Non-key attribute organizations

- How to select the records of a large table R that satisfy a selective condition on non-key attributes?
- Indexes on the non-key attributes
- Parameters:
  - $N_{rec}(R)$, $N_{pag}(R)$
  - L_I , L_{RID}
  - $N_{key}(Idx)$, $N_{leaf}(Idx)$

Example

- Table

RID	Code	City	BY
1	100	MI	1972
2	101	PI	1970
3	102	PI	1971
4	104	FI	1970
5	106	MI	1970
6	107	PI	1972

- Indexes

Code	RID
100	1
101	2
102	3
104	4
106	5
107	6

Index on Code

BY	RID
1970	2
1970	4
1970	5
1971	3
1972	1
1972	6

Index on BY

Inverted indexes

- An **inverted index**: collection of pairs $<k_i, Inf_i>$, where k_i is a **non key** value and $Inf_i = (n_i, (r^i_1, r^i_2, \ldots, r^i_l))$, is the sorted list of RIDs of the records for k_i
- Table

RID	Code	City	BY
1	100	MI	1972
2	101	PI	1970
3	102	PI	1971
4	104	FI	1970
5	106	MI	1970
6	107	PI	1972

- Indexes

City	n	RID-List		
FI	1	4		
MI	2	1	5	
PI	3	2	3	6

Index on City

BY	n	RID-List		
1970	3	2	4	5
1971	1	3		
1972	2	1	6	

Index on BY

Assumptions

- The index-key values are uniformly distributed
- Records are uniformly distributed
- The index organization is a B+-tree with the sorted rid-lists stored in the leaves.
- Cost: $C_I + C_D$
  - C_I = cost of accessing the index leaves
  - C_D = cost of accessing the data pages
- $E_{rec} = \lceil s_f(\psi) \cdot N_{rec}(R) \rceil$
- $C_I = \lceil s_f(\psi) \cdot N_{leaf}(Idx) \rceil$

Equality search ($\psi = (A = v)$)

- $s_f(\psi) = s_f(A = v) = 1 / N_{key}(Idx)$
- Average length of a rid-list (AvgLRidList)

  $= \lceil s_f(\psi) \cdot N_{rec}(R) \rceil$

  $= \lceil N_{rec}(R) / N_{key}(Idx) \rceil$

- Space:

$$N_{\text{leaf}}(A) = \frac{N_{\text{reg}}(R) \times L_{\text{RID}} + N_{\text{key}}(A) \times L_A}{D_{\text{pag}} \times f_r}$$

Equality search(cont)

- NoPagesToVisitForRidList:
  - If the index is unclustered, with unsorted rid-lists…
  - If the index is clustered, with sorted rid-lists
    - $\lceil s_f(\psi) \times N_{pag}(R) \rceil$
  - If the index is unclustered, with sorted rid-lists
    - $\Phi(AvgLRidList, N_{pag}(R))$
    - Φ is called the Cardenas' formula
    - $\Phi(k,n) = n(1-(1-1/n)^k) \leq \min(k, n)$

Range key search($\psi = (v1 \leq A \leq v2)$)

- $s_f(\psi) = s_f(v1 \leq A \leq v2) = (v2 - v1)/(\max(A) - \min(A))$
- $C_l = \lceil sf(\psi) \times Nleaf(Idx) \rceil$

Range key search($\psi = (v1 \leq A \leq v2)$)

- CD = NoIndexKeyValues × NoPagesToAccessForRidList
- NoIndexKeyValues $= \lceil s_f(\psi) \times N_{key}(Idx) \rceil$
- If the index is clustered, with sorted rid-lists
  - NoPagesToAccessForRidList $= \lceil N_{pag}(R) / N_{key}(Idx) \rceil$
- If the index is unclustered, with sorted rid-lists,
  - NoPagesToAccessForRidList $= \lceil \Phi(N_{rec}(R)/ N_{key}(Idx), N_{pag}(R)) \rceil$
- Unclustered, unsorted rid-lists
  - NoPagesToAccessForRidList $= \lceil N_{rec}(R)/ N_{key}(Idx) \rceil$

Other operations

- Selection condition with AND or with OR
- Insertion
- Deletion
- Update

Multi attribute indexes

Age	Sal	RID
35	10	
35	20	
40	80	
45	75	

Sal	Age	RID
10	35	
20	35	
75	45	
80	40	

Name	Age	Sal
Bob	35	10
Cal	40	80
Joe	35	20
Sue	45	75

Data records
sorted by name

RID	Age
	35
	35
	40
	45

RID	Sal
	10
	20
	75
	80

Multi attribute indexes

- Exact queries
- Range queries

Bitmap index

Table

RID	StudCode	City	BirthYear
1	100	MI	1972
2	101	PI	1970
3	102	PI	1971
4	104	FI	1970
5	106	MI	1970
6	107	PI	1972

Inverted indexes

City	n	RID Lists		
FI	1	4		
MI	2	1	5	
PI	3	2	3	6

Index on **City**

BirthYear	n	RID Lists		
1970	3	2	4	5
1971	1	3		
1972	2	1	6	

Index on **BirthYear**

Bitmap indexes

City	Bitmaps					
FI	0	0	0	1	0	0
MI	1	0	0	0	1	0
PI	0	1	1	0	0	1

BirthYear	Bitmaps					
1970	0	1	0	1	1	0
1971	0	0	1	0	0	0
1972	1	0	0	0	0	1

Bitmap index

- The RID list becomes a bitmap
- The length of the bitmap is N_{rec}
- The i-th bit is set if the i-th record of the base table has the value for the indexed attribute.
- A BM index is used in all DBMS for constant tables when the number of distinct values of an indexed attribute is small (i.e. the attribute is **not selective**). An inverted lists index would be useless.

Advantages of bitmap index

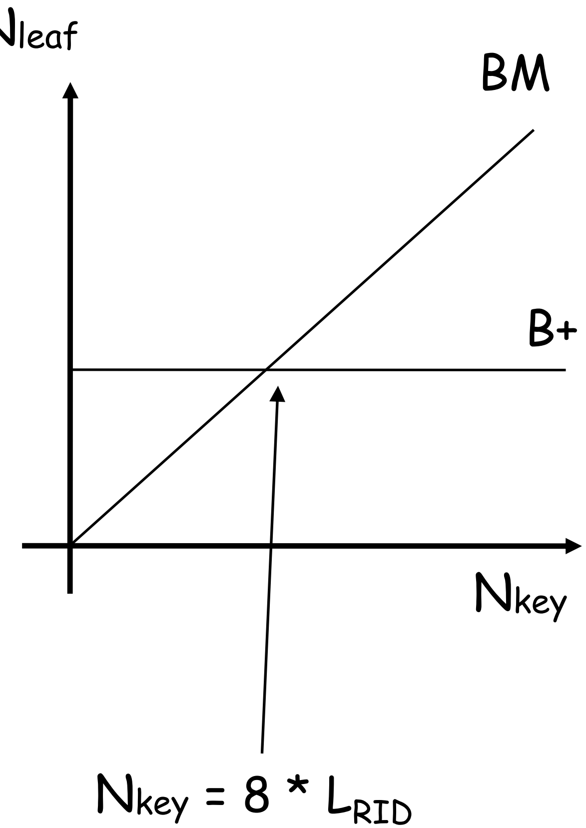
- Multi-attribute complex queries can be solved using bit operations
  - City in (‘Pisa’, ‘Lucca’) and (Year = 1972)
- Good if domain cardinality is small, but bit vectors can always be compressed

City	Bitmap					
FI	0	0	0	1	0	0
MI	1	0	0	0	1	0
PI	0	1	1	0	0	1

BirthYear	Bitmap					
1970	0	1	0	1	1	0
1971	0	0	1	0	0	0
1972	1	0	0	0	0	1

Memory comparison

- Hyp: full index nodes
- Inverted indexes
  - $N_{leaf} = (N_{key} * L_k + N_{rec} * L_{RID}) / D_{pag}$
    $\approx N_{rec} * L_{RID} / D_{pag}$
- Bitmap indexes
  - $N_{leaf} = (N_{key} * L_k + N_{key} * N_{rec} / 8) / D_{pag}$
    $\approx N_{rec} * N_{key} / (D_{pag} * 8)$

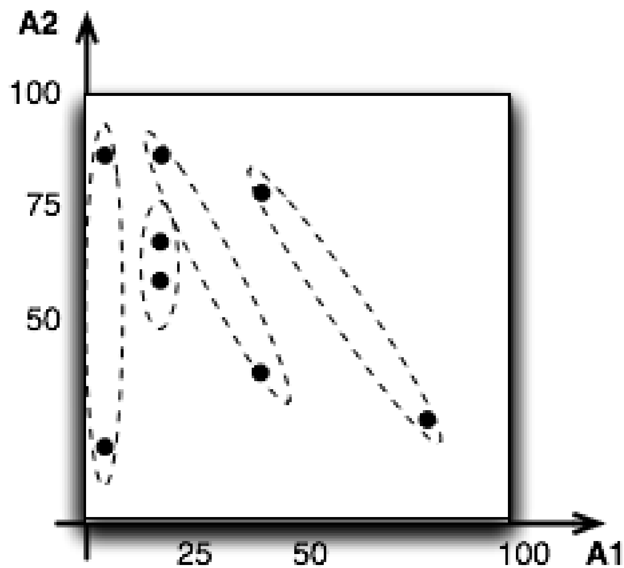


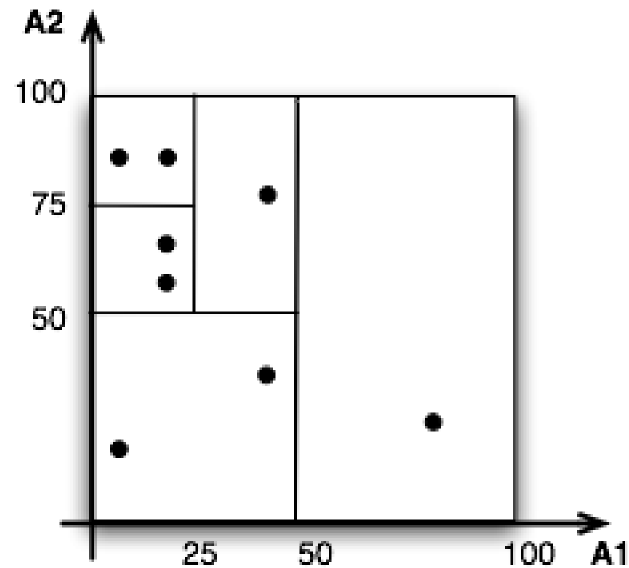
Oracle: Bitmap indexes are compressed and are suggested if $N_{key} < N_{rec}/2$

Multidimensional data organization

Multi-attribute index B^+-Tree (c=2)   multidimensional index

A1	A2	Rid
10	20	...
10	90	...
20	60	...
20	65	...
20	90	...
45	40	...
45	77	...
80	25	...





Point and region search: 15≤A1≤50 and 20≤A2≤40

Alternative: Store near points in the same page

Store (PartitionCode,PID) in a B^+-tree

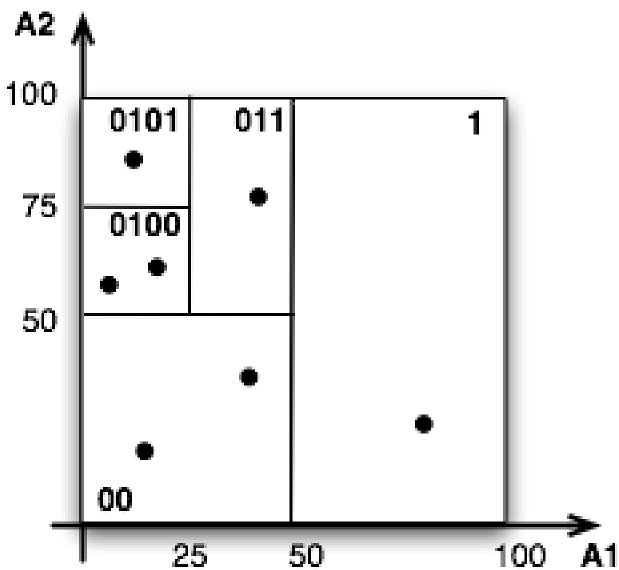
Multidimensional data organization

- Several proposals...

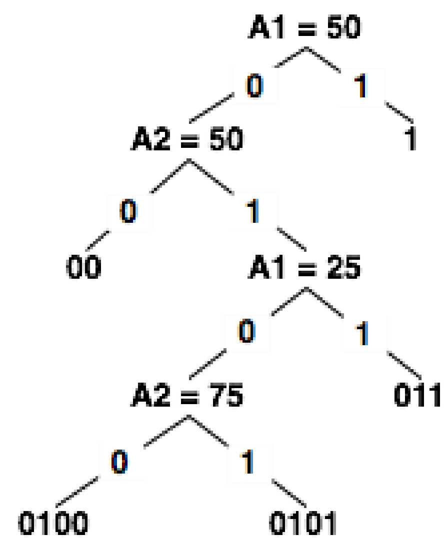
Quad Tree [Finkel 1974]
R-tree [Guttman 1984]
R^+-tree [Sellis 1987]
R^*-tree [Geckmann 1990]
Vp-tree [Chiueh 1994]
UB-tree [Bayer 1996]
SS-tree [White 1996]
M-tree [Ciaccia 1996]
Pyramid [Berchtold 1998]
DABS-tree [Böhm 1999]
Slim-tree [Faloutsos 2000]
P-Sphere-tree [Goldstein 2000]
K-D-B-Tree [Robinson 1981]
Grid File [Nievergelt 1984]
LSD-tree [Henrich 1989]
hB-tree [Lomet 1990]
TV-tree [Lin 1994]
hB-$^{\Pi}$-tree [Evangelidis 1995]
X-tree [Berchtold 1996]
SR-tree [Katayama 1997]
Hybrid-tree [Chakrabarti 1999]
IQ-tree [Böhm 2000]
Landmark file [Böhm 2000]
A-tree [Sakurai 2000]

Multidimensional data organization: the G-tree

Hyp: page capacity = 2

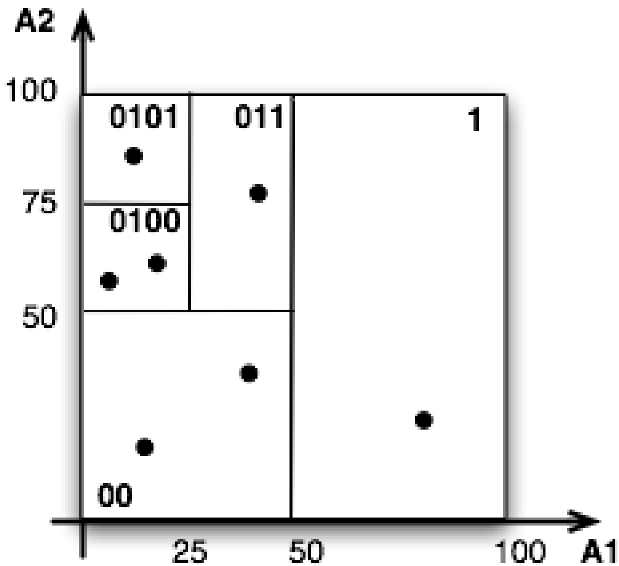


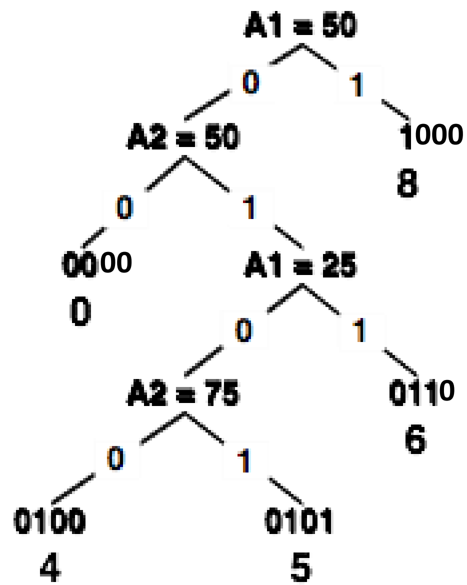
Space partitioning



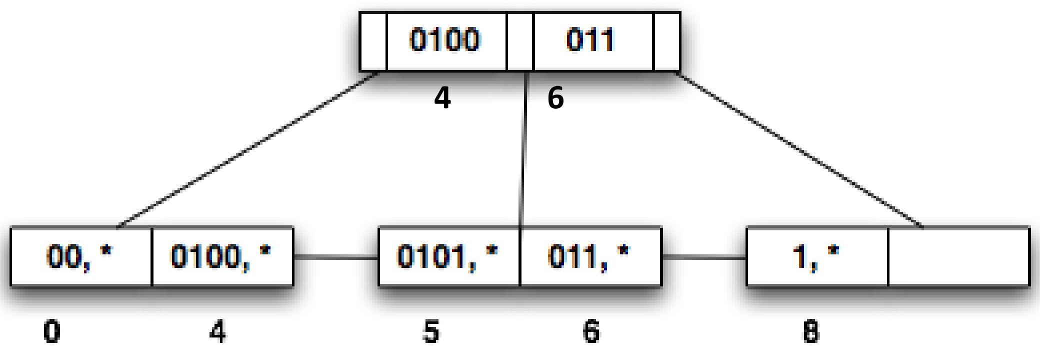
Partition tree: a partition has a binary code, with max length M

A tree structure for region codes: G-tree



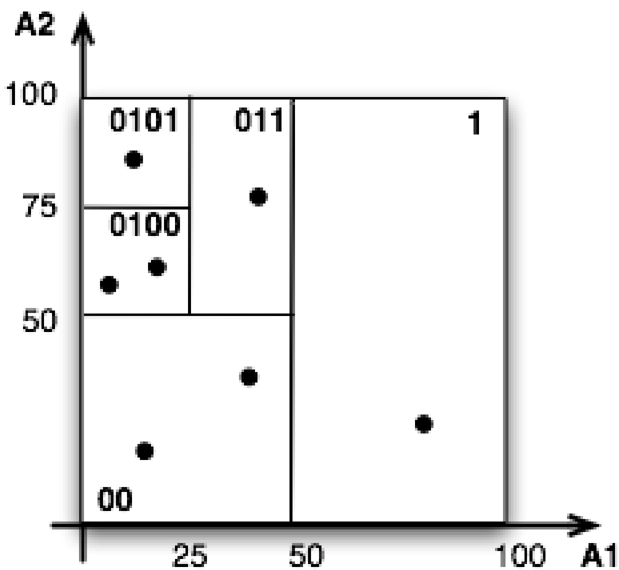


G-tree

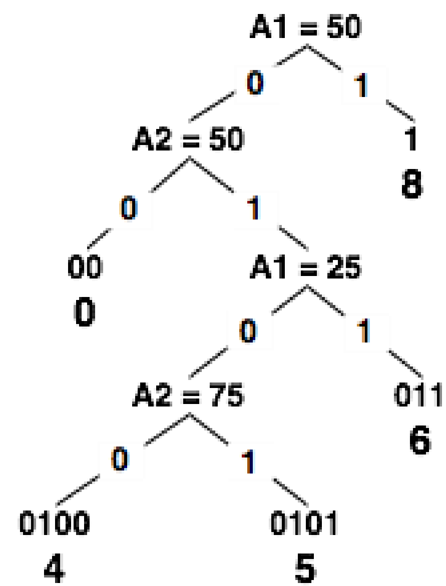


A partition binary code, can be seen as a decimal code, to better grasp the order

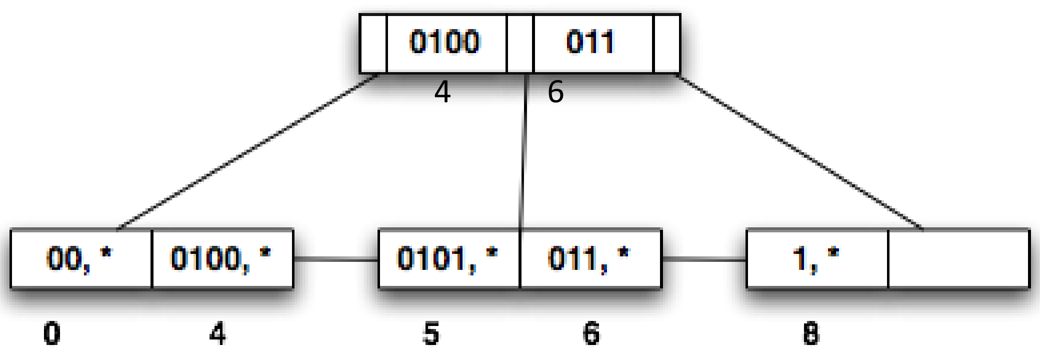
Point search: P(A1, A2) = (30,60)



Which partition contains P, if present?

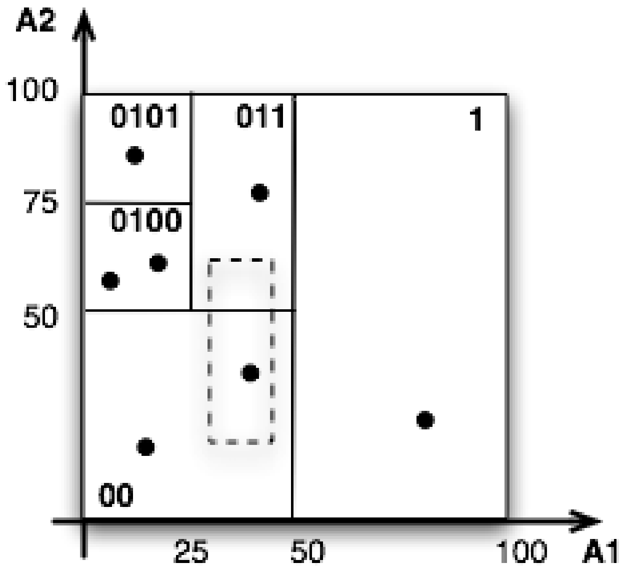


← P = (30, 60)

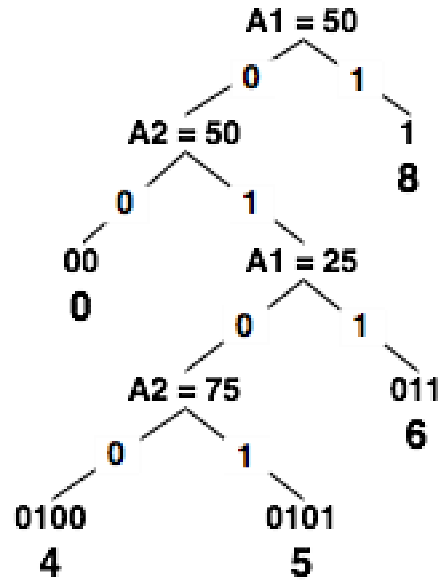


Spatial range search

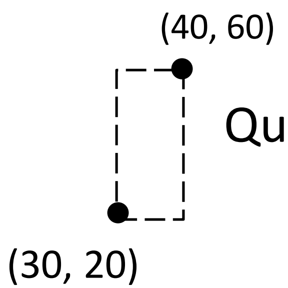
$30 \leq A1 \leq 40$
$20 \leq A2 \leq 60$



P = (30, 20) →

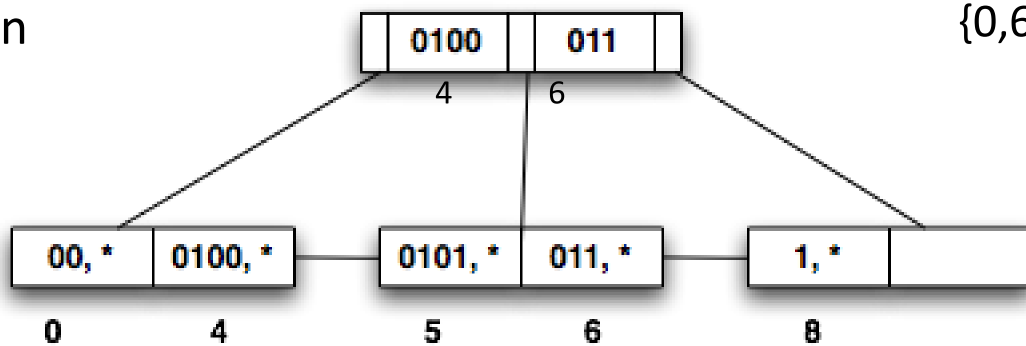


← P = (40, 60)

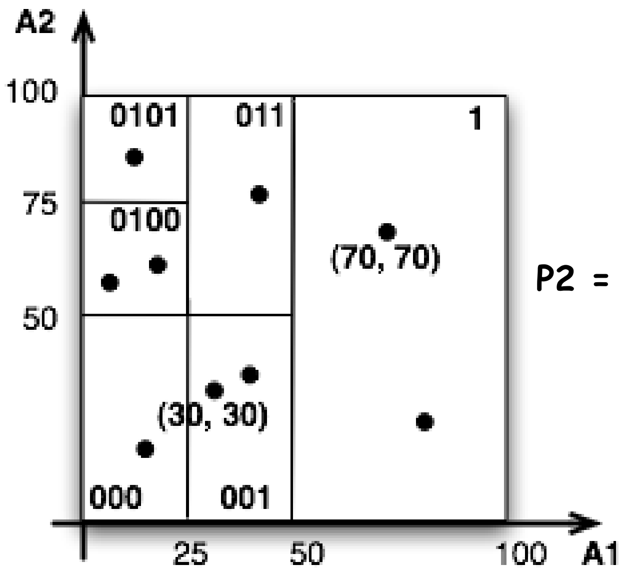


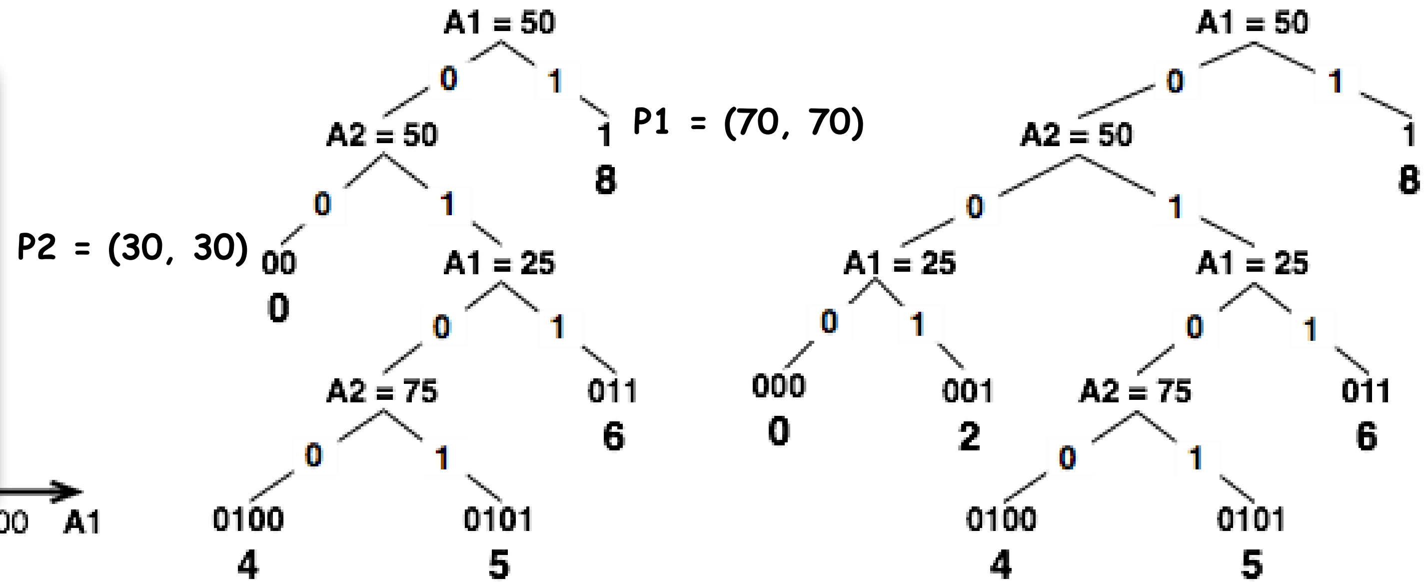
Query region

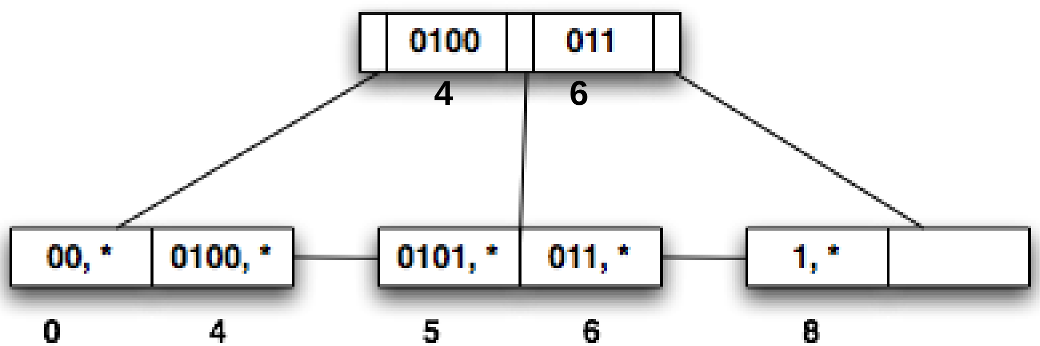
Which regions are checked?
{0,6} or {0,4,5,6} ?



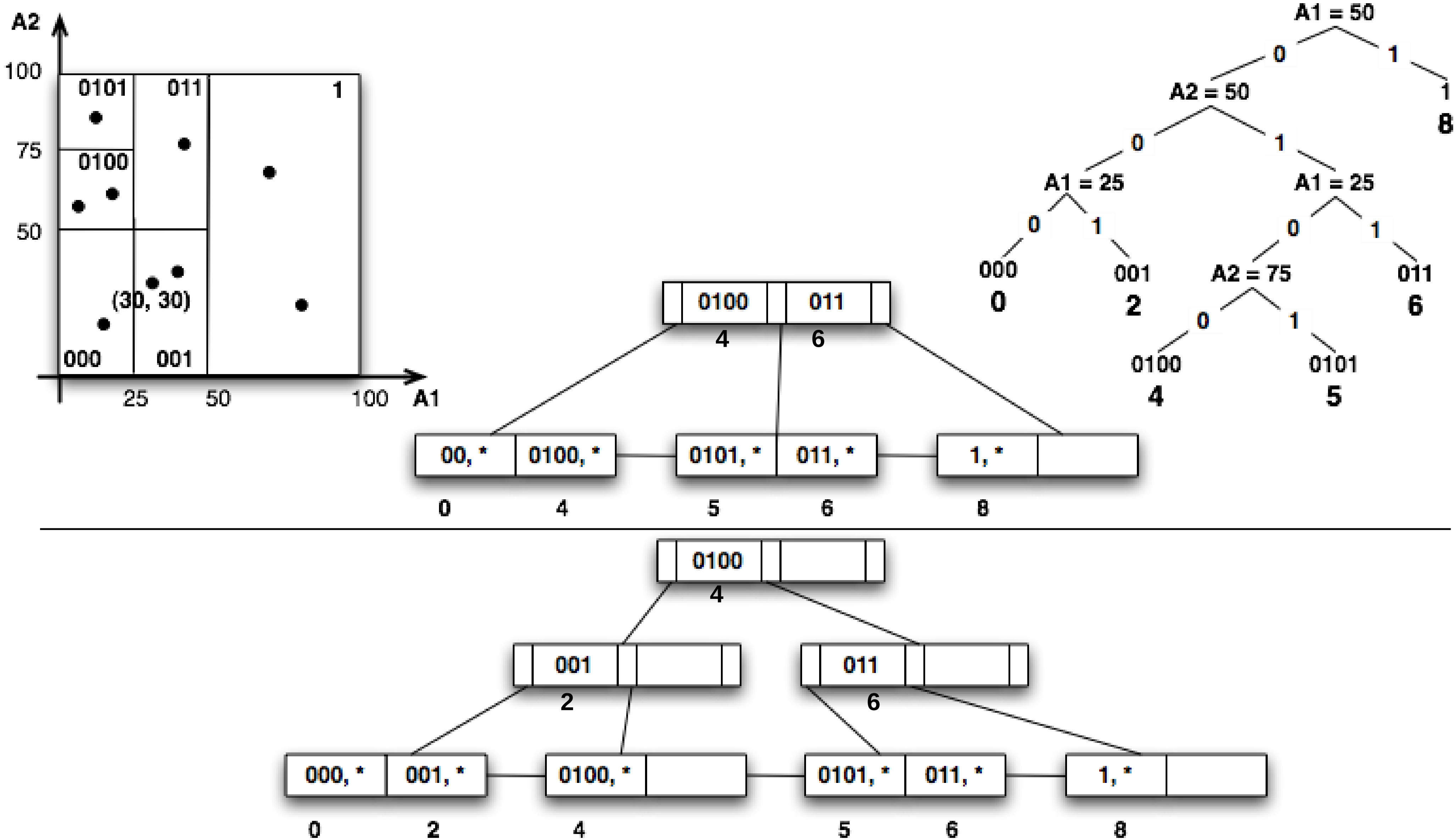
Insertion: p1 = (70, 70) and p2 = (30, 30)



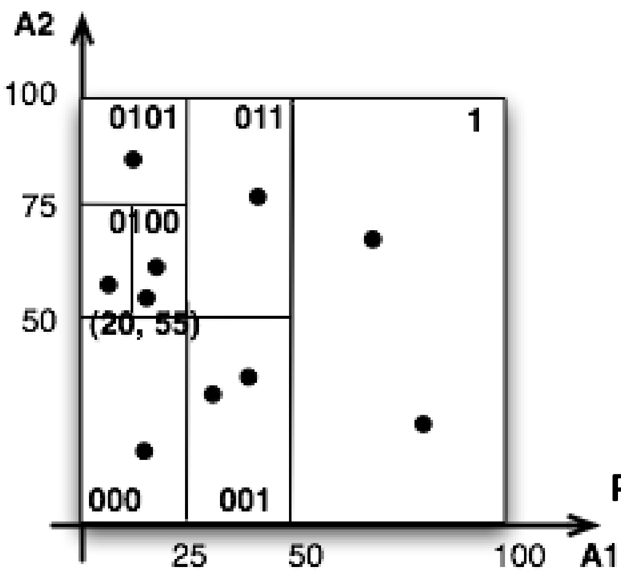


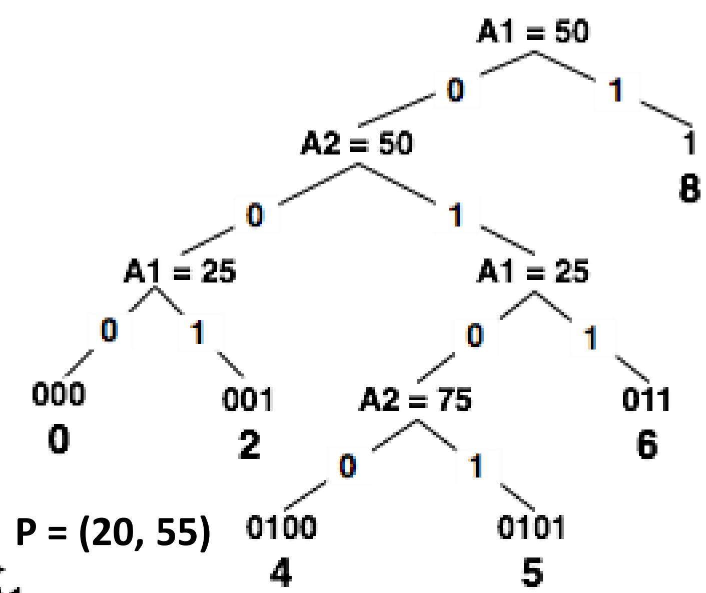


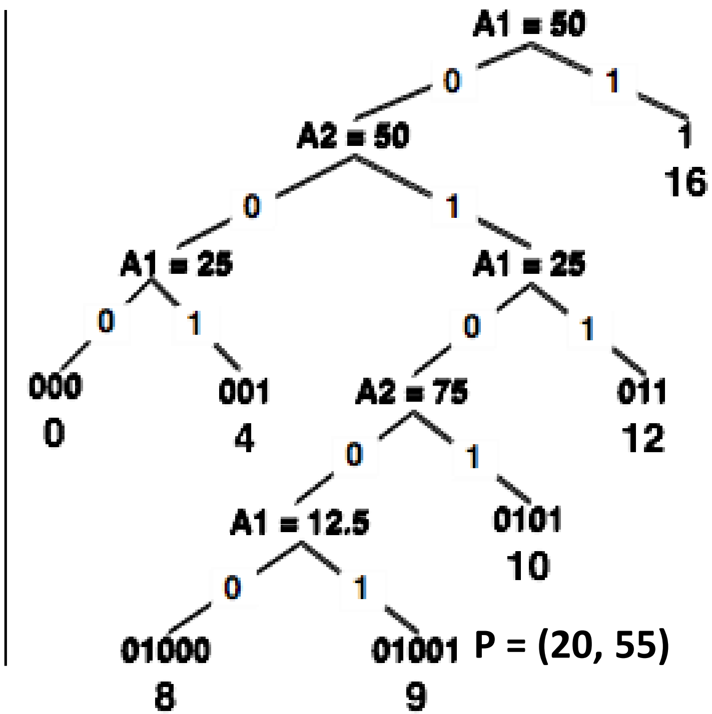
Point insertion: P2 = (30, 30)

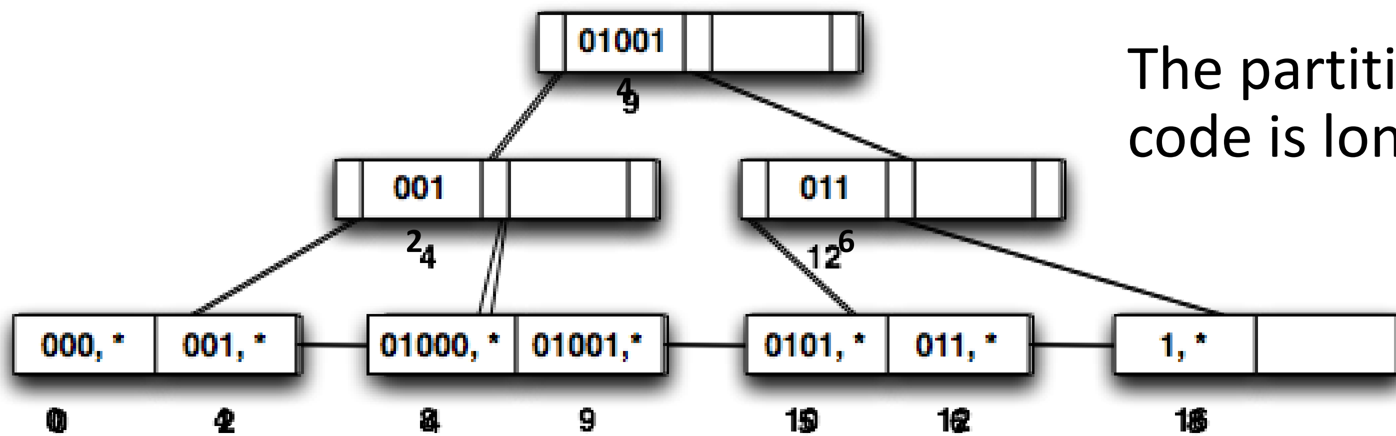


Point insertion with longer encoding









The partition binary code is longer

Geographical data

- R*-trees