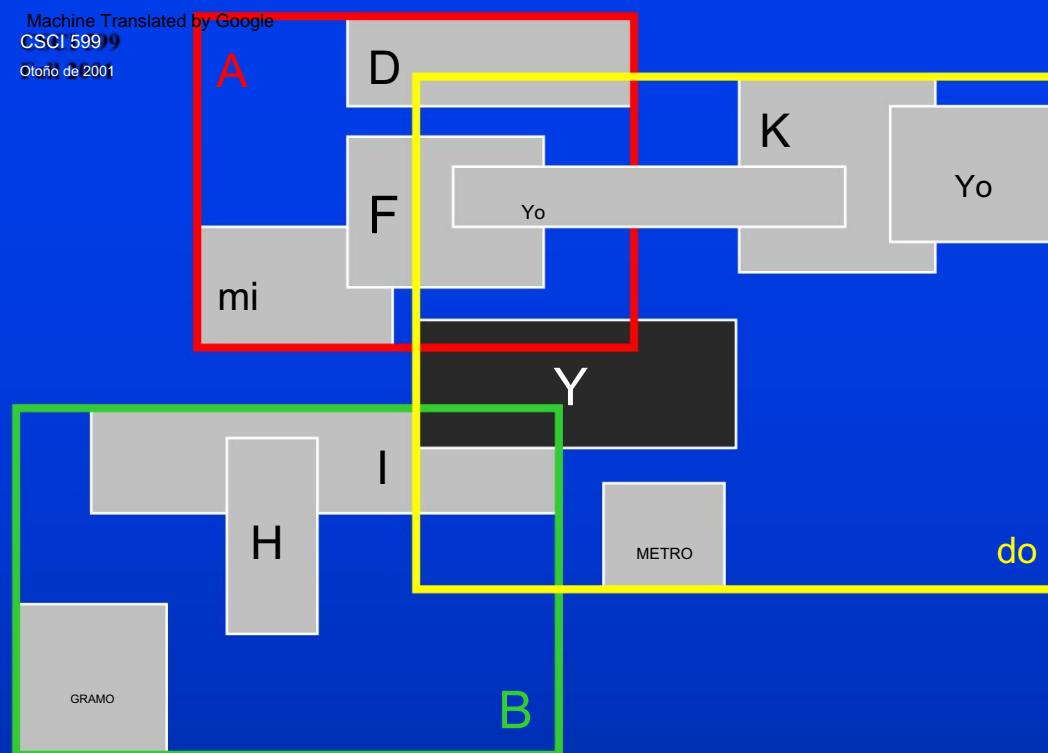
JKLM

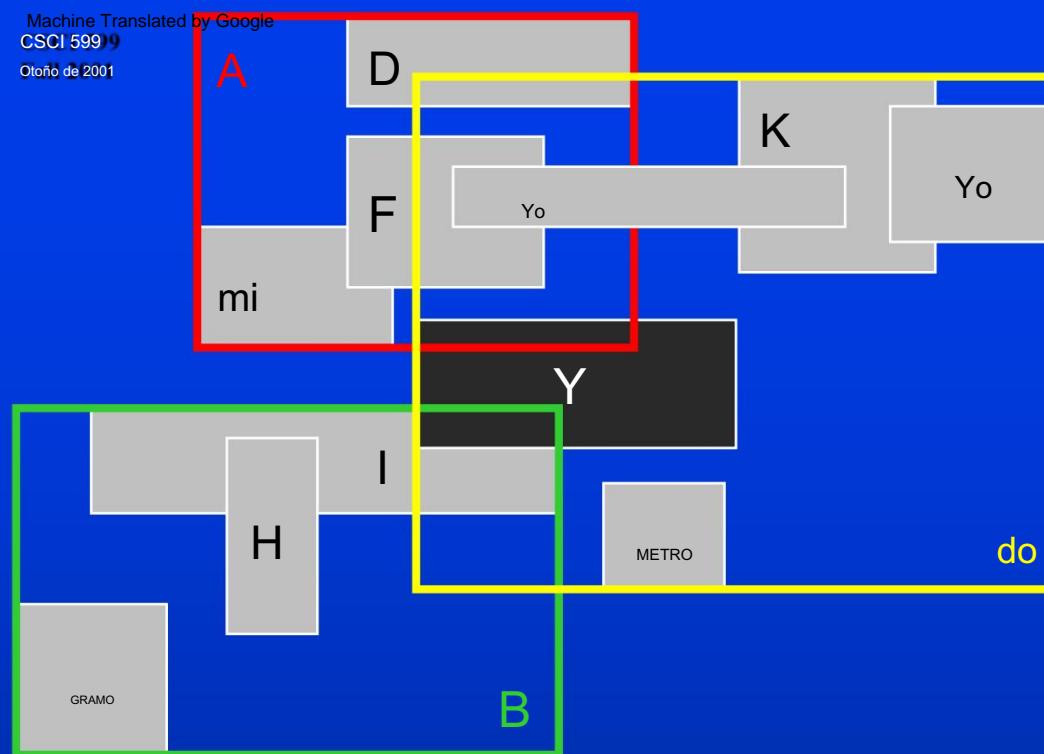


DEF

GHI

JKLM





abecedario

N = Hoja

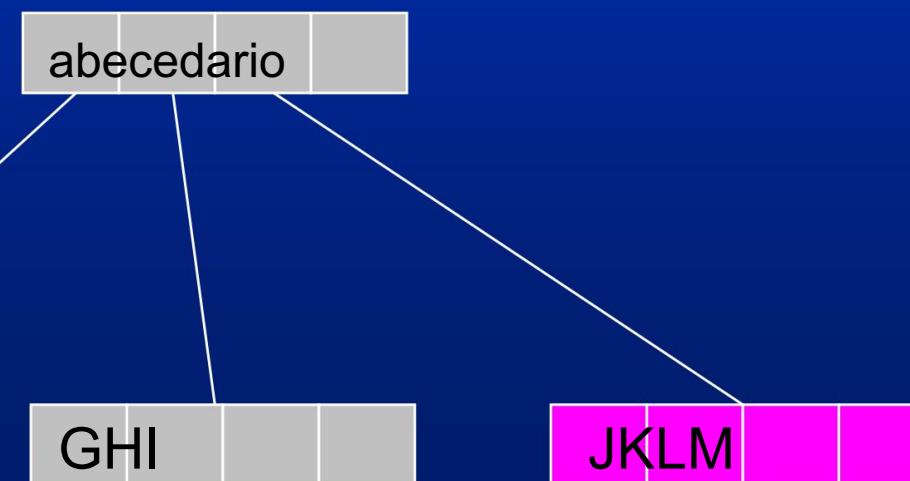
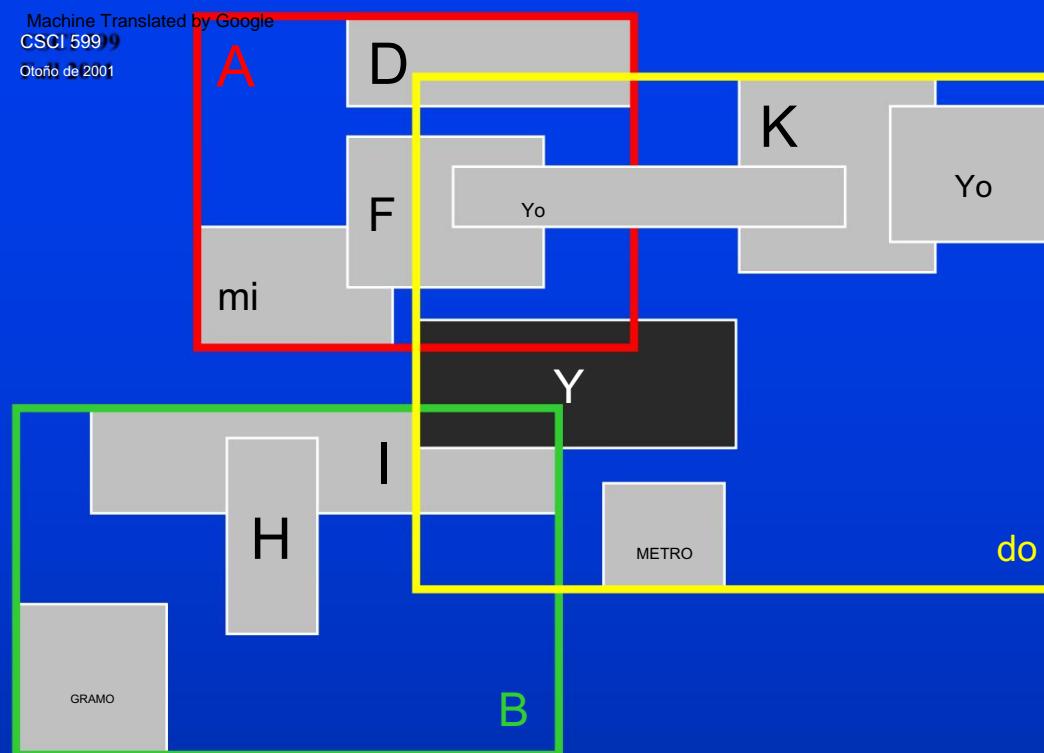
DEF

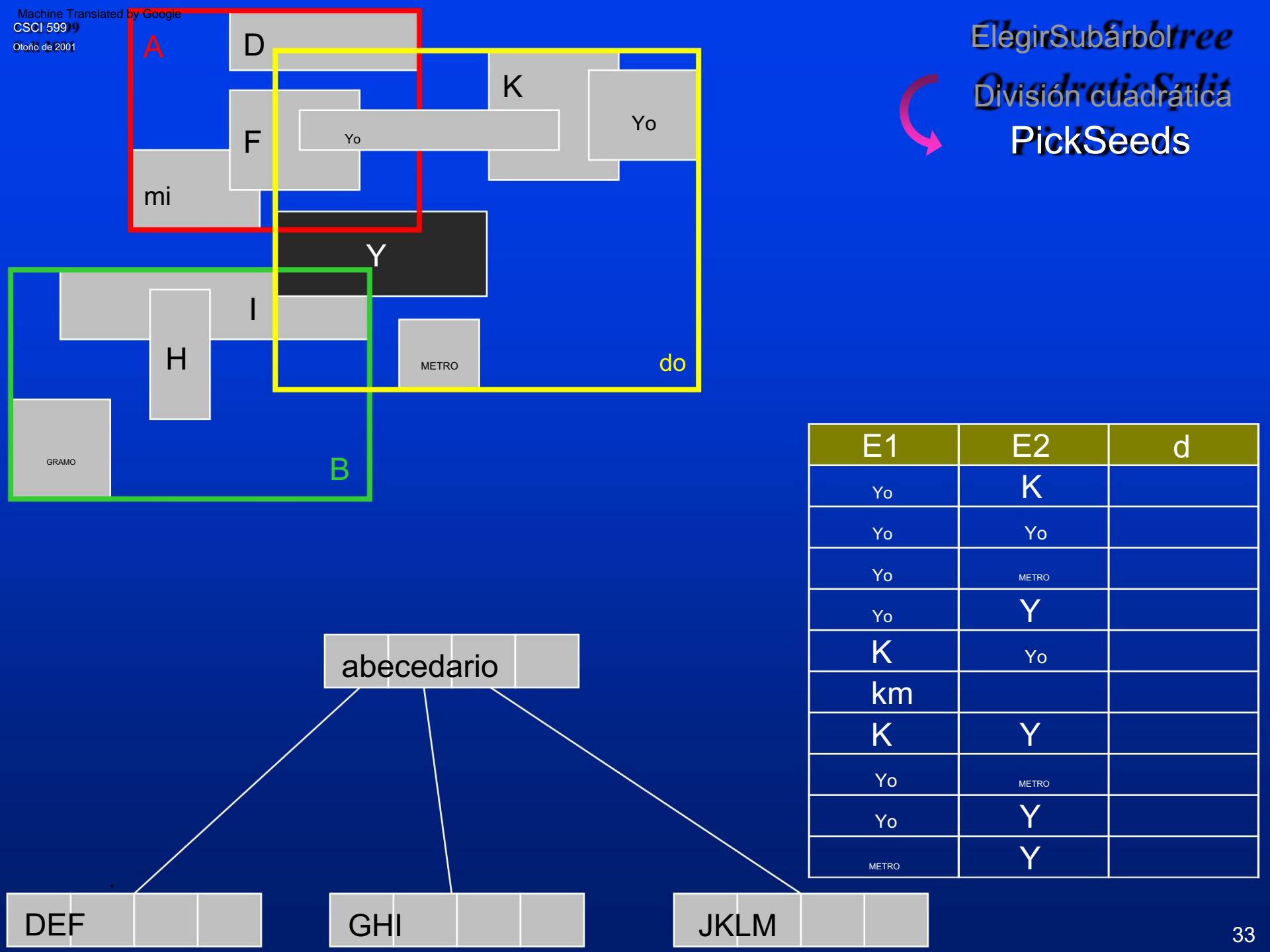
GHI

JKLM

El nodo está lleno

ElegirSubárbol
QuadraticSplit
División cuadrática



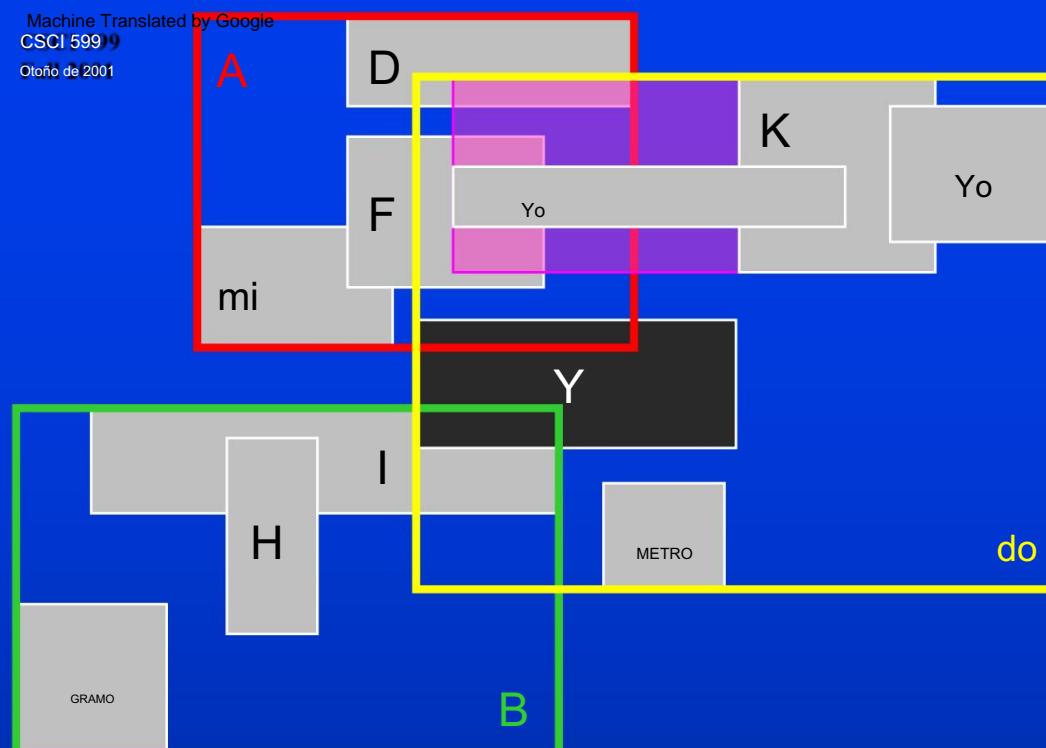


ElegirSubárbol

QuadraticSplit

División cuadrática

PickSeeds



abecedario

JKLM

GHI

DEF

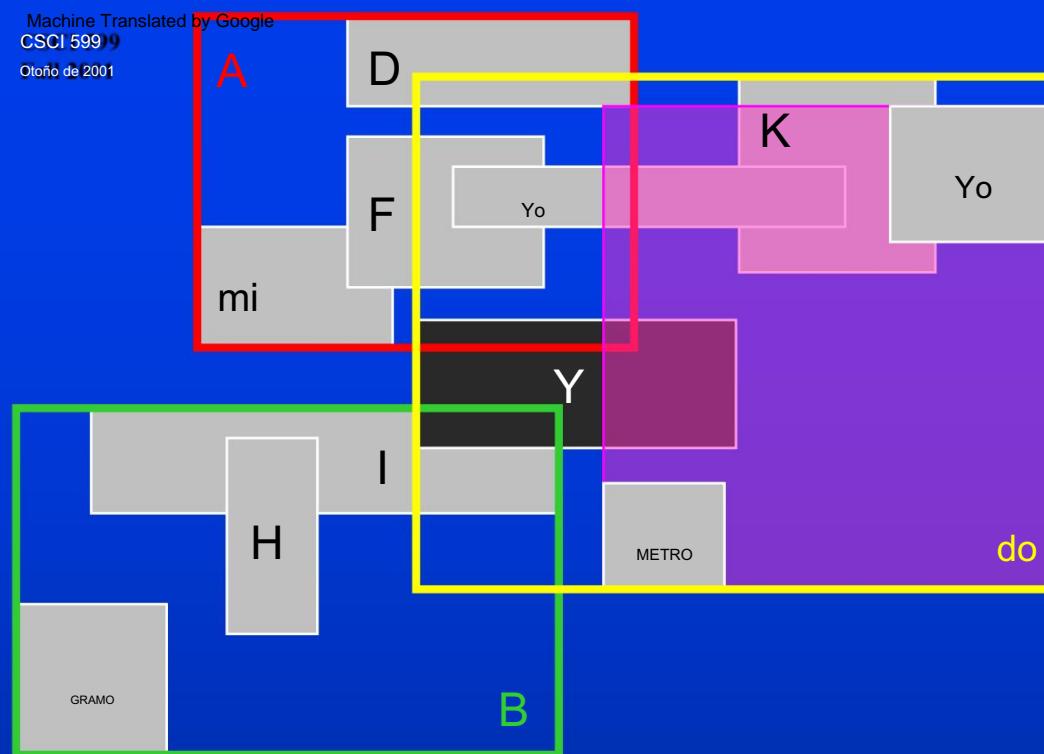
E1	E2	d
Yo	K 1.0	
Yo	Yo	
Yo	METRO	
Yo	Y	
K	Yo	
km		
K	Y	
Yo	METRO	
Yo	Y	
METRO	Y	

ElegirSubárbol

QuadraticSplit

División cuadrática

PickSeeds



abecedario

DEF

GHI

JKLM

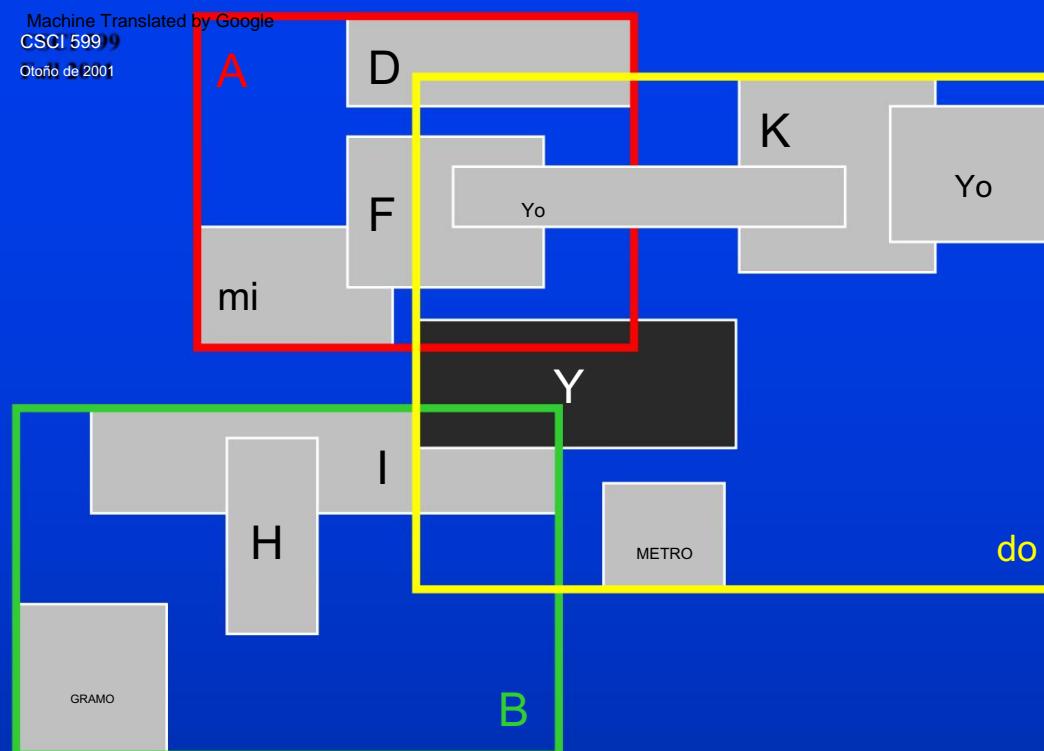
E1	E2	d
K 1.0	Yo	1.1
M 4.0	Año 1.7	
JJJJKL	KM	0.1
3.6		
KY 3.5		
LM 5.6		
Año académico 4.7		
MI 0.9		

ElegirSubárbol

QuadraticSplit

División cuadrática

PickSeeds



G1	G2
Yo	METRO

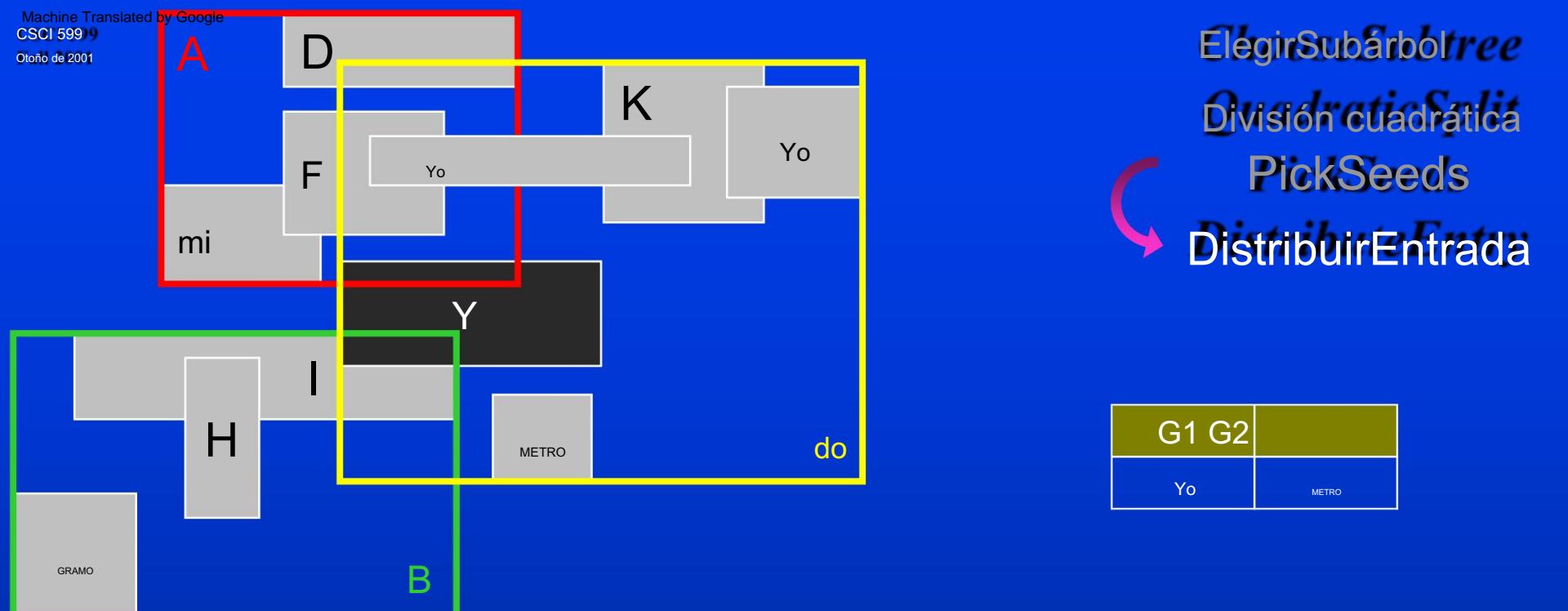
E1	E2	d
Yo	K	1.0
Yo	Yo	1.1
Yo	M 4.0	
Yo	Y	1.7
KLKM	3.6	0.1
KY	3.5	
Año académico	4.7	
MI	0.9	

abecedario

DEF

GHI

JKLM



ElegirSubárbol
QuadraticSplit
División cuadrática
PickSeeds
DistribuirEntrada

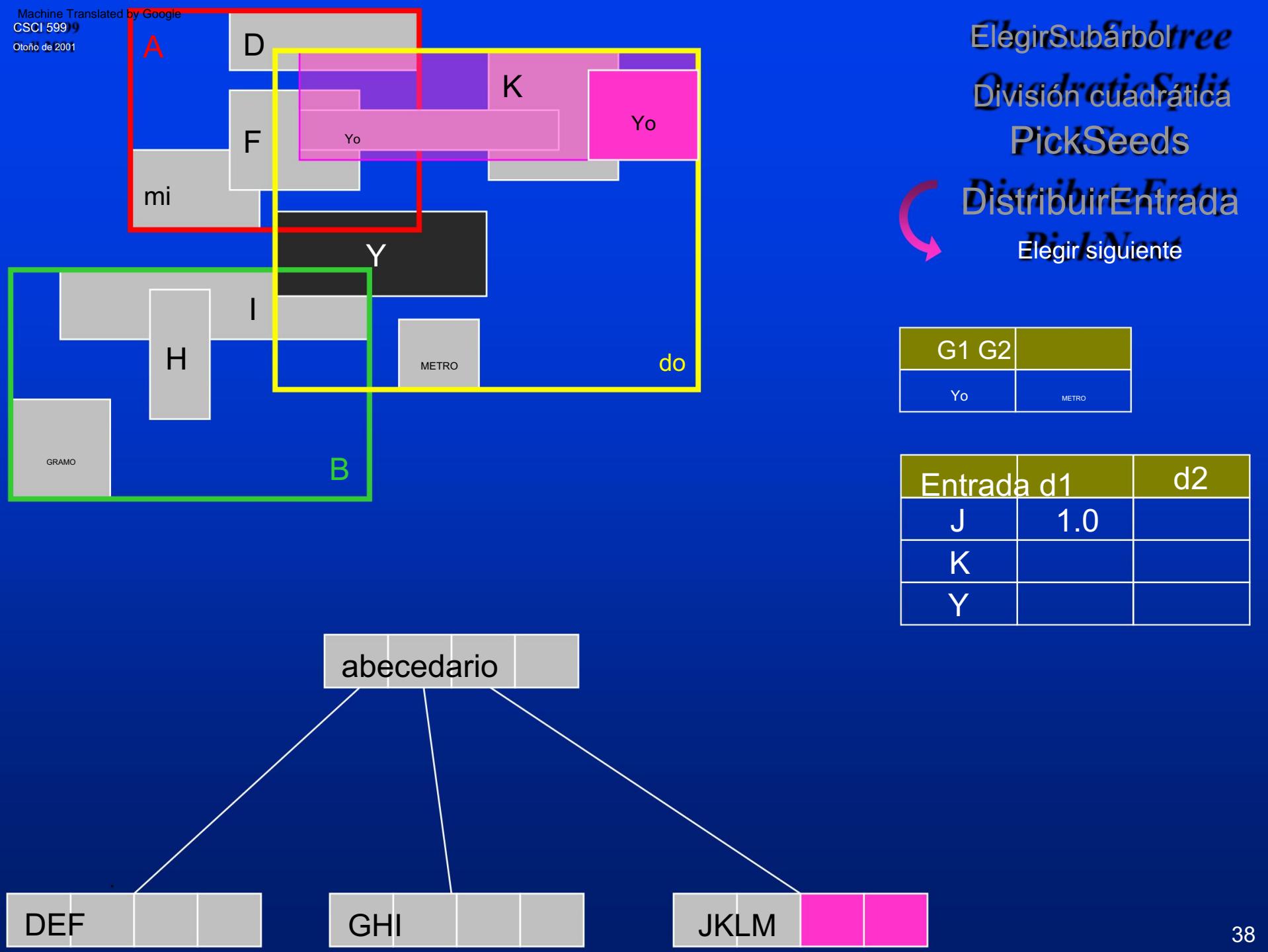
G1 G2	
Yo	METRO

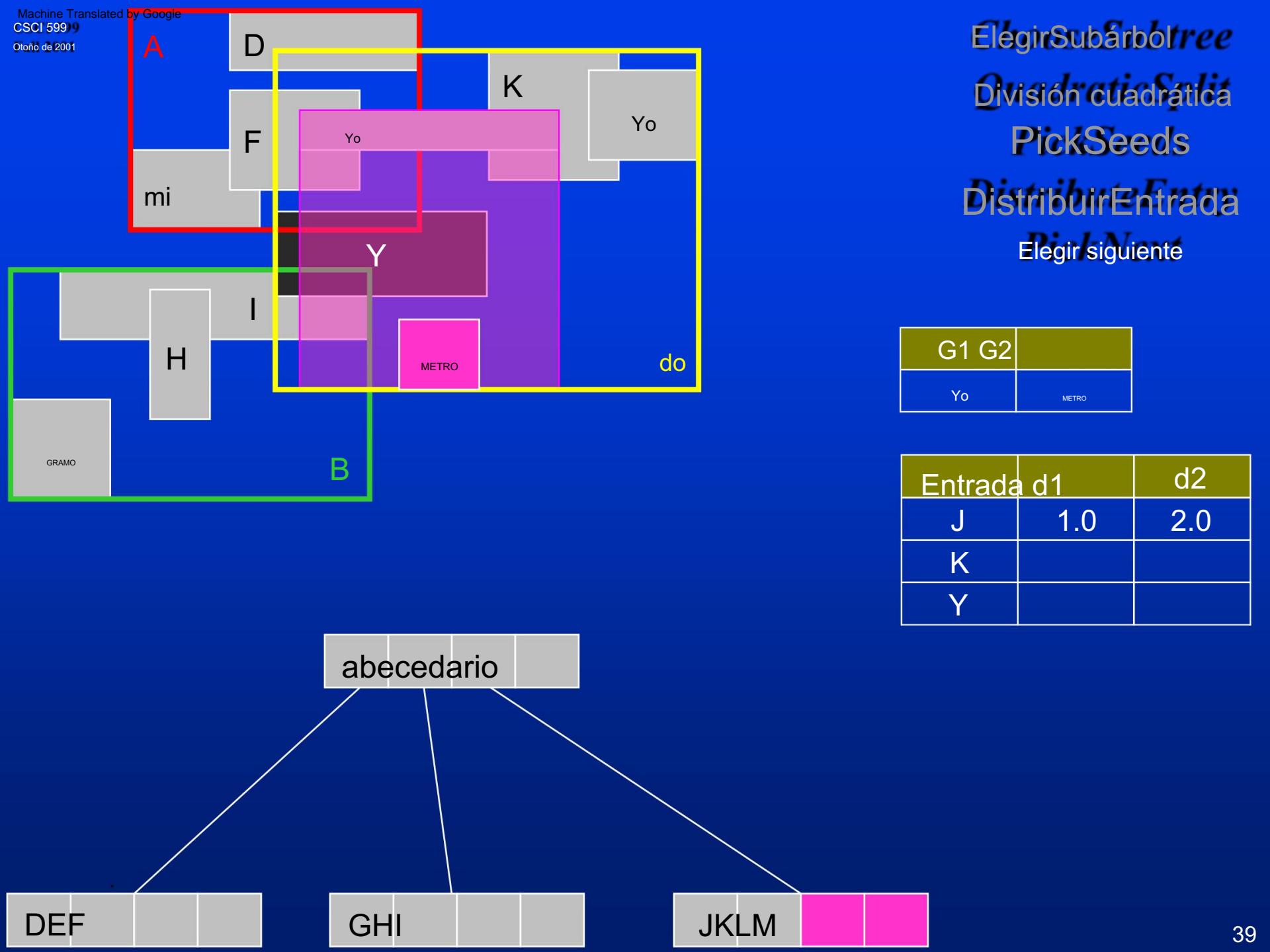


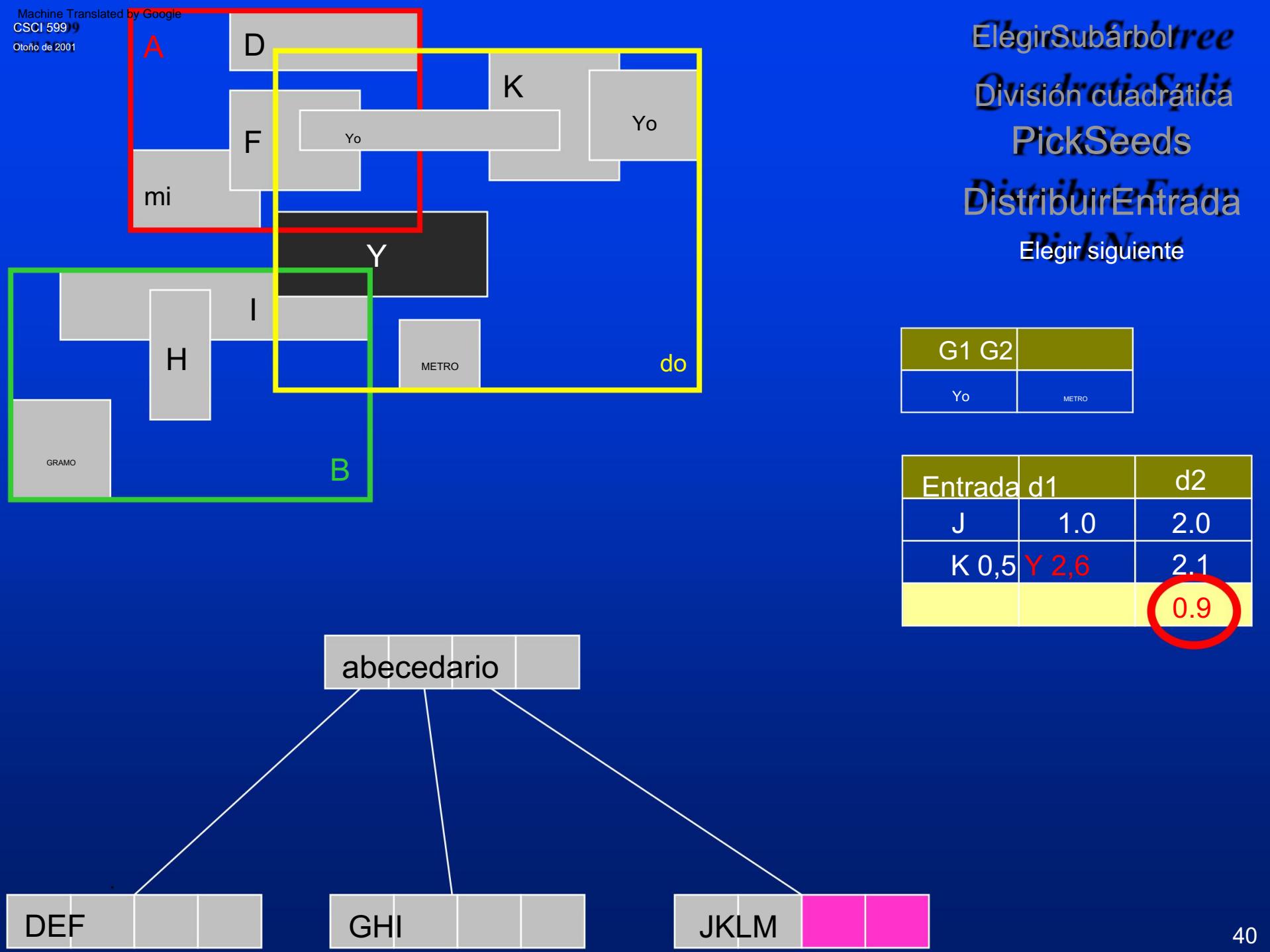
DEF

GHI

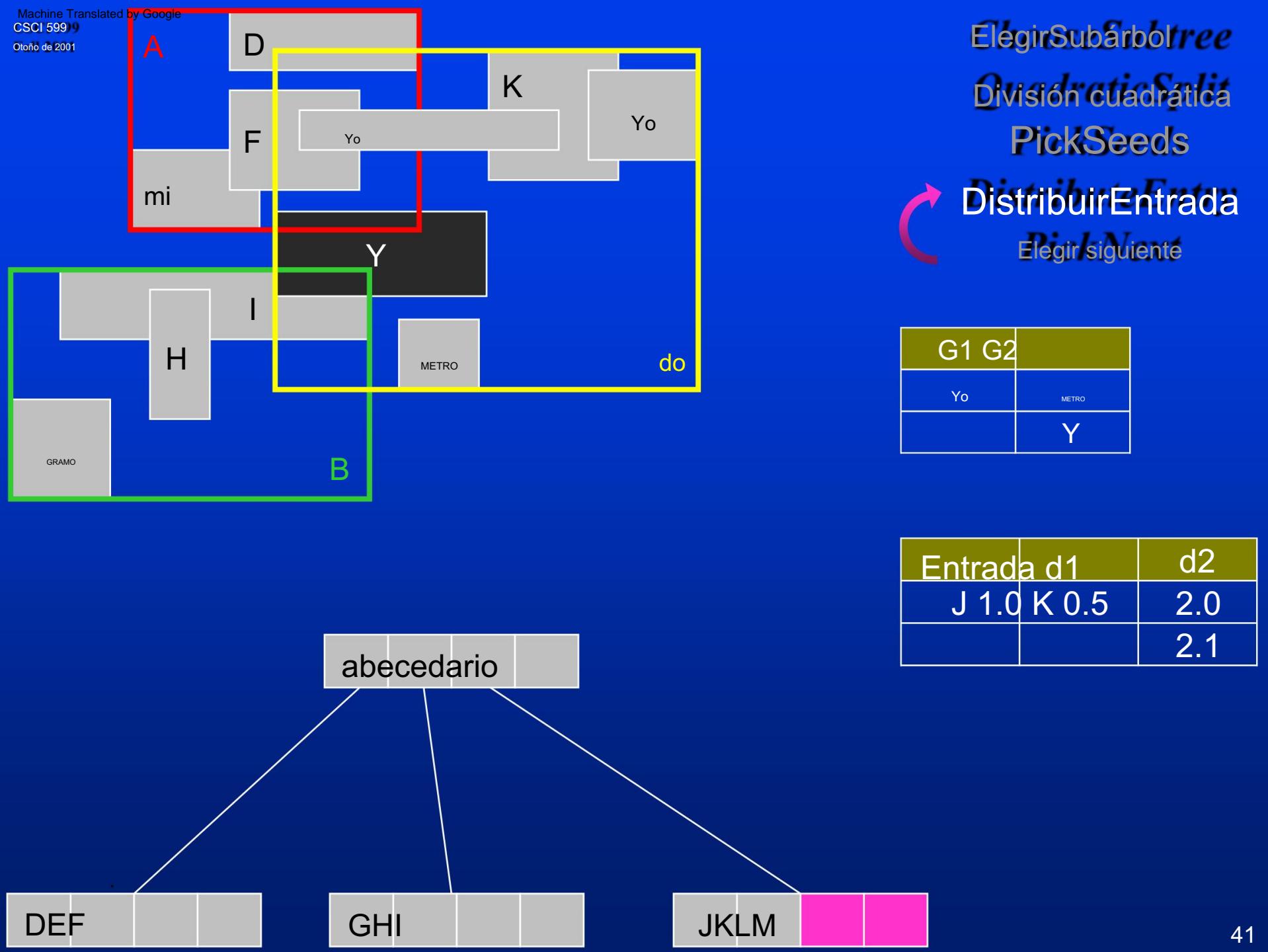
JKLM







ElegirSubárbol
 QuadraticSplit
 División cuadrática
 PickSeeds
 DistribuirEntrada
 PickNext
 Elegir siguiente



ElegirSubárbol

QuadraticSplit

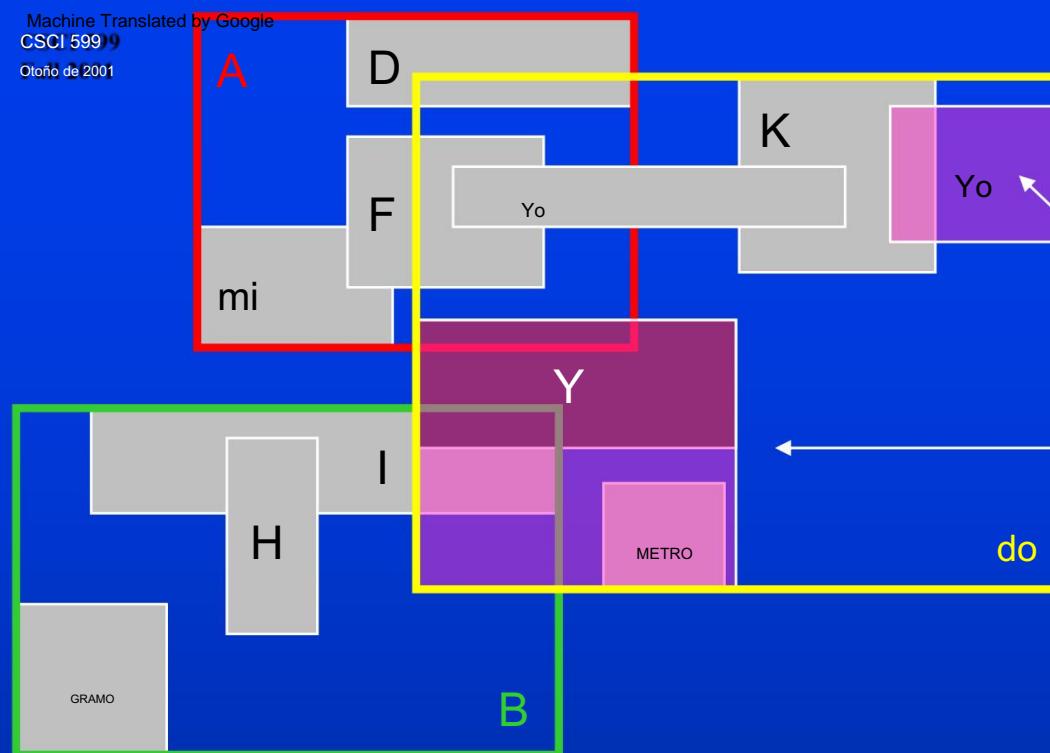
PickSeeds

DistribuirEntrada

PickNext

G1	G2
Yo	METRO
	Y

ElegirSubárbol
 QuadraticSplit
 División cuadrática
 PickSeeds
 DistribuirEntrada
 PickNext
 Elegir siguiente



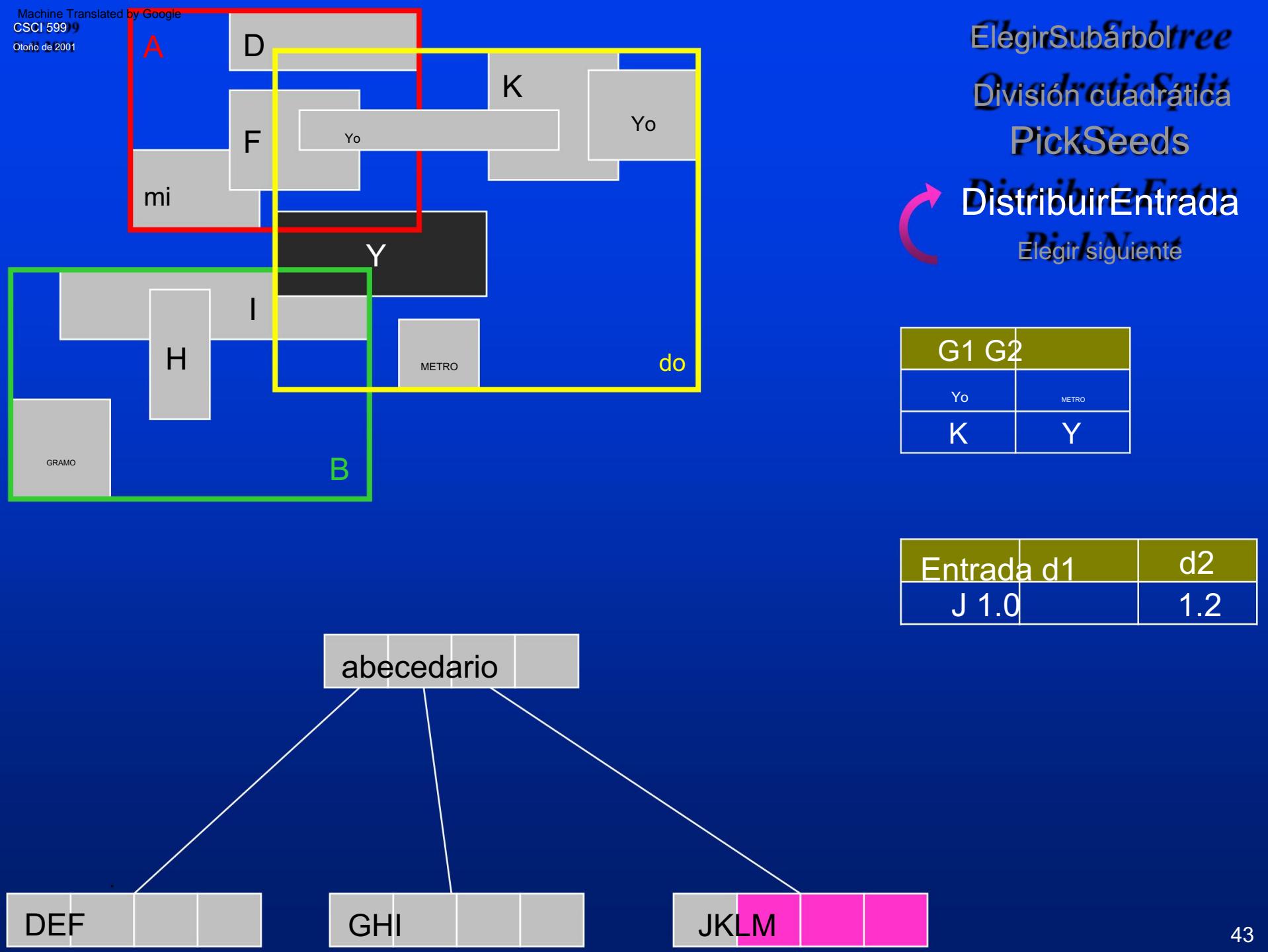
Entrada d1	d2
J 1.0 K 0.5	1.2
	2.3

abecedario

DEF

GHI

JKLM



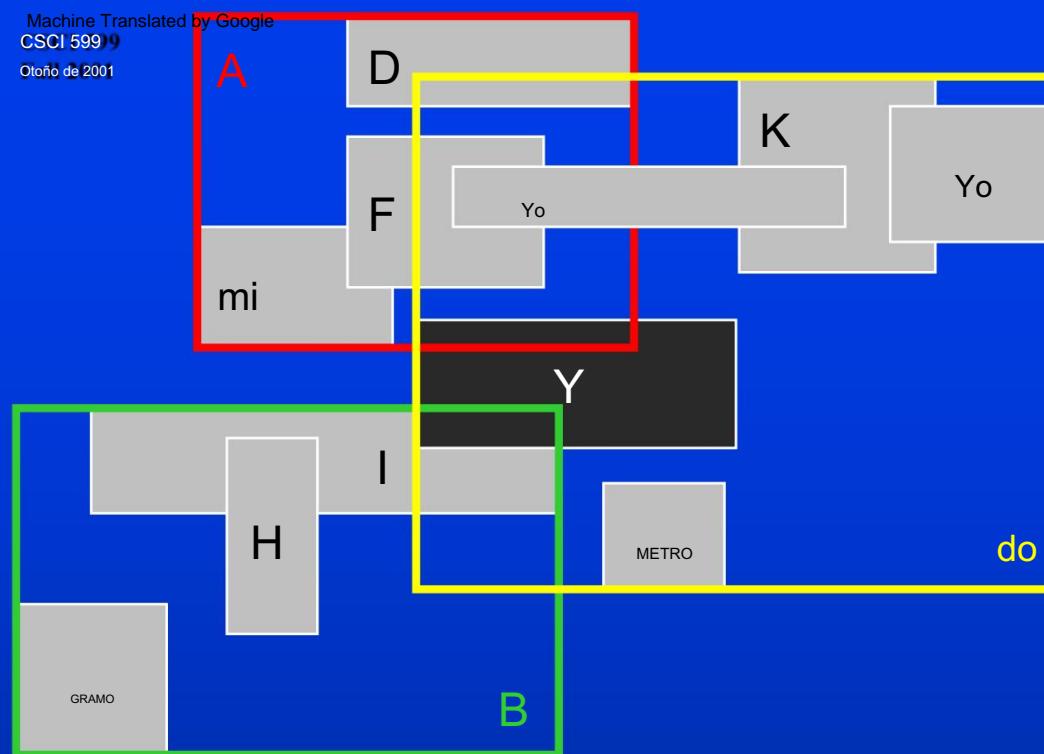
ElegirSubárbol
 QuadraticSplit
 División cuadrática
 PickSeeds
 DistribuirEntrada
 PickNext
 Elegir siguiente



G1	G2
Yo	METRO
K	Y

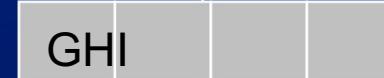
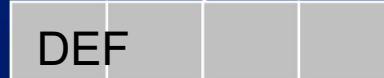
Entrada d1	d2
J 0.7	1.2

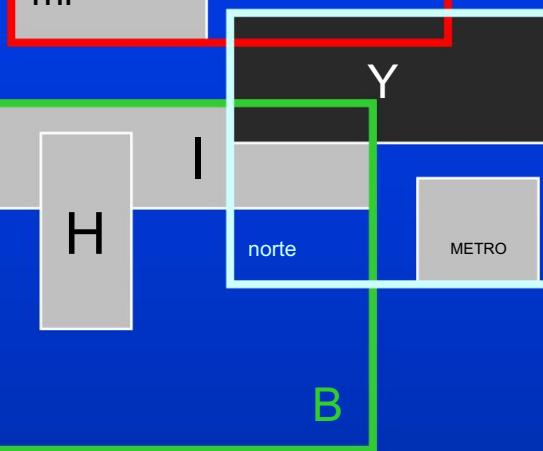
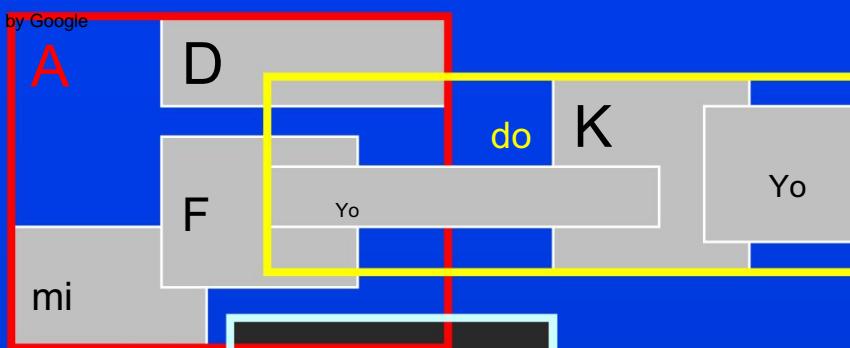




ElegirSubárbol
QuadraticSplit
División cuadrática
PickSeeds
DistribuirEntrada
PickNext
Elegir siguiente

G1	G2
Yo	METRO
K	Y
Yo	





ElecciónSubárbol
 QuadraticSplit
 División cuadrática
 PickSeeds
 DistribuirEntrada
 PickNext
 Elegir siguiente

G1	G2
Yo	METRO
K	Y
Yo	



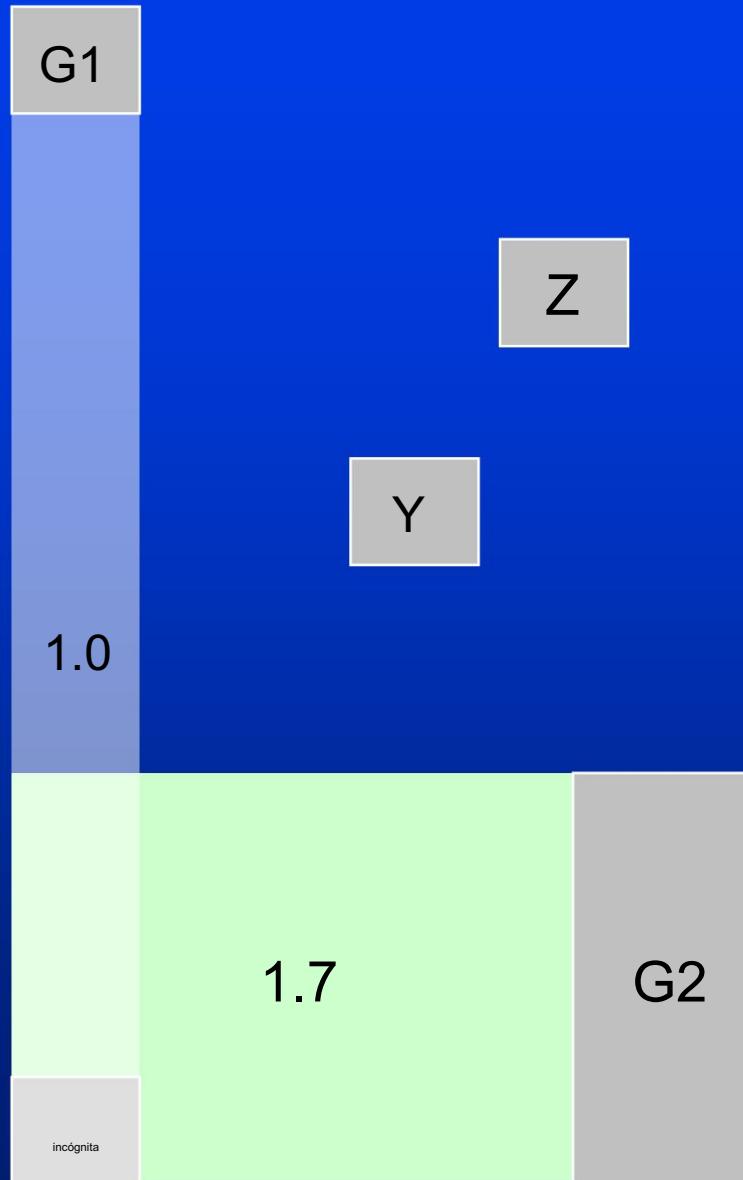
DEF

GHI

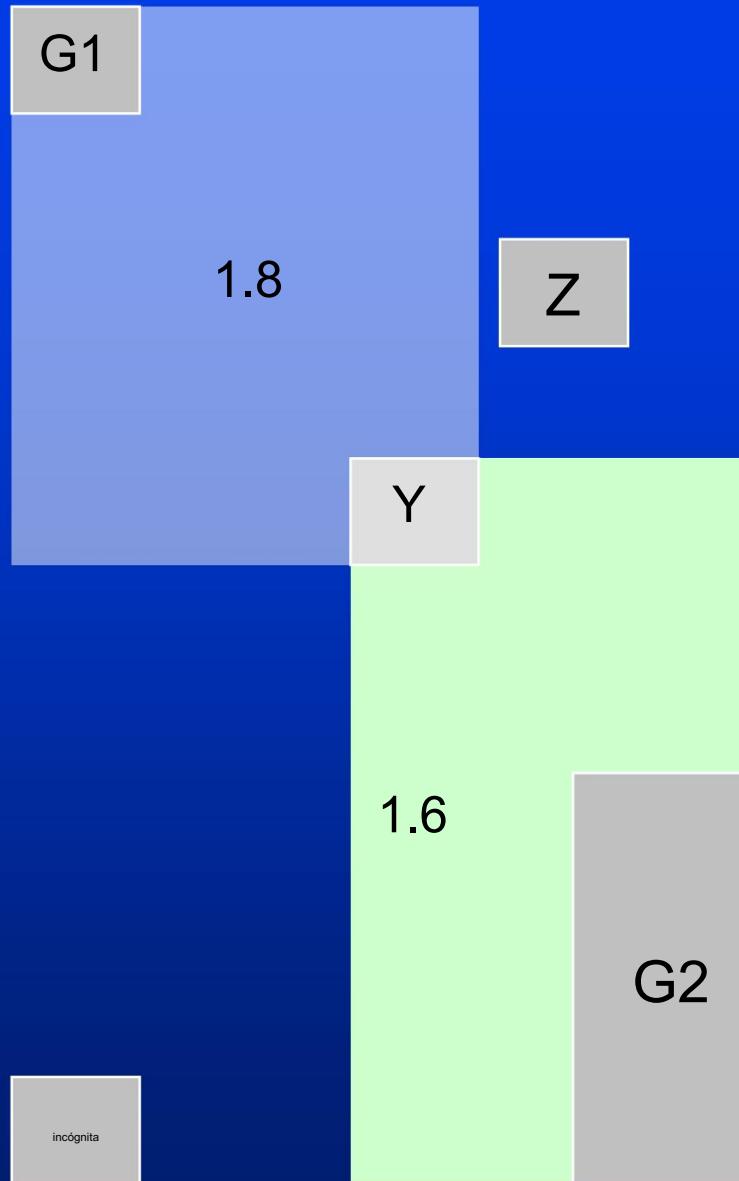
JKL

MI

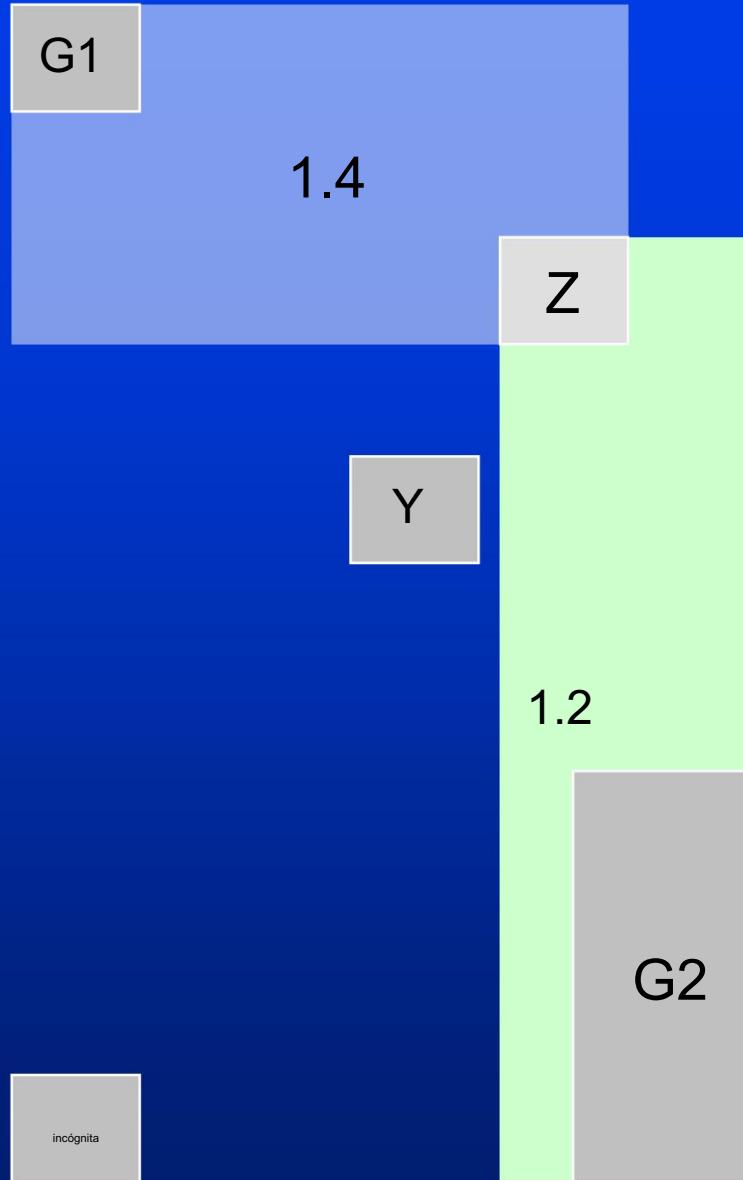
Problema n.º 1: semillas pequeñas



Problema n.º 1: semillas pequeñas



Problema n.º 1: semillas pequeñas



X se asigna a G1.
Pero la distancia es grande
y, por lo tanto, una mala
división. X se asignará a G2
en su lugar.

Problema n.^o 1: semillas pequeñas

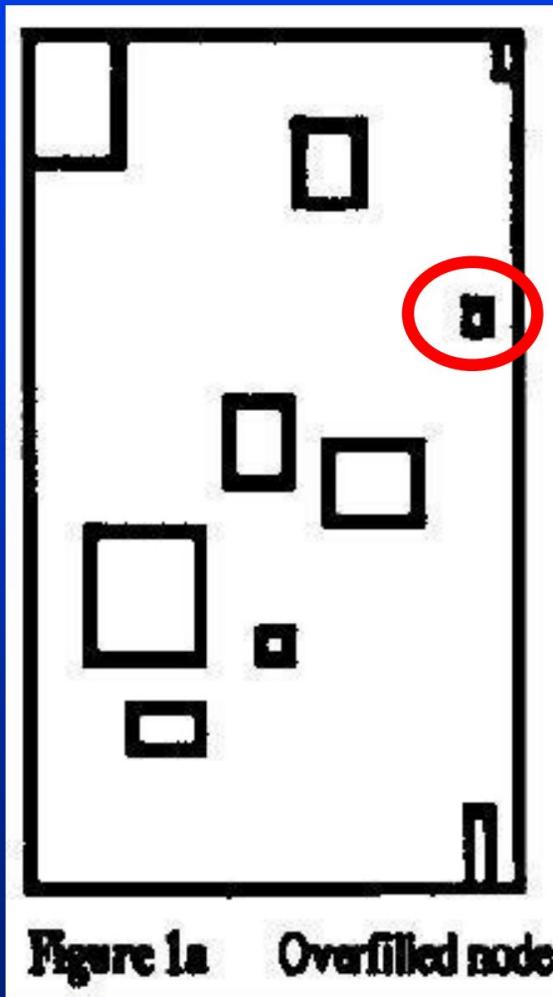


Figure 1a Overfilled node

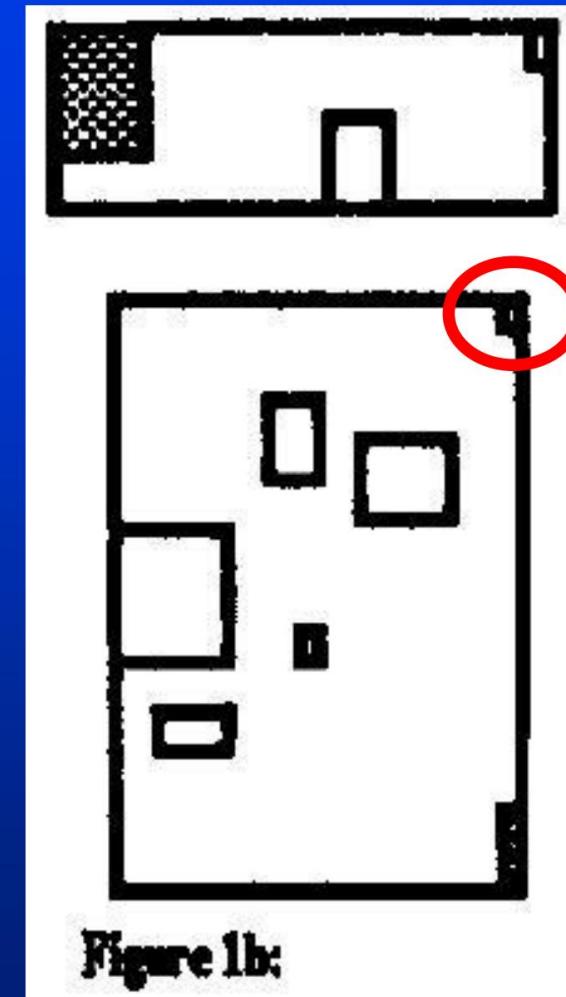
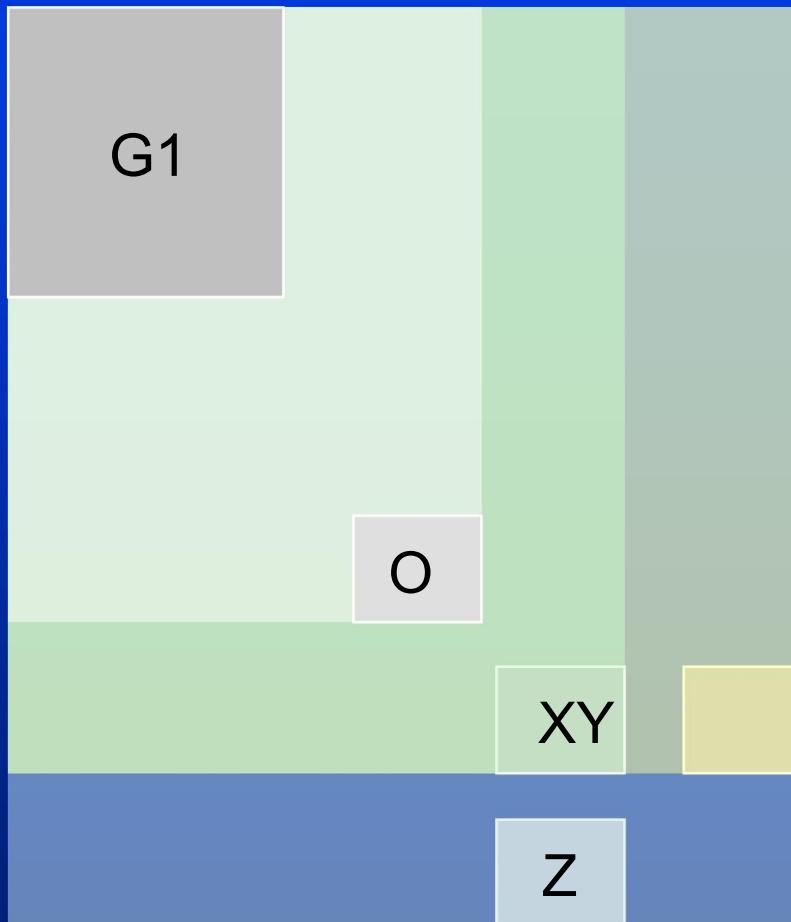


Figure 1b:
Split of the quadratic
R-tree, m = 30%

Problema n.º 2: Preferir el rectángulo delimitador



Asignar un rectángulo a una semilla

Área ampliada

Se necesita una ampliación de área menor para incluir la siguiente entrada

Ampliar de nuevo

Continúa.....

G2

Problema n.º 2: Preferir el rectángulo delimitador

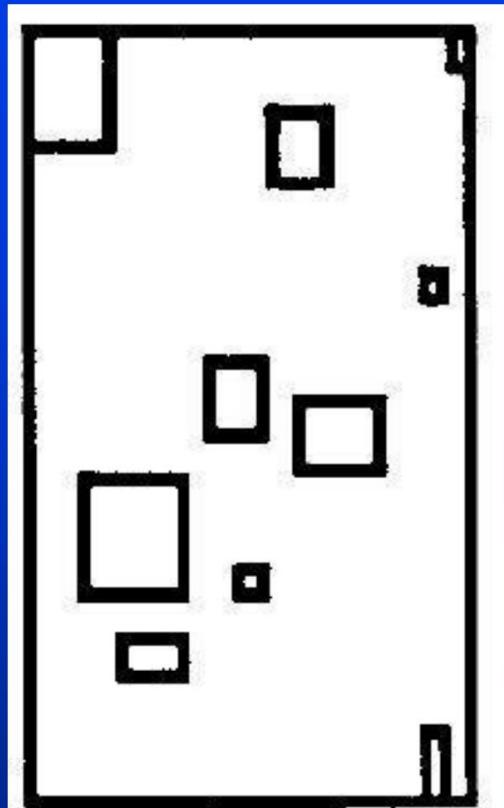


Figure 1a Overfilled node

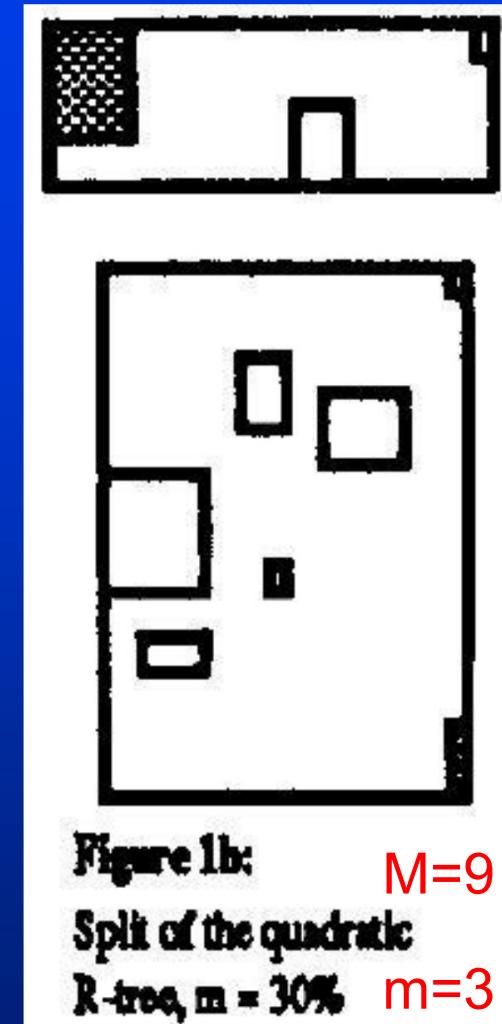


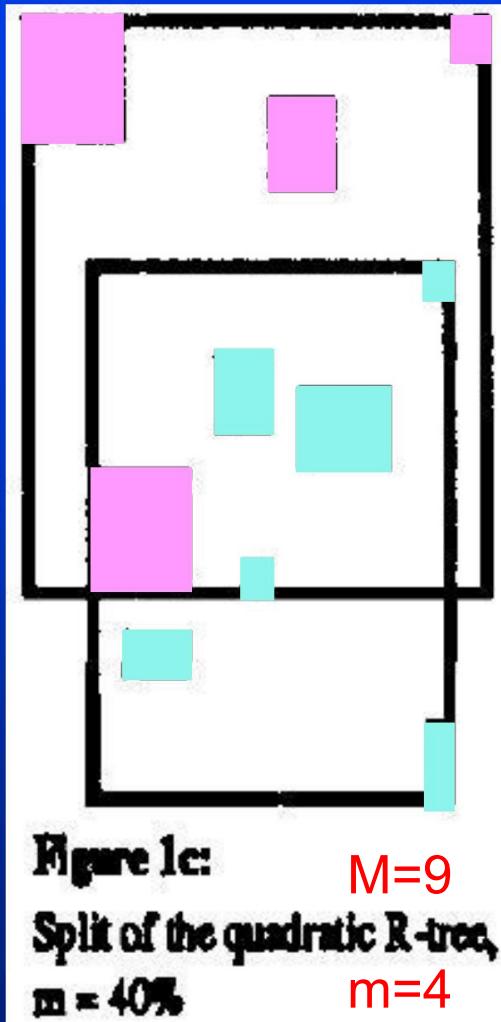
Figure 1b:
 $M=9$
Split of the quadratic
R-tree, $m = 30\%$ $m=3$

$$M - m + 1 = 7$$

Distribución desigual

Reducir la utilización del almacenamiento

Problema #3



Si se alcanza $\tilde{M}m+1$, todos
Las m entradas restantes se
asignan al otro grupo sin
tener en cuenta las propiedades
geométricas

Mejor rendimiento de recuperación
cuando $m = 40\%$

$$M - m + 1 = 6$$

Gran superposición

Aumentar el número de aths a recorrer

Variantes del árbol R - Greene

Algoritmo de división alternativo

Utilice el algoritmo ChooseSubtree de Guttman

El único criterio geométrico: elección del eje dividido

Se necesitan más criterios de optimización geométrica para mejorar el rendimiento de la recuperación

Es posible que no se encuentre el eje “correcto” y que se produzca una división incorrecta

Algoritmo de división de Greene

[Dividir un conjunto de entradas $M+1$ en dos grupos]

GS1 Invoca ChooseAxis para determinar el eje perpendicular al cual se realizará la división

GS2 Invocar Distribuir

Variantes del árbol R - Greene

Algoritmo ChooseAxis

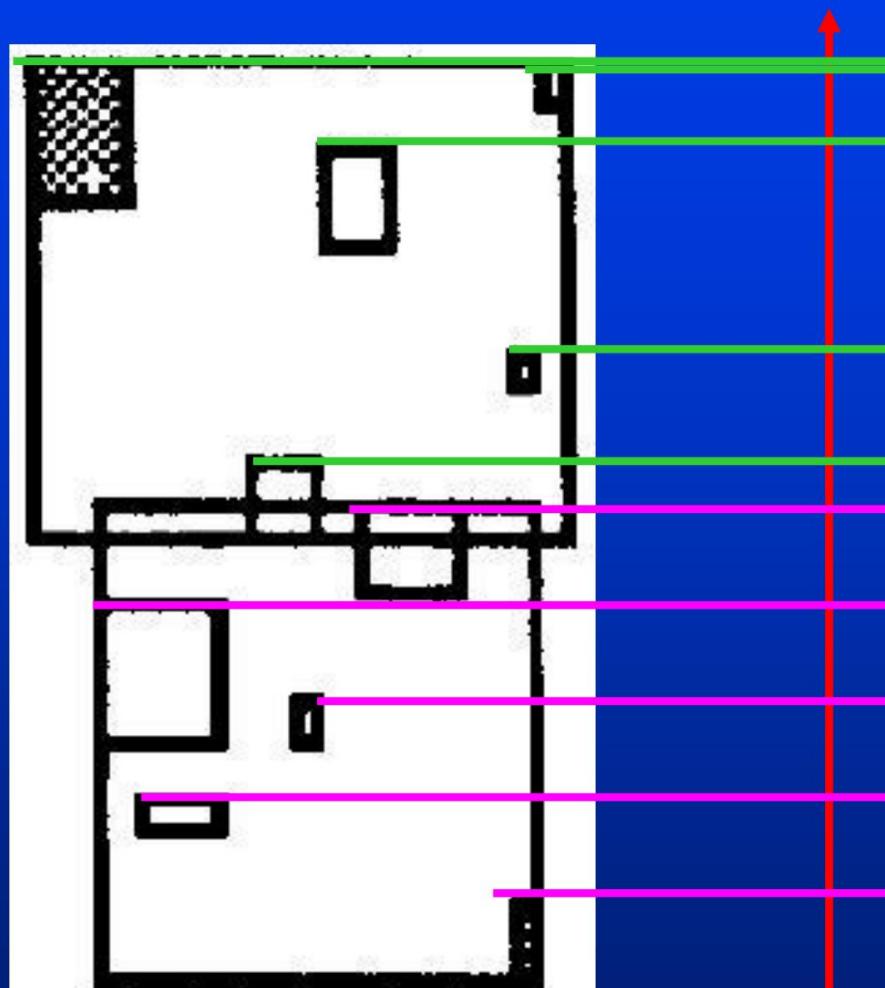
[Dividir un conjunto de entradas $M+1$ en dos grupos]

- CA1 Invoca **PickSeeds** para encontrar los dos rectángulos más distantes del nodo actual
- CA2 Para cada eje registre la separación de las dos semillas
- CA3 Normaliza las separaciones dividiéndolas por la longitud de los nodos que encierran el rectángulo a lo largo del eje apropiado
- CA4 Devuelve el eje con la mayor separación normalizada

Algoritmo Distribuir

- D1 Ordenar las entradas por el valor bajo de sus rectángulos a lo largo del eje elegido
- D2 Asignar las primeras $(M+1) / 2$ entradas a un grupo, las últimas $(M+1) / 2$ entradas al otro
- D3 Si $M+1$ es impar, entonces asigne la entrada restante al grupo cuyo rectángulo circundante se incrementará menos con su adición.

Variantes del árbol R - Greene



$$M = 9$$

$$(M + 1) \text{ div } 2 = 5$$

Figure 1d
Greene's split

Árbol R*

Los árboles R anteriores solo consideran el parámetro de área al elegir la ruta de inserción

El árbol R* considera el área, el margen y la superposición en diferentes combinaciones

Algoritmo ChooseSubtree

CS1 Establezca N como nodo raíz

CS2 Si N es una hoja,

devolver N

demás

Si los punteros secundarios en N apuntan a hojas

[Determinar el costo mínimo de superposición]

Seleccione la entrada en N cuyo rectángulo necesita la menor ampliación de superposición para incluir el nuevo rectángulo de datos. Resuelva los empates eligiendo la entrada cuyo rectángulo necesita la menor ampliación de área y luego la entrada con el rectángulo de área más pequeña.

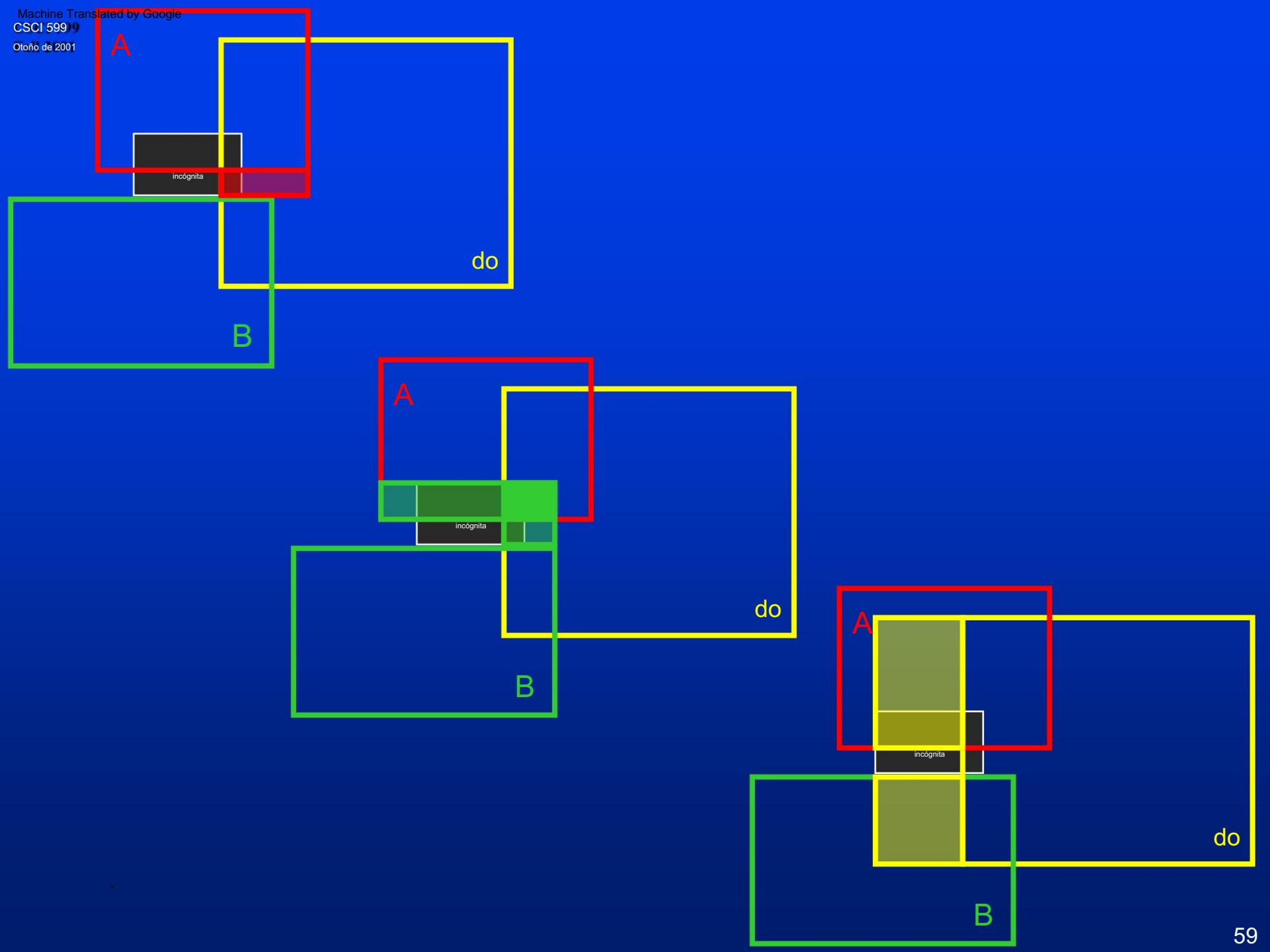
demás

[determinar el costo mínimo del área]

Seleccione la entrada en N cuyo rectángulo requiera la menor ampliación de área para incluir el nuevo rectángulo de datos. Resuelva los empates seleccionando la entrada con el rectángulo de menor área.

fin

CS3 Establezca N como el nodo secundario al que apunta el puntero secundario de la entrada seleccionada. Repetir desde CS2.



Algoritmo ChooseSubtree

Para elegir el mejor nodo hoja, minimizar la superposición tuvo un rendimiento ligeramente mejor (una superposición menor significa una menor cantidad de rutas y, por lo tanto, un mayor rendimiento)

El costo de la CPU para determinar la superposición es cuadrático en el número de entradas

Para tamaños de nodos grandes, reduzca el cálculo modificando:

[Determinar el coste de superposición casi mínimo]

Ordena los rectángulos en N en orden creciente de su área
Ampliación necesaria para incluir el nuevo rectángulo de datos.

Sea A el grupo de las primeras p entradas. De las entradas en A , considerando todas las entradas en N , elija la entrada cuyo rectángulo requiera la menor ampliación por superposición . Resuelva los empates como se describió anteriormente.

Algoritmo ChooseSubtree

Para 2-D, $p = 32$: casi ninguna reducción del rendimiento de recuperación en comparación con el modelo no modificado

El costo de la CPU sigue siendo más alto que el ChooseSubtree original

Mejorar el rendimiento de recuperación, particularmente en consultas con pequeños rectángulos de consulta en archivos de datos con pequeños rectángulos o puntos distribuidos de manera no uniforme.

Mejorar la robustez

División del árbol R*

A lo largo de cada eje

Las entradas se ordenan primero por el valor más bajo de sus rectángulos, luego ordenado por valor superior

determinar $(\tilde{M}2m+2)$ distribuciones de $(M+1)$ entradas en dos grupos para cada tipo

Distribución #	Grupo 1	1 1 2 3	Grupo 2
	metro		$M - m + 1$
	$m + 1$		$M-m$
	$m + 2$		$M-m-1$
.....
$M - 2 m + 2$	$M - m + 1$	m	

p. ej. $M = 9$, $m = 4$, 10 entradas

Distribución #	Grupo 1	Grupo 2
1	4	6
2	5	5
3	6	4



Figure 1a Overfilled node

Ordenar #1

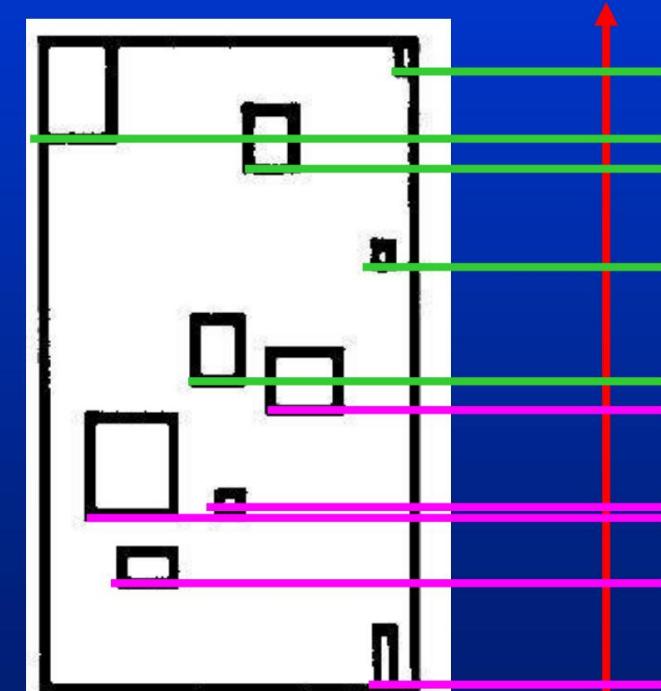


Figure 1a Overfilled node

Ordenar #2

División del árbol R*

Determinar valores de bondad para cada distribución

tres valores de bondad

- área-valor $\text{área}[\text{bb}(\text{primer grupo})] + \text{área}[\text{bb}(\text{segundo grupo})]$
- valor de margen $\text{margen}[\text{bb}(\text{primer grupo})] + \text{margen}[\text{bb}(\text{segundo grupo})]$
- área de valores superpuestos $\text{superpuestos}[\text{bb}(\text{primer grupo}) \text{ y } \text{bb}(\text{segundo grupo})]$

depende de los valores de bondad, la distribución final está determinada

Algorithm Split

- S1 Invoque ChooseSplitAxis para determinar el eje perpendicular al cual se realiza la división
- S2 Invoque ChooseSplitIndex para determinar la mejor distribución en dos grupos a lo largo de ese eje
- S3 Distribuya las entradas en dos grupos

Algorithm ChooseSplitAxis

CSA1 Para cada eje

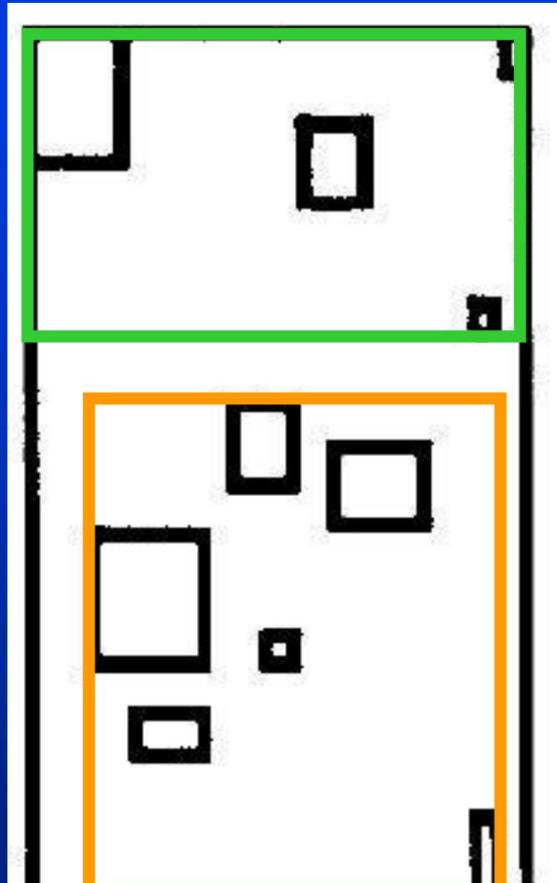
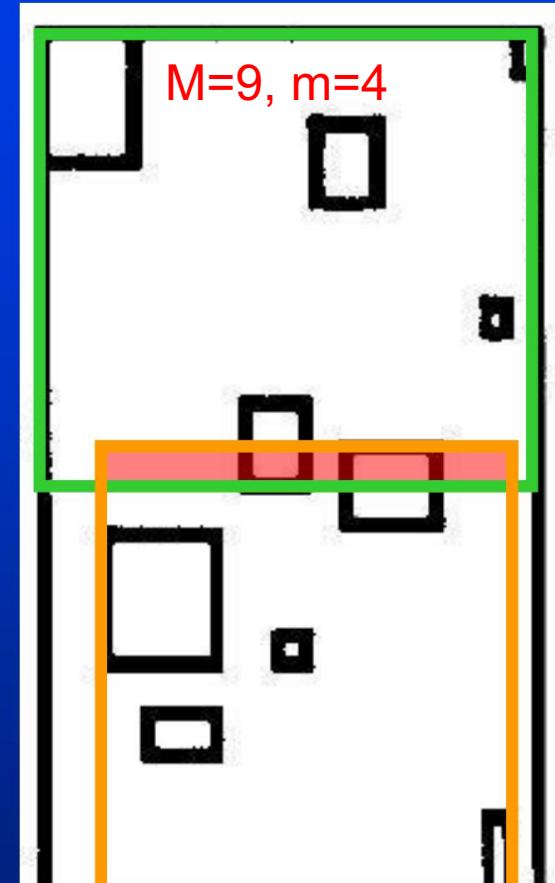
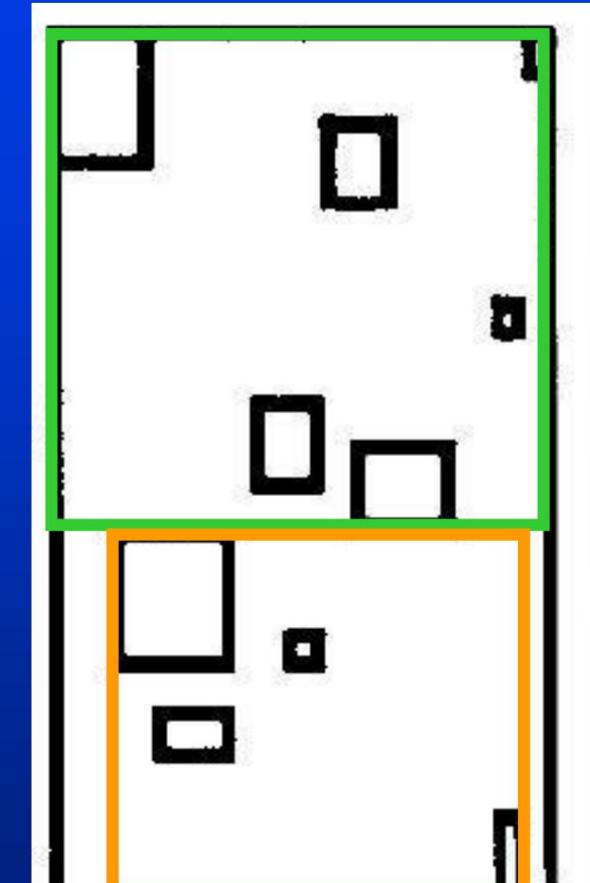
Ordene las entradas por el valor inferior y luego por el superior de sus rectángulos y determine todas las distribuciones como se describió anteriormente. Calcule S, la suma de todos los valores de margen de las diferentes distribuciones.

fin

CSA2 Elija el eje con el mínimo S como eje dividido

Algorithm ChooseSplitIndex

CSA1 A lo largo del eje de división seleccionado, elija la distribución con el mínimo valor de superposición. Resuelva los empates eligiendo la distribución con el mínimo valor de área.

Distribución #1**Figure 1a Overfilled node****Distribución #2****Figure 1a Overfilled node****Distribución #3****Figure 1a Overfilled node**

Valor del área = 1,00

Valor del área = 1,11

División del árbol R*

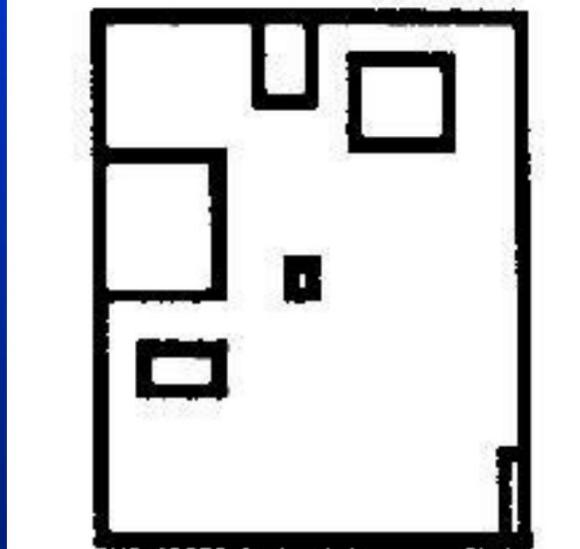
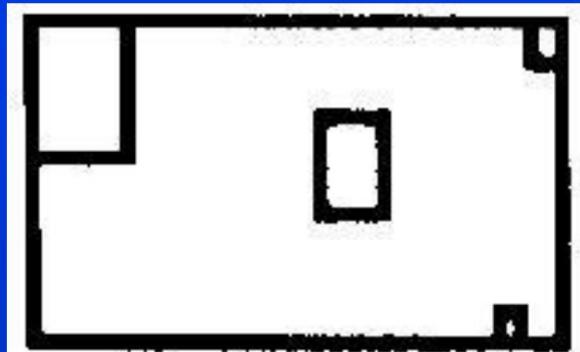


Figure 1e M=9, m=4

Split of the R*-tree, m = 40%

Pruebas con $m = 20\%, 30\%, 40\%$ y 45% de M

Mejor rendimiento cuando $m = 40\%$

Otro ejemplo

Distribución #1

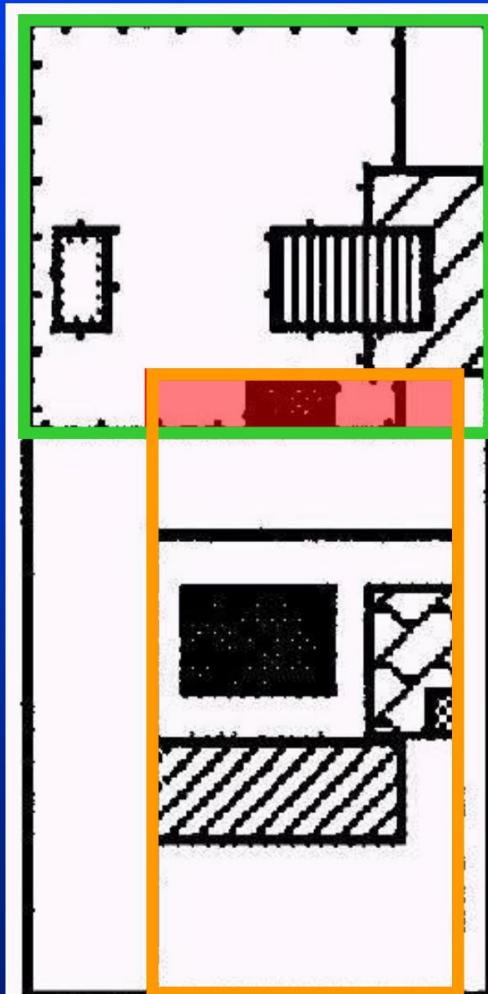


Figure 2a Overfilled node

Distribución #2

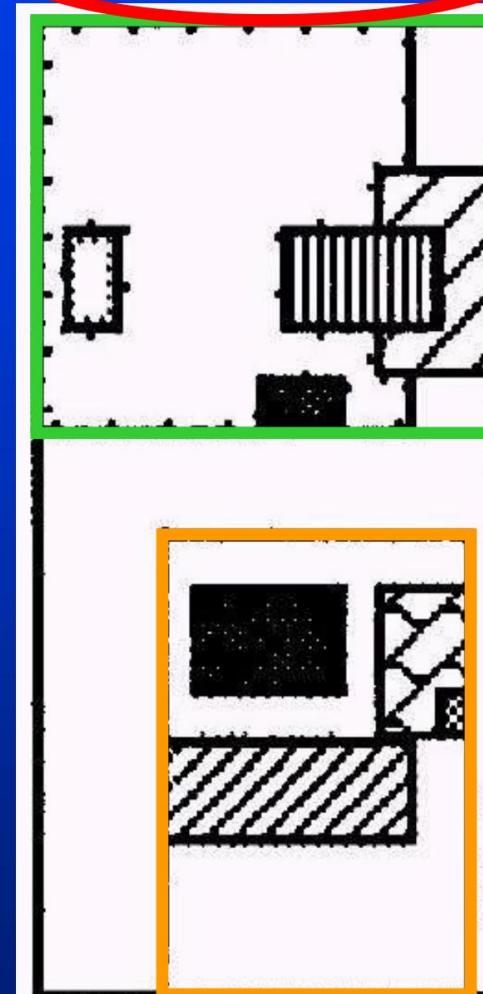


Figure 2a Overfilled node

Distribución #3

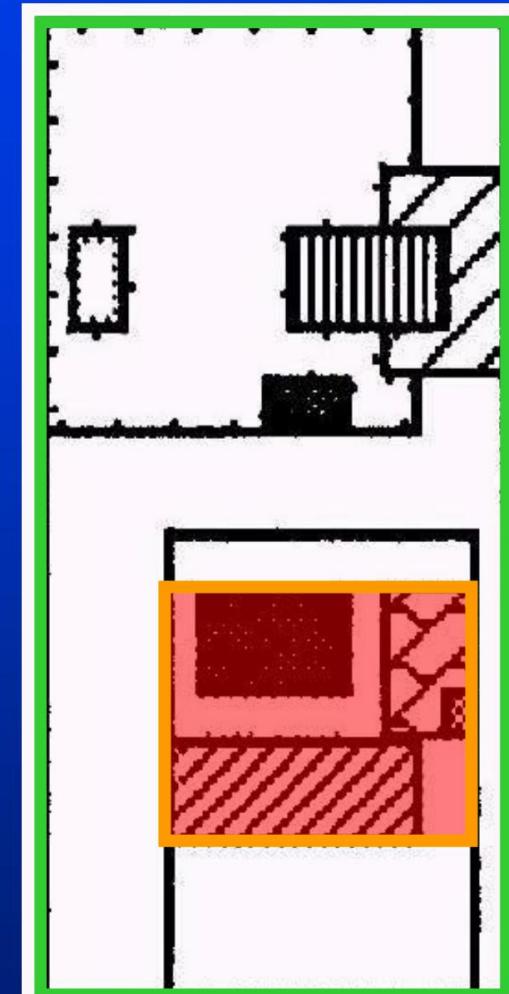
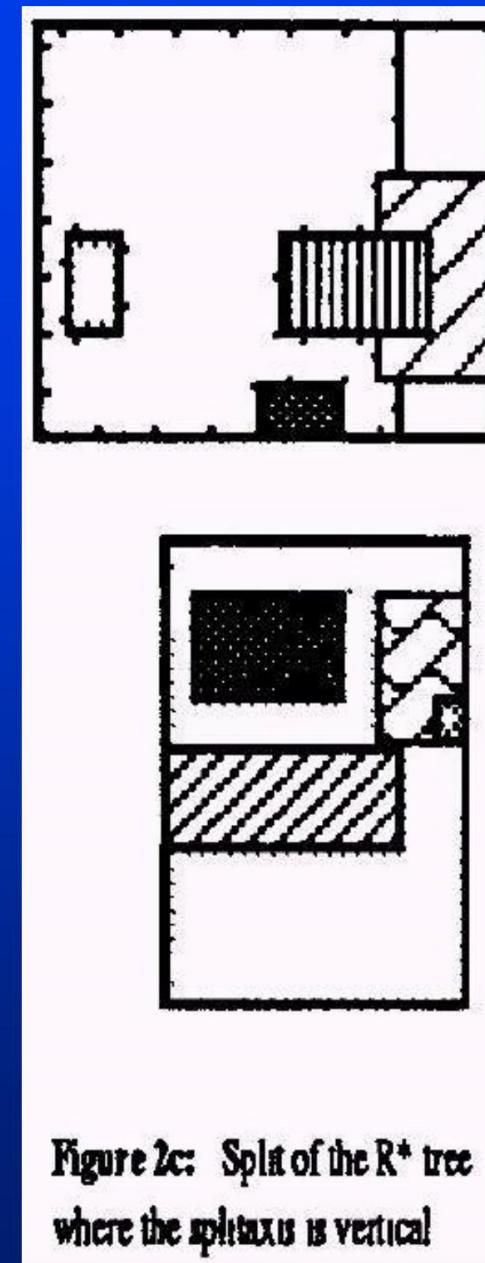


Figure 2a Overfilled node



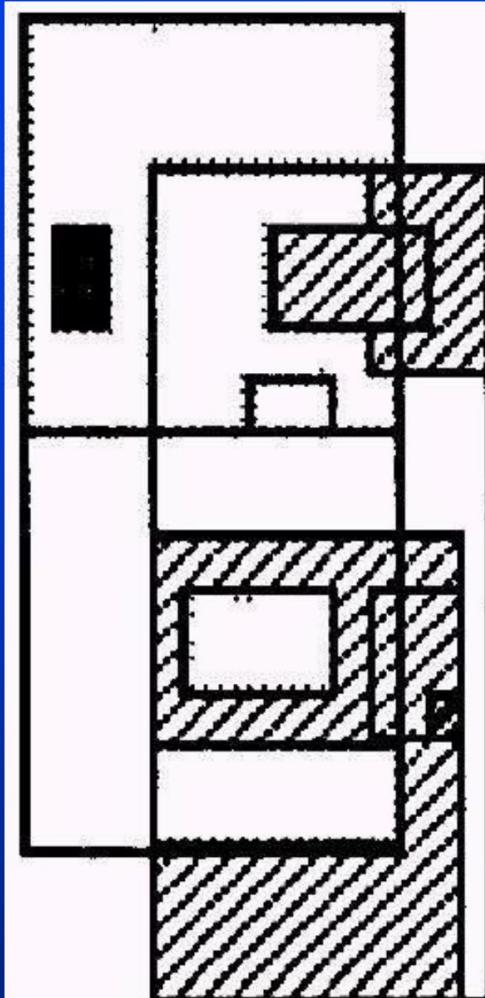


Figure 2b: Greene's split where the split axis is horizontal

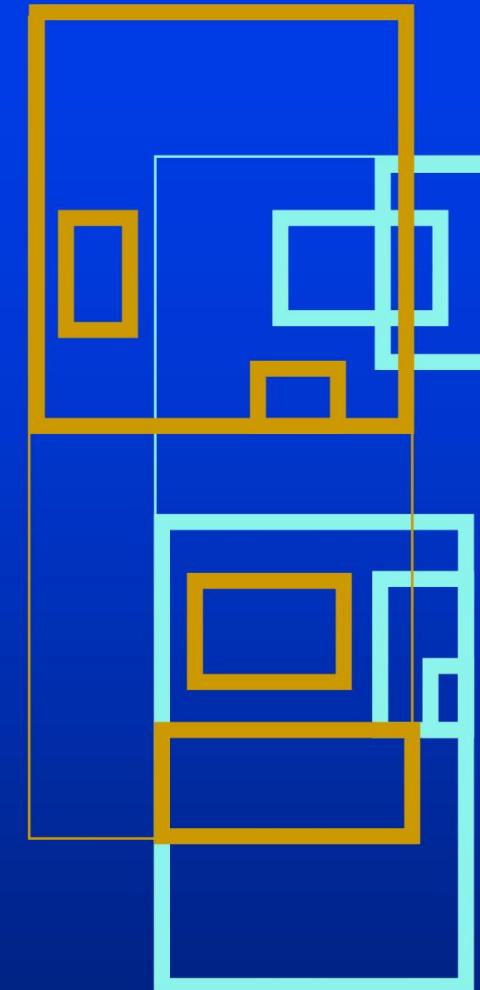


Figure 2b': Greene's split where the split axis is horizontal

Reinserción forzada

El árbol R y el árbol R* no son deterministas

diferentes secuencias de inserciones construirán diferentes árboles

El árbol R sufre por sus antiguas entradas

Durante una división se realiza una reorganización muy local de los rectángulos de directorio.

es bastante pobre

Necesita un instrumento más potente y menos local para reorganizar la estructura.

Algoritmos de eliminación e inserción por Guttman

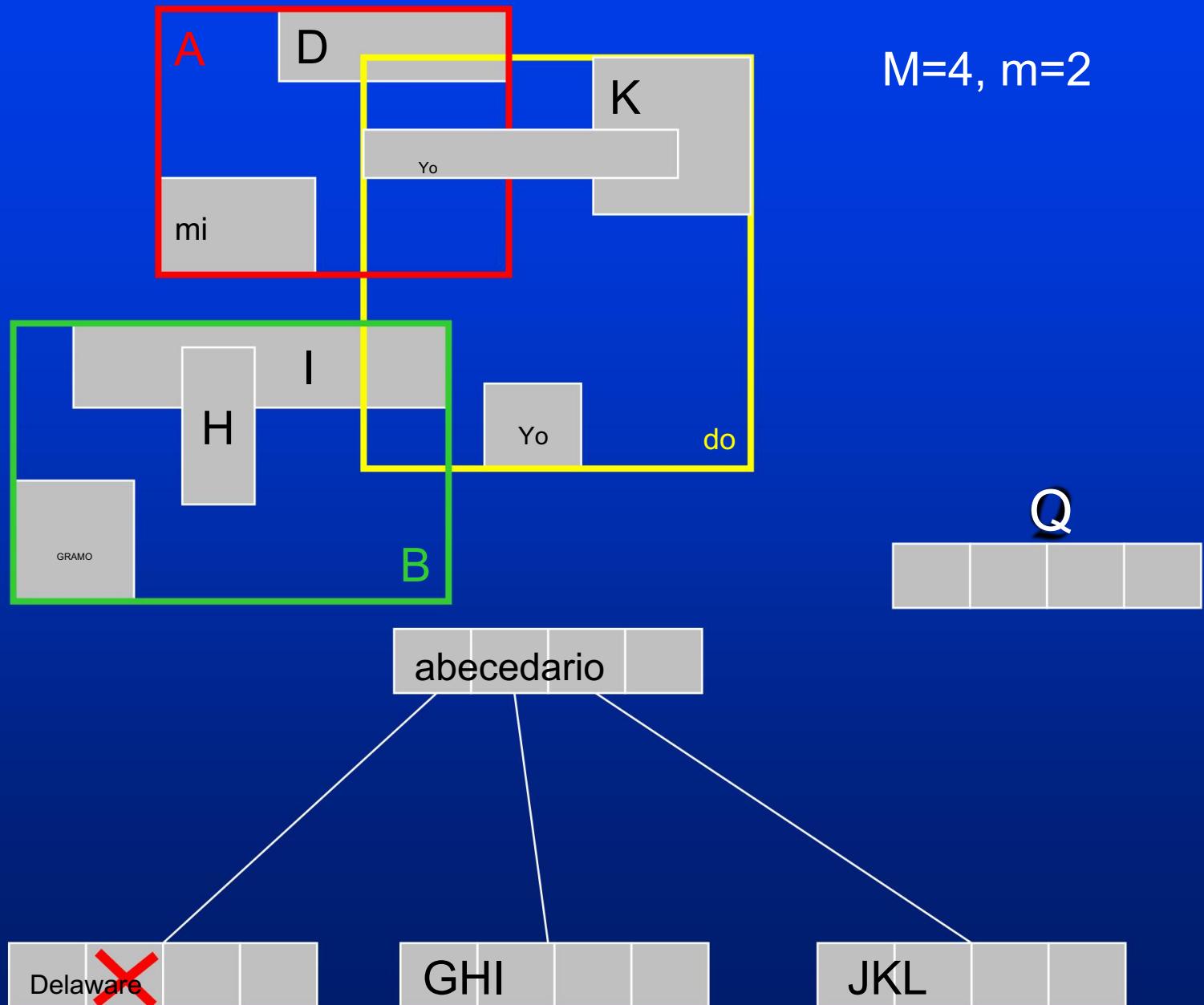
Algoritmo Deletion

- D1 Invocar Findleaf
- D2 Eliminar registro de índice E
- D3 Invocar CondenseTree -----> Invocar Insertar
- D4 Si es necesario, hazle al niño el nuevo techo.

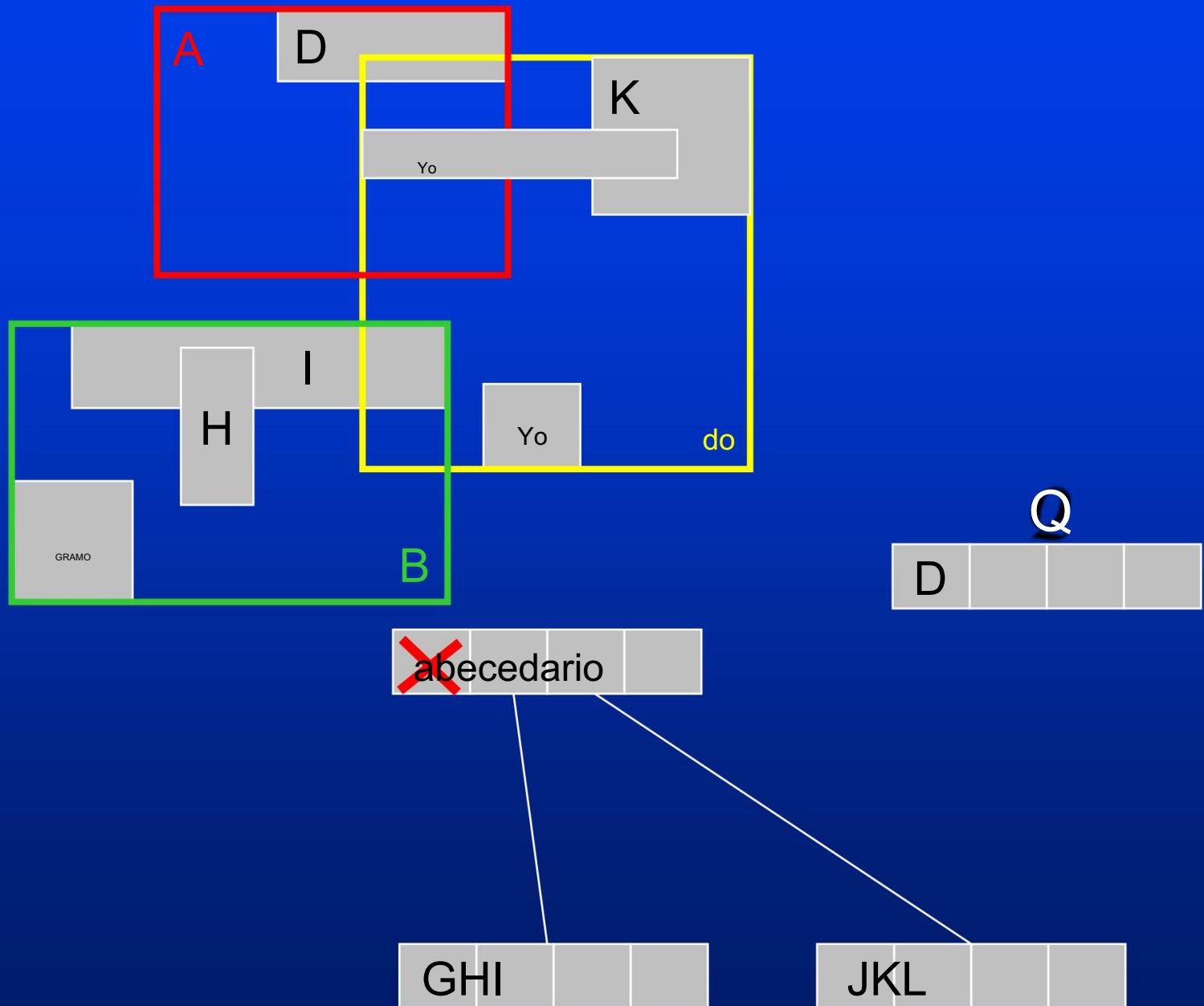
Inserción de algoritmo

- I1 Invocar ChooseSubtree
- I2 Instalar E, o invocar SplitNode (QuadraticSplit)
- I3 Invocar AdjustTree
- I4 Si es necesario, cree una nueva raíz

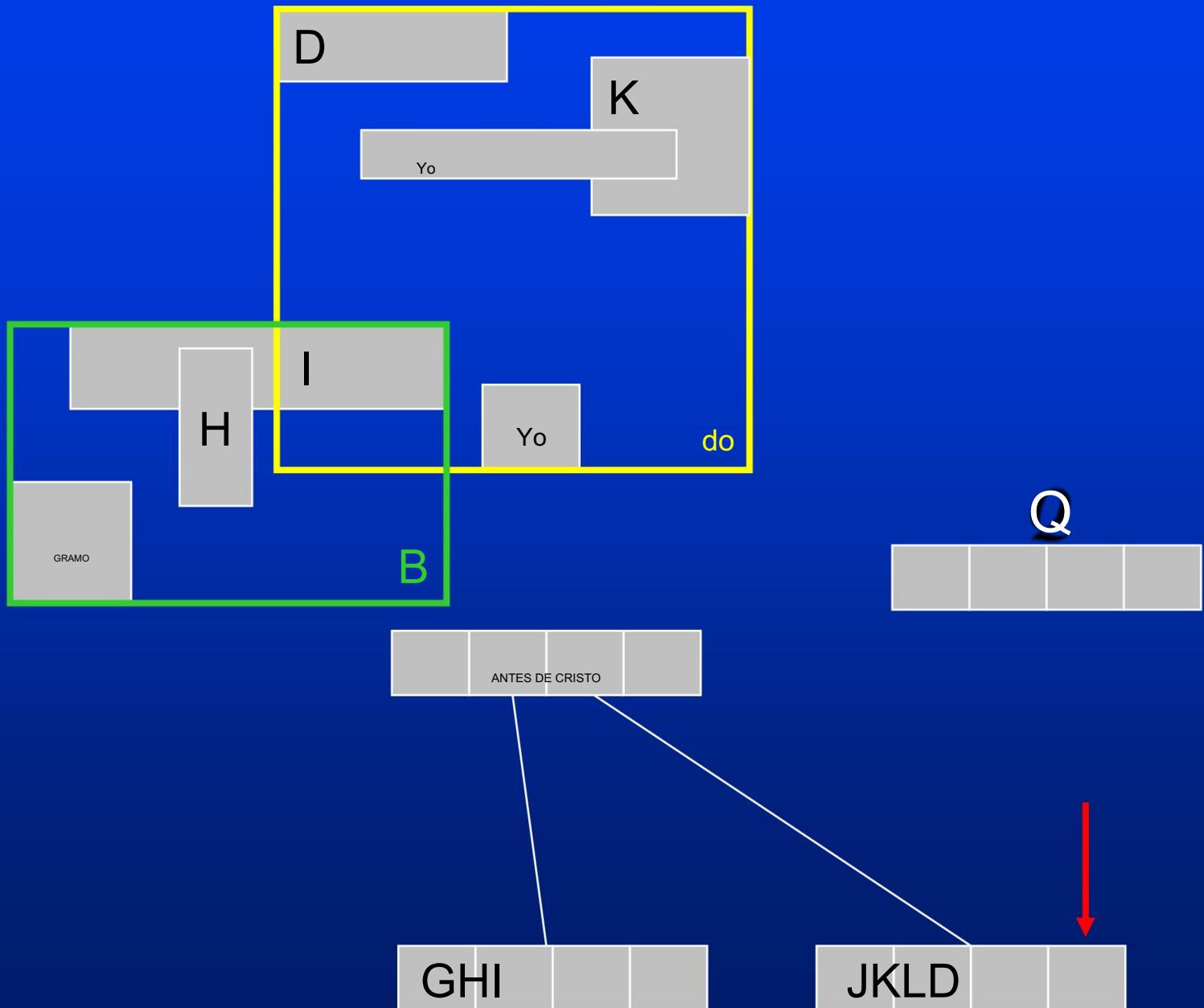
Tratamiento de nodos con poco relleno en el árbol R



Tratamiento de nodos con poco relleno en el árbol R



Tratamiento de nodos con poco relleno en el árbol R



Reinserción forzada

Se espera que la eliminación y reinserción de rectángulos de datos antiguos mejoren el rendimiento de recuperación.

El experimento mostró una mejora del rendimiento del 20% ~ 50% (situación estática)

Para lograr reorganizaciones dinámicas, el árbol R* fuerza la reinserción de las entradas durante las rutinas de inserción algoritmos iguales a Guttman, excepto por el tratamiento de desbordamiento

Reinserción forzada

Algoritmo Insertar Datos

Identificación1

Invoque Insertar comenzando con el nivel de hoja como parámetro, para insertar un nuevo rectángulo de datos

Inserción de algoritmo

- I1 Invoque ChooseSubtree, con el nivel como parámetro, para encontrar un nodo N apropiado, en el que colocar la nueva entrada E
- I2 Si N tiene menos de M entradas, acomodar E en N. Si N tiene M entradas, invocar OverflowTreatment con el nivel de N como parámetro [para reinserción o división]
- I3 Si se llamó a OverflowTreatment y se realizó una división, propague OverflowTreatment hacia arriba si es necesario
Si OverflowTreatment provocó una división de la raíz, cree una nueva raíz
- I4 Ajuste todos los rectángulos de cobertura en la ruta de inserción de modo que sean MBR que encierran los rectángulos secundarios.

Algorithm OverflowTreatment

OT1 Si el nivel no es el nivel raíz y esta es la primera llamada de

OverflowTreatment en el nivel dado durante la inserción de un rectángulo de datos, luego

invocar Reinsertar

demás

invocar Split

fin

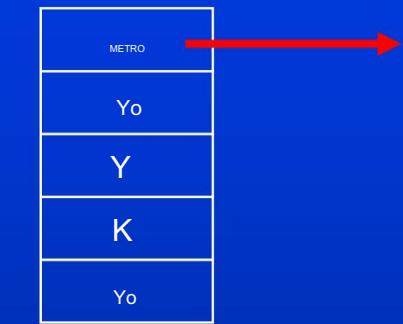
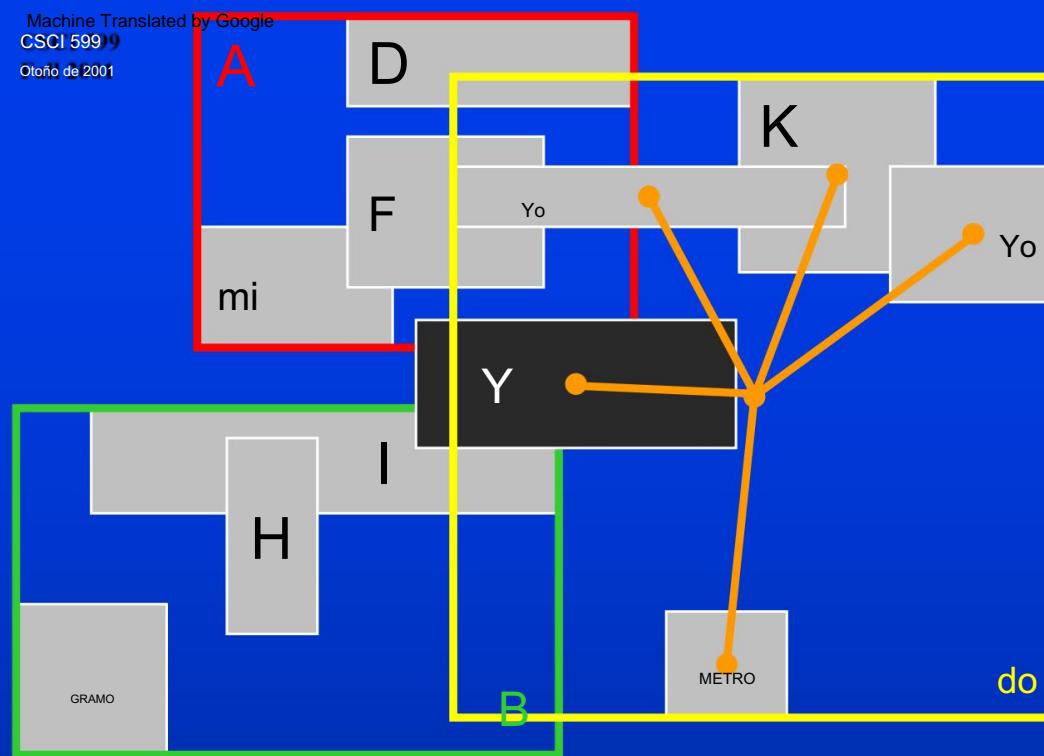
Algorithm Reinsert

RI1 Para todas las $M+1$ entradas de un nodo N, calcule la distancia entre los centros de sus rectángulos y el centro del rectángulo delimitador de N

RI2 Ordene las entradas en orden decreciente de sus distancias calculadas en RI1

RI3 Elimina las primeras p entradas de N y ajusta el rectángulo delimitador de N

RI4 En la clasificación, definida en RI2, comenzando con la distancia máxima (= reinserción lejana) o la distancia mínima (= reinserción cercana), invoque Insertar para reinsertar las entradas



M = 4

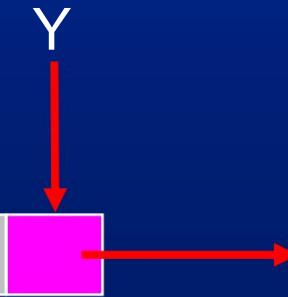
p = 30% de M = 1

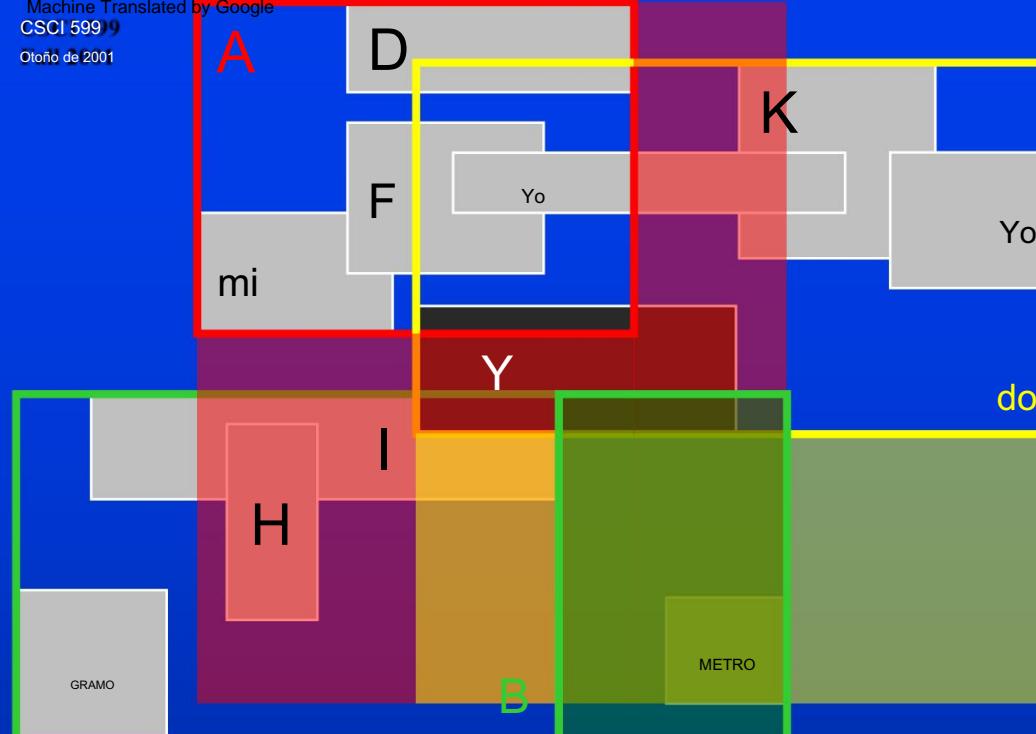
abecedario

DEF

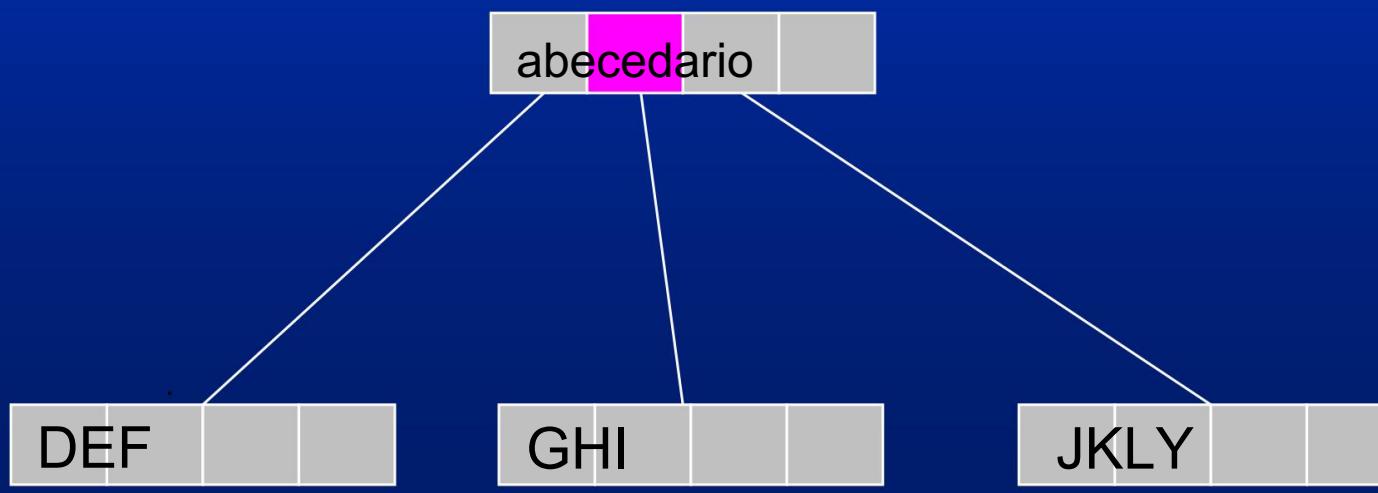
GHI

JKLM





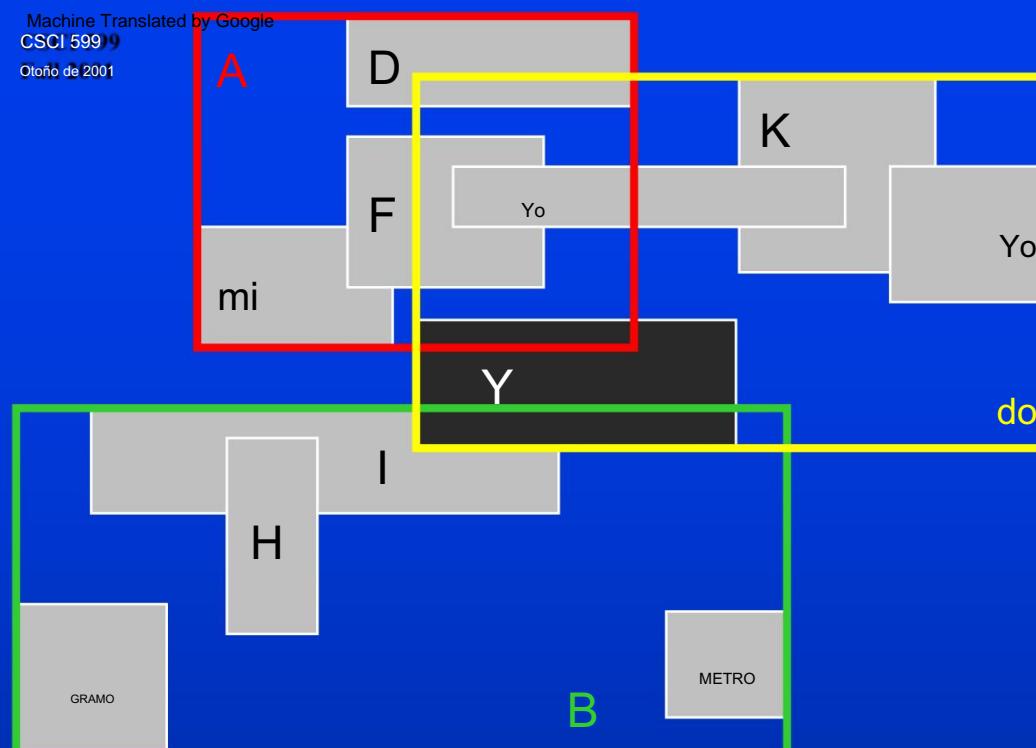
Reinsertar
Insertar
ElegirSubárboltree



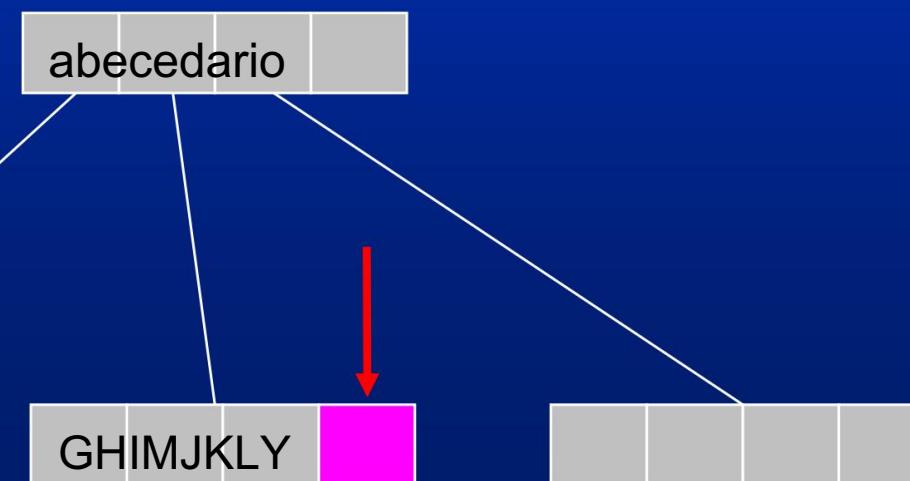
Reinsertar

Insertar

ElegirSubárbol



¡Se evita la división!



Reinserción forzada

$p = 30\%$ de M para todos los nodos produce el mejor rendimiento

Cambiar entradas entre nodos vecinos

disminuir la superposición

Mejorar la utilización del almacenamiento

Debido a una mayor reestructuración, se producen menos divisiones

Los rectángulos exteriores de un nodo se reinsertan

forma de los rectángulos de directorio más cuadráticos

Mayor costo de CPU (más llamadas de inserción), aliviado por menos divisiones

Configuración experimental

Cuatro variantes del árbol R

Árbol R con algoritmo de división cuadrática (qua.Gut)

Árbol R con algoritmo de división lineal (lin.Gut)

Variante de Greene del árbol R (Greene)

Árbol R*

Configuración experimental

Seis archivos de datos que contienen aproximadamente 100.000 rectángulos 2D

Distribución de centros de rectángulos

Uniforme: distribución uniforme independiente 2D

Clúster: distribución con 640 clústeres, 1600 objetos/clúster

Parcela : 100.000 rectángulos disjuntos. Expande cada área por un factor de 2,5.

Datos reales: MBR de líneas de elevación a partir de datos cartográficos reales

Gaussiano: distribución gaussiana independiente 2D

Uniforme Mixto: Distribución uniforme independiente 2D (añadir
1.000 rectángulos grandes a 99.000 rectángulos pequeños y luego fusionarlos)

Configuración experimental

Tipos de consultas

Consulta de intersección de rectángulos

- dado un rectángulo S, encuentre todos los rectángulos R en el archivo con $R \cap S$

Consulta de puntos

- dado un punto P, busque todos los rectángulos R en el archivo con $P \in R$.

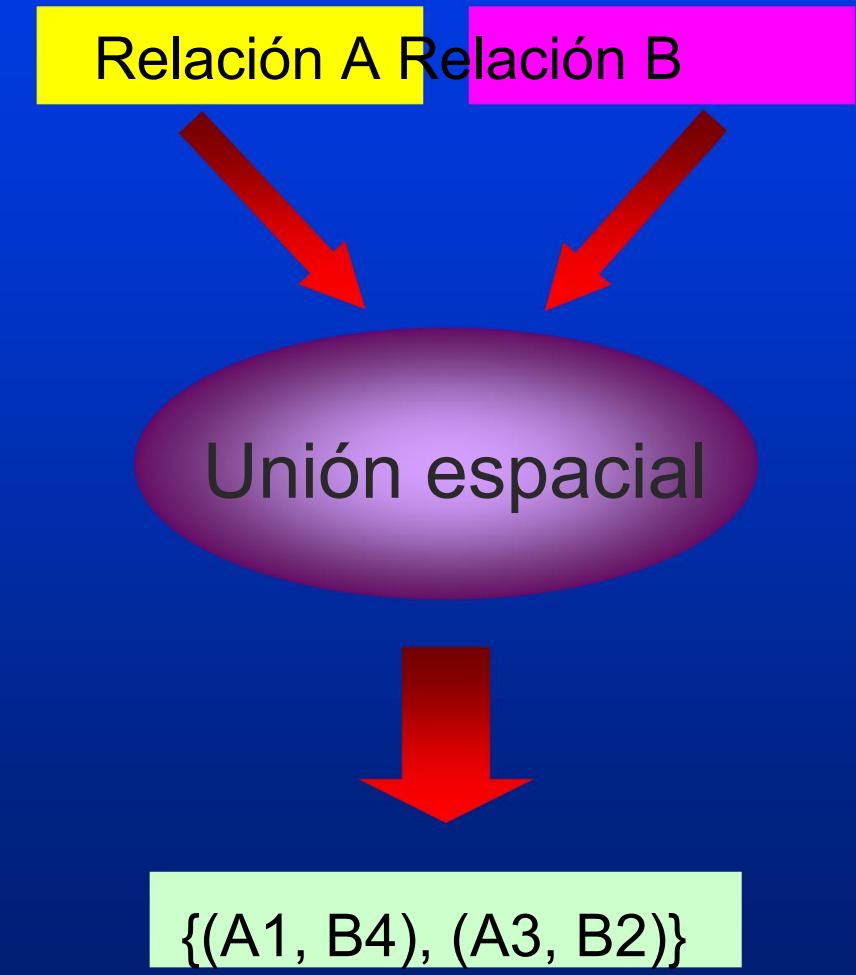
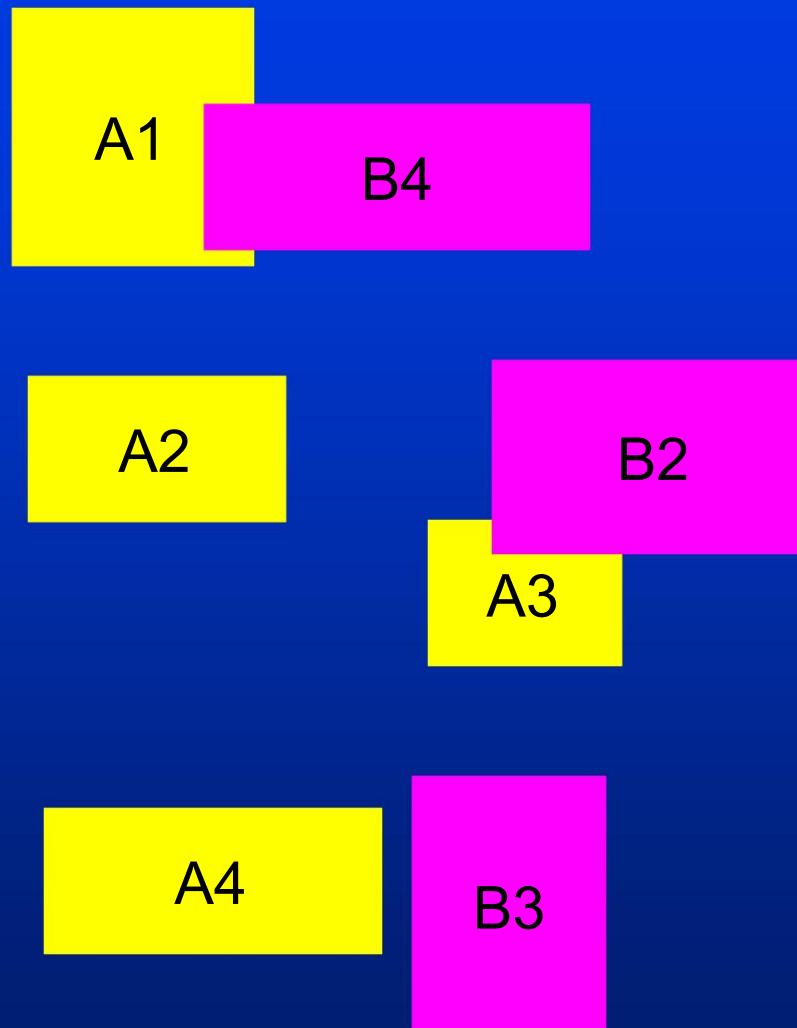
Consulta de recinto rectangular

- dado un rectángulo S, encuentre todos los rectángulos R en el archivo con $R \supseteq S$

Unión espacial

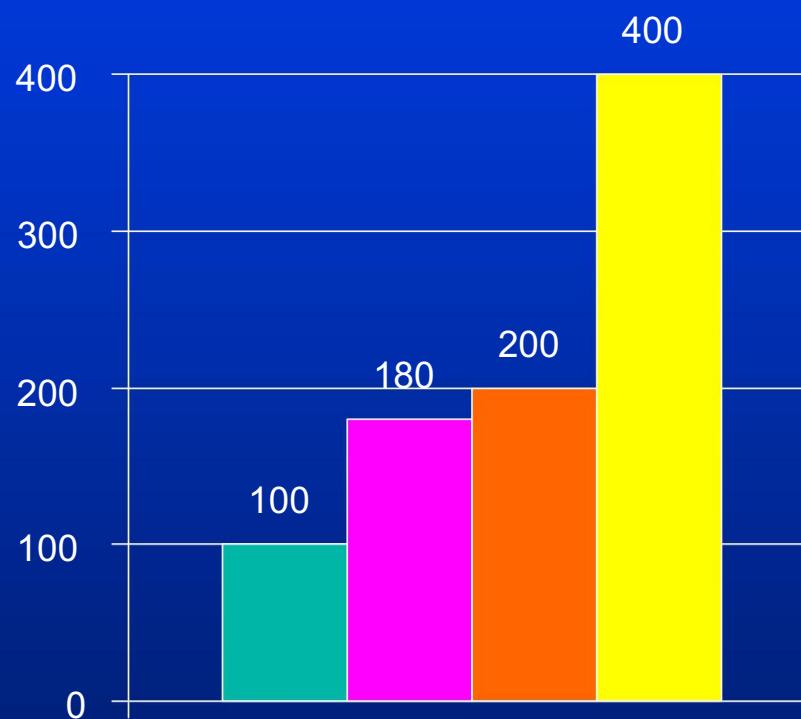
Todos los experimentos se midieron en número de accesos al disco.

Unión espacial (superposición de mapas)

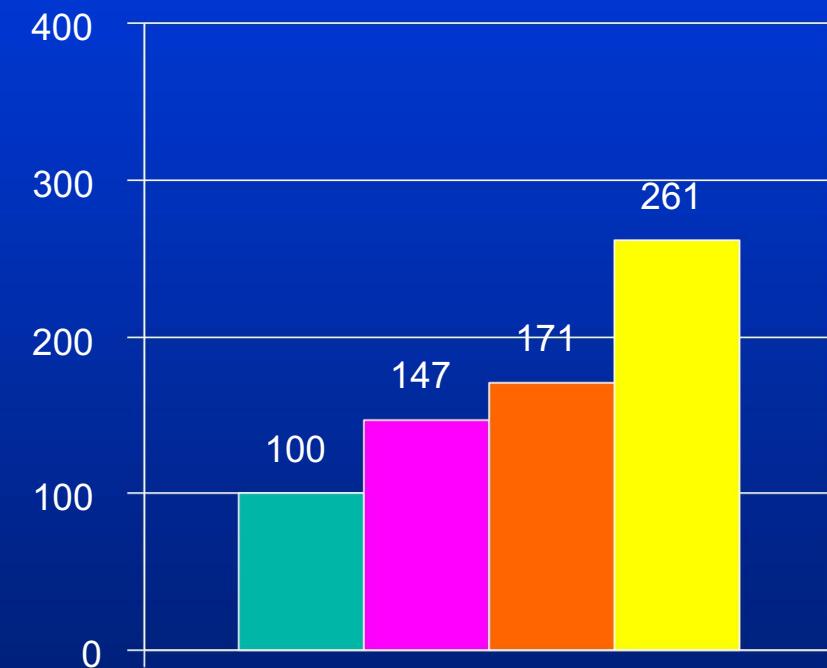


Resultados de los experimentos

Máxima ganancia de rendimiento de los árboles R* en todas las consultas



Ganancia de rendimiento promedio de árboles R* para unión espacial



Resultados de los experimentos

El árbol R* supera a las variantes del árbol R en todos los experimentos

Mayor ganancia en el árbol R* para rectángulos de consulta más pequeños

La utilización del almacenamiento se vuelve más importante para rectángulos más grandes

El árbol R* tiene la mejor utilización del almacenamiento

A pesar del uso de Reinserción Forzada, la inserción promedio

El costo del árbol R* sigue siendo el más bajo

Otoño 2001 R*-tree: un método eficiente de acceso a puntos

Otro experimento de referencia para PAM [KSSS 89]

incluye un archivo de cuadrícula de 2 niveles, un PAM muy popular

La ganancia de rendimiento es considerablemente mayor para los puntos que para los rectángulos

El árbol R* supera a las variantes del árbol R para datos puntuales y Utilización del almacenamiento, incluso archivos de cuadrícula

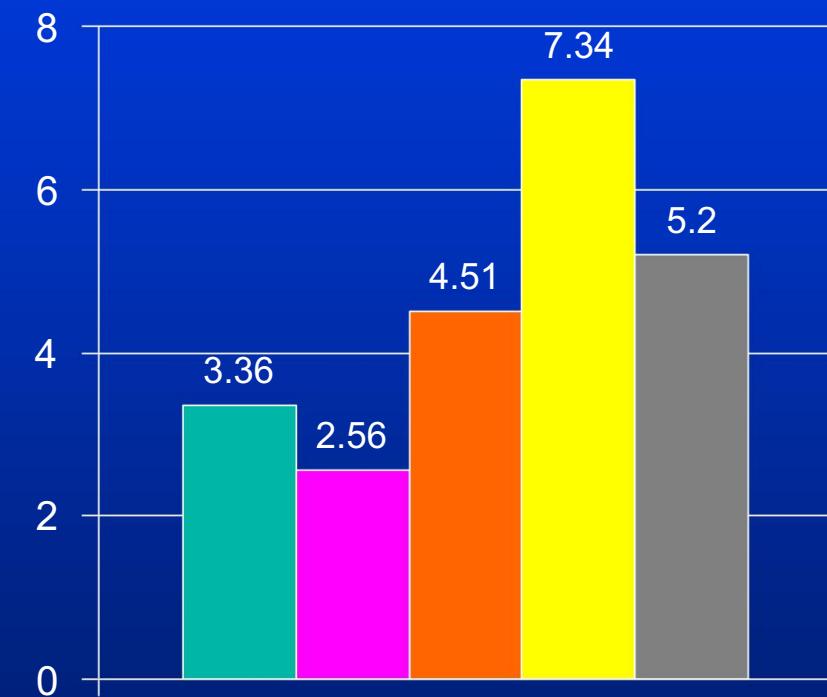
Resultados de los experimentos

Costo promedio de consulta promediado
sobre todas las consultas y archivos de datos



■ Cuadrícula de árbol R* □ cuadrático
■ lineal ■ Greene

Coste medio de inserción



■ Cuadrícula de árbol R* □ cuadrático
■ lineal ■ Greene

Conclusión

El árbol R* se puede utilizar eficientemente como método de acceso en sistemas de bases de datos que organizan puntos multidimensionales y datos espaciales.

Dado que el área, el margen y la superposición se reducen, el árbol R* es muy robusto frente a distribuciones de datos poco uniformes.