Module 5: Disk-based Index Structures

Lecture 26: Pyramid Technique and VA-files

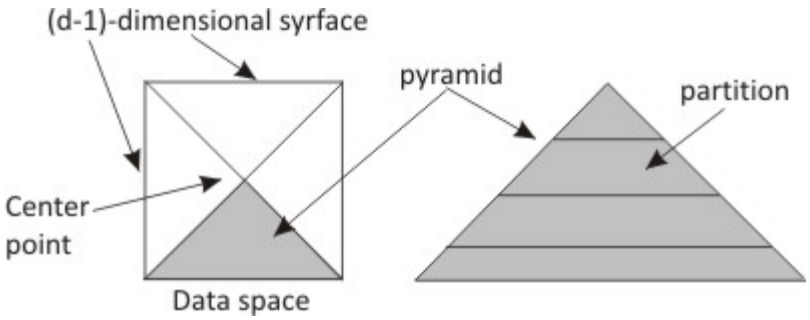
Prev topic | Next topic | Prev page | Next page

The Lecture Contains:

- Pyramid technique
  - Locational code
  - Searching
  - Extended pyramid technique
- VA-file
  - Analysis
  - Searching
- VA + - file

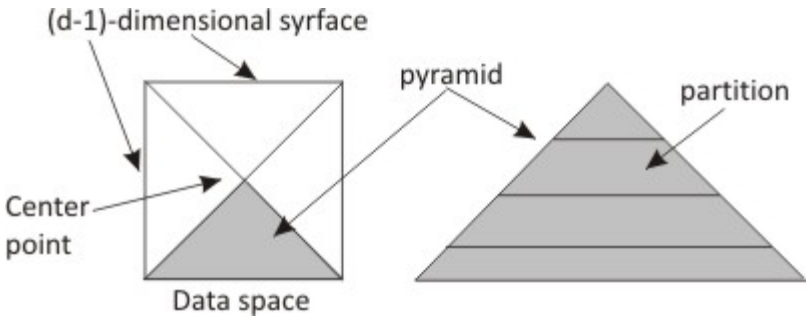
Pyramid technique

- Index nodes are disjoint pyramids
- $2d$ pyramids with centre as top
- Pyramids separated into partitions corresponding to pages



Pyramid technique

- Index nodes are disjoint pyramids
- $2d$ pyramids with centre as top
- Pyramids separated into partitions corresponding to pages
- Intersection, etc. is harder in d dimensions
- Therefore, transformation into 1-dimensional space
- A B+- tree can then be used
- Locational code for every d-dimensional point



Locational code

- Numbering of pyramids
  - Pyramid p_j has base 0 for dimension j
  - Pyramid p_{j+d} has base 1 for dimension j

Locational code

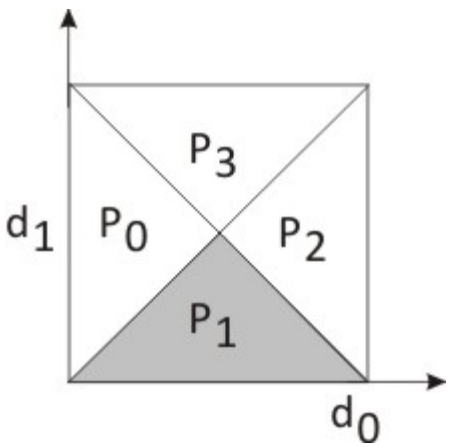
- Numbering of pyramids
  - Pyramid p_j has base 0 for dimension j
  - Pyramid p_{j+d} has base 1 for dimension j
- Determination of pyramid for a point $v = (v_0, v_1, \ldots, v_{d-1})$
  - Find dimension j where $|v_j - 0.5|$ is maximized
  - If $v_j < 0.5$, $i_v = j$ i.e., $v \in p_i$ where $i = j$
  - If $v_j \geq 0.5$, $i_v = j + d$, i.e., $v \in p_i$ where $i = j + d$

Locational code

- Numbering of pyramids
  - Pyramid p_j has base 0 for dimension j
  - Pyramid p_{j+d} has base 1 for dimension j
- Determination of pyramid for a point $v = (v_0, v_1, \ldots, v_{d-1})$
  - Find dimension j where $|v_j - 0.5|$ is maximized
  - If $v_j < 0.5$, $i_v = j$ i.e., $v \in p_i$ where $i = j$
  - If $v_j \geq 0.5$, $i_v = j + d$, i.e., $v \in p_i$ where $i = j + d$
- Height of a point v in a pyramid
  - $h_v = |0.5 - v_{i \mod d}|$ where $v \in p_i$

Locational code

- Numbering of pyramids
  - Pyramid p_j has base 0 for dimension j
  - Pyramid p_{j+d} has base 1 for dimension j
- Determination of pyramid for a point $v = (v_0, v_1, \ldots, v_{d-1})$
  - Find dimension j where $|v_j - 0.5|$ is maximized
  - If $v_j < 0.5$, $i_v = j$ i.e., $v \in p_i$ where $i = j$
  - If $v_j \geq 0.5$, $i_v = j + d$, i.e., $v \in p_i$ where $i = j + d$
- Height of a point v in a pyramid
  - $h_v = |0.5 - v_{i \mod d}|$ where $v \in p_i$
- Locational code for point v
  - $l_v = i_v + h_v$
  - Many-to-one mapping

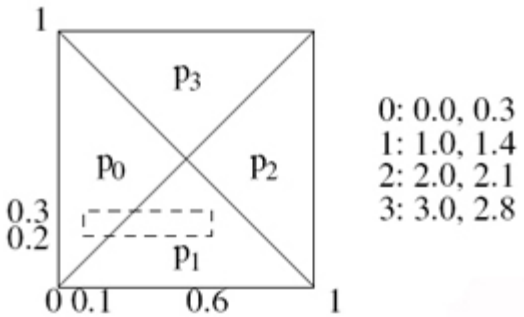


Searching

- For point queries, check candidates for true match
- Hyper-dimensional range queries
  - Transform one d-dimensional range query to $2d$ one-dimensional range queries, one for each pyramid
  - Difficult to find exact ranges
  - For pyramid p_j , range becomes $[j, j + (0.5 - min_j)]$
  - For pyramid p_{j+d} , range becomes $[j + d, j + d + (max_j - 0.5)]$
  - Requires further refinement
  - If true answer set is T, and $1 - d$ queries produce T', then $o \in T \implies o \in T'$ but not vice versa

Searching

- For point queries, check candidates for true match
- Hyper-dimensional range queries
  - Transform one d-dimensional range query to $2d$ one-dimensional range queries, one for each pyramid
  - Difficult to find exact ranges
  - For pyramid p_j , range becomes $[j, j + (0.5 - min_j)]$
  - For pyramid p_{j+d} , range becomes $[j + d, j + d + (max_j - 0.5)]$
  - Requires further refinement
  - If true answer set is T, and $1 - d$ queries produce T', then $o \in T \implies o \in T'$ but not vice versa
- Scales well with upto $\sim$25 dimensions for "squarish" queries

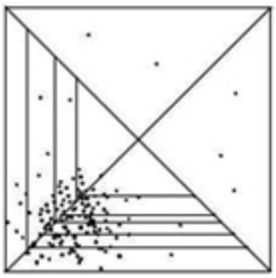


Extended pyramid technique

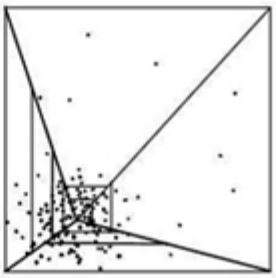
- Pyramid technique works well with uniformly distributed data

Extended pyramid technique

- Pyramid technique works well with uniformly distributed data
- Centre is not geographical, but d-dimensional median
- Transformation such that median becomes 0.5
  - $t_i(x) = x^r$ where $r = -1/lg(med_i)$
- Then, all algorithms follow



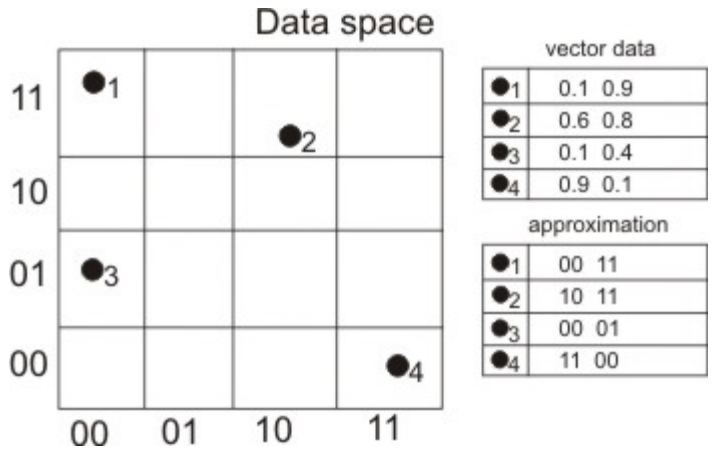
Pyramid technique



Extended pyramid technique

VA-file

- Vector approximation file
- Each dimension quantized into a fixed number of bits
- Total representation takes $b = \sum_{i=1}^{d} b_i$ bits
- Essentially, grids
- Each data point is approximated by the grid cell corresponding to its bit string
  - Concatenation of the bits for each dimension
- Maintained as an unordered file with pointers to actual objects



Analysis

- Dimension i has 2^{b_i} slices along it
- Slices contain nearly equal amount of data
- Slices are kept constant
    - May require periodic re-organization for efficiency

Analysis

- Dimension i has 2^{b_i} slices along it
- Slices contain nearly equal amount of data
- Slices are kept constant
  - May require periodic re-organization for efficiency
- Probability that there is a point inside a cell is 2^{-b}
- Probability that a grid cell contains more than one point is

$$P(\text{share}) = 1 - (1 - 2^{-b})^{N-1} \approx N/2^b$$

  - Example: $N = 10^8,\ b = 50:\ P(\text{share}) = 2^{-23}$
- More dimensions are better

Analysis

- Dimension i has 2^{b_i} slices along it
- Slices contain nearly equal amount of data
- Slices are kept constant
  - May require periodic re-organization for efficiency
- Probability that there is a point inside a cell is 2^{-b}
- Probability that a grid cell contains more than one point is

$$P(\text{share}) = 1 - (1 - 2^{-b})^{N-1} \approx N/2^b$$

  - Example: $N = 10^8,\ b = 50:\ P(\text{share}) = 2^{-23}$
- More dimensions are better
- Most cells are empty

Searching

- Two steps: filter (prune most of database) and refine (find true answers)
- Range searching
  - Filter
    - Compute lower bounds for data points using cells
    - If lower bound $> r$, prune
  - Refine
    - Find actual distances for candidates not pruned in filter step

Searching

- Two steps: filter (prune most of database) and refine (find true answers)
- Range searching
  - Filter
    - Compute lower bounds for data points using cells
    - If lower bound $>$ r , prune
  - Refine
    - Find actual distances for candidates not pruned in filter step
- Nearest neighbor searching
  - Filter
    - Compute lower and upper bounds for data points using cells
    - If lower bound $>$ upper bound of some other, prune
  - Refine
    - Find actual distances in increasing order of lower bound for candidates not pruned in filter step
    - If lower bound $>$ actual distance of some other, prune

Searching

- Two steps: filter (prune most of database) and refine (find true answers)
- Range searching
  - Filter
    - Compute lower bounds for data points using cells
    - If lower bound $>$ r , prune
  - Refine
    - Find actual distances for candidates not pruned in filter step
- Nearest neighbor searching
  - Filter
    - Compute lower and upper bounds for data points using cells
    - If lower bound $>$ upper bound of some other, prune
  - Refine
    - Find actual distances in increasing order of lower bound for candidates not pruned in filter step
    - If lower bound $>$ actual distance of some other, prune
- Less than 0.2% of the candidates remain after lter step
- Pruning gets better with more dimensions

VA+- file

- Improvement over VA-file in three ways
  - Transformed axes to better represent the data
  - Number of bits for each axis allocated non-uniformly
  - Optimal quantization for each axis