Module 4: Index Structures

Lecture 16: Voronoi Diagrams and Tries

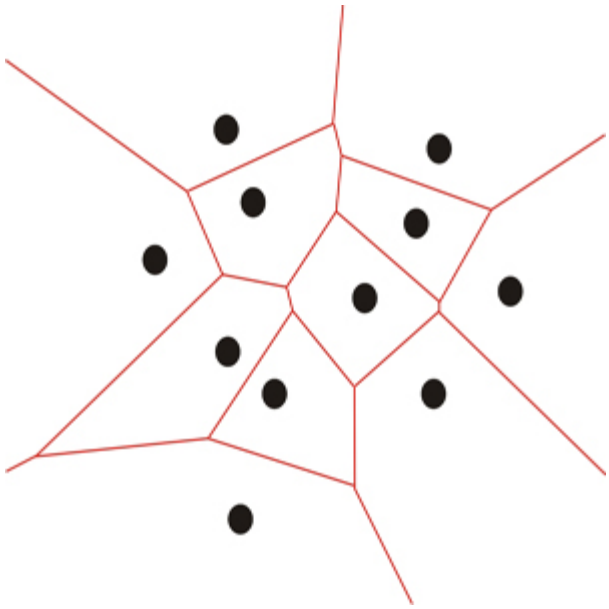
Prev topic Next topic Prev page
Next page

The Lecture Contains:

- Voronoi diagrams
  - Delaunay triangulation
  - Algorithms
  - Extensions
- Tries
- Index structures
  - 1-dimensional index structures
  - Memory-based index structures
  - Disk-based index structures
  - Classification of index structures

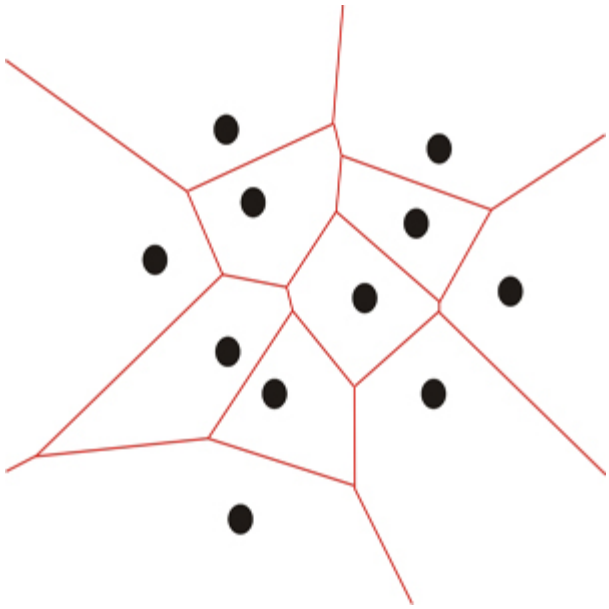
Voronoi diagrams

- Partition of a plane with N points (sites) into convex polygons such that
  - Each polygon contains one and only one site
  - Every point in the polygon is closest to the polygon's site than any other site
- Also known as Voronoi tessellation or Dirichlet tessellation



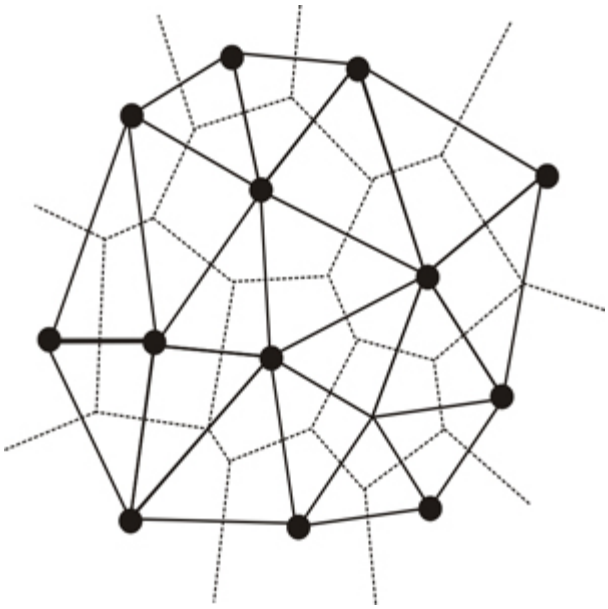
Voronoi diagrams

- Partition of a plane with N points (sites) into convex polygons such that
  - Each polygon contains one and only one site
  - Every point in the polygon is closest to the polygon's site than any other site
- Also known as Voronoi tessellation or Dirichlet tessellation
- Polygons are called Voronoi regions or Voronoi cells or Thiessen polygons or Dirichlet domains
- Vertices of the resulting graph Voronoi vertices
- Edges are Voronoi edges



Delaunay triangulation

- Dual of Voronoi diagram
- Triangulate such that circumcircles of every triangle is empty
- Voronoi vertices are circumcircles of Delaunay triangles



Algorithms

- Used for 1-NN searches
- Fortune's algorithm to construct Voronoi diagrams
    - Plane sweep
- Time to construct is $O(N \lg N)$
    - Optimal
- Requires $O(N)$ space to store

Extensions

- L_1 or Mahalanobis distance may be used instead of L_2
    - No guarantee that such a structure exists
- Weighted L_2 may be used

Extensions

- L_1 or Mahalanobis distance may be used instead of L_2
    - No guarantee that such a structure exists
- Weighted L_2 may be used
- In d dimensions, requires $O(N^{\lceil d/2 \rceil})$ space

Extensions

- L_1 or Mahalanobis distance may be used instead of L_2
    - No guarantee that such a structure exists
- Weighted L_2 may be used
- In d dimensions, requires $O(N^{\lceil d/2 \rceil})$ space
- Voronoi diagram of order- k
    - Partition according to k closest sites
    - Useful for k-NN searches
    - Ordinary Voronoi is order-1
    - Order-($(N-1)$) is called farthest point Voronoi diagram
    - In two dimensions, requires $O(k(N-k))$ space and $O(N \lg N + k^2(N-k))$ time

Extensions

- L_1 or Mahalanobis distance may be used instead of L_2
    - No guarantee that such a structure exists

Extensions

- L_1 or Mahalanobis distance may be used instead of L_2
    - No guarantee that such a structure exists
- Weighted L_2 may be used

Extensions

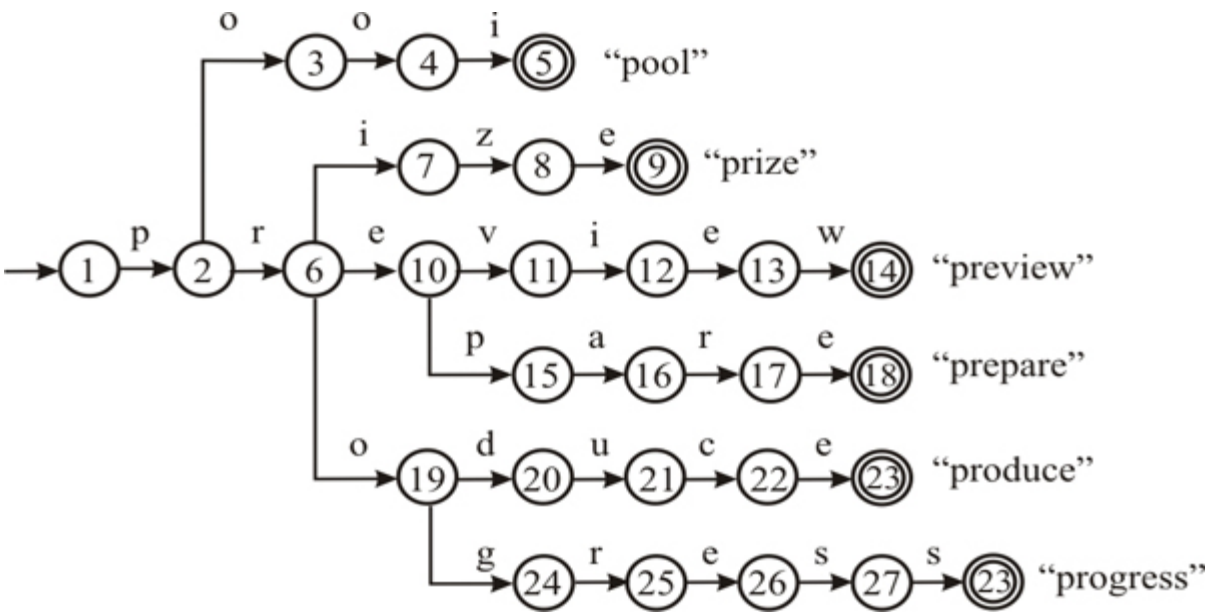
- L_1 or Mahalanobis distance may be used instead of L_2
    - No guarantee that such a structure exists
- Weighted L_2 may be used
- In d dimensions, requires $O(N^{\lceil d/2 \rceil})$ space

Extensions

- L_1 or Mahalanobis distance may be used instead of L_2
    - No guarantee that such a structure exists
- Weighted L_2 may be used
- In d dimensions, requires $O(N^{\lceil d/2 \rceil})$ space
- Voronoi diagram of order- k
    - Partition according to k closest sites
    - Useful for k-NN searches
    - Ordinary Voronoi is order-1
    - Order-($(N-1)$) is called farthest point Voronoi diagram
    - In two dimensions,requires $O(k(N-k))$ space and $O(N \lg N + k^2(N-k))$ time

Tries

- Comes from the word retrieval
- Mostly used for strings
- Structure
  - Root represents null string
  - Each edge defines the next character
  - Each node stores a string or a preffix of a string
  - Strings with same preffix share the path
- Advantages over binary search trees
  - Search time is $O(m)$ where m is the length of the query
  - Size is generally less
- Related structures: preffix tree, radix tree, suffix tree



Index structures

- Organize objects for effcient searching and retrieval
- Three main classes
    - 1-d index structures
    - Memory-based index structures
    - Disk-based index structures

1-dimensional index structures

- Hashing
    - Hash on single-dimensional key
    - Static hashing, Dynamic hashing
    - Some structures can be extended to multiple dimensions
- B-tree based structures
    - Key is single-dimensional
    - Supports disk-based accesses

Memory-based index structures

- K-d-tree, Range tree, Quadtree, etc.
- No page-based organization of nodes.
- Fanout (i.e., number of children) does not depend on page capacity.
- Mostly used for low-dimensional data.

Disk-based index structures

- Grid le, R-tree, K-d-B-tree, X-tree, M-tree.
- Not all objects t in memory.
- Disk-page-oriented organization of objects.
- Aim is to reduce number of random accesses for queries.

Classification of index structures

- Space-partitioning
    - Fanout independent of dimensionality
    - Problem of dead space indexing
    - No guarantee on space usage
    - K-d-B-tree, K-d-tree, Quadtree, Grid le
- Data-partitioning
    - Fanout decreases with dimensionality
    - Eliminates dead space
    - Guarantee on space usage
    - R-tree, R*-tree, X-tree

Classification of index structures

- Space-partitioning
    - Fanout independent of dimensionality
    - Problem of dead space indexing
    - No guarantee on space usage
    - K-d-B-tree, K-d-tree, Quadtree, Grid le
- Data-partitioning
    - Fanout decreases with dimensionality
    - Eliminates dead space
    - Guarantee on space usage
    - R-tree, R*-tree, X-tree
- Point access methods (PAM)
    - Only points are indexed
- Spatial access methods (SAM)
    - Spatial objects can be indexed