CSE 594: Spatial Data Science & Engineering

Lecture 5

Spatial Data Storing and Indexing

Part 1

Frequently Used Spatial Queries

- **Range Query**:- Find the spatial objects that lie in a query window. Also, known as box query or window query.
- **Point Query**:- Find the objects containing a given query point
- **K-Nearest Neighbor or KNN Query**:- Find the k nearest neighbors to a query object
- **Distance Query**:- Find all objects within a certain distance from a query point
- **Spatial Join Query**:- Find all pairs of objects between two datasets that satisfy a given predicate

Spatial Index

A data structure to access spatial objects efficiently from spatial databases avoiding sequential scan of data

Plays a core role in time-critical spatial applications

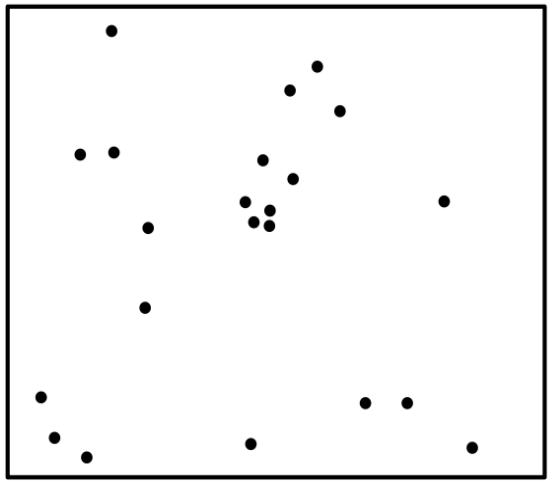
Allows efficient processing of spatial range, spatial join, and KNN queries

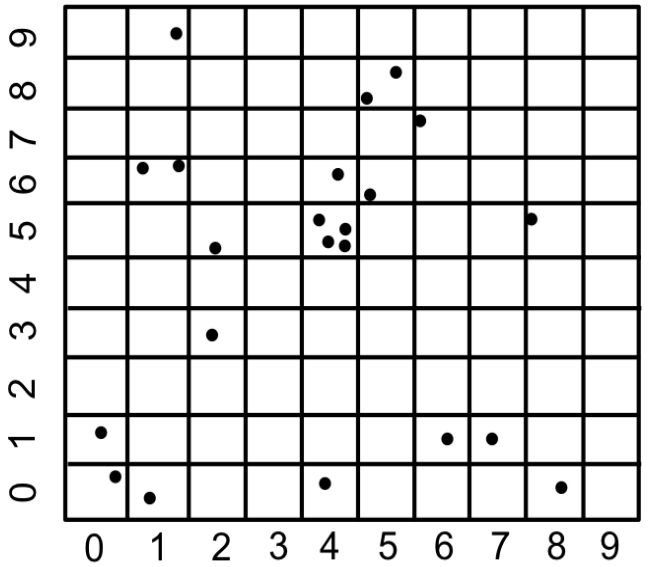
Various spatial indexing techniques include

- Grid Indexing
- Z-Ordering and Geohash
- K-d Tree
- Quad Tree
- R-Tree
- H3 Indexing

Grid Indexing

- We have little information about the location of the data. Roughly, where it is, some kind of bounds
- Convert the space into a uniform grid
- We don’t need to store the the grid cells. They can be computed.
- Each grid cell points to a disk page, which stores the points in that cell.
- If a disk page corresponding to a cell is filled up, they are chained like the leaves of a B+tree.
- The index file translates the grid cells to disk pages





Grid Indexing

Advantages

- Very simple
- Can utilize a hash table to make the actual index file smaller
- If the grid is dense in all parts, no empty cells !!

Disadvantages

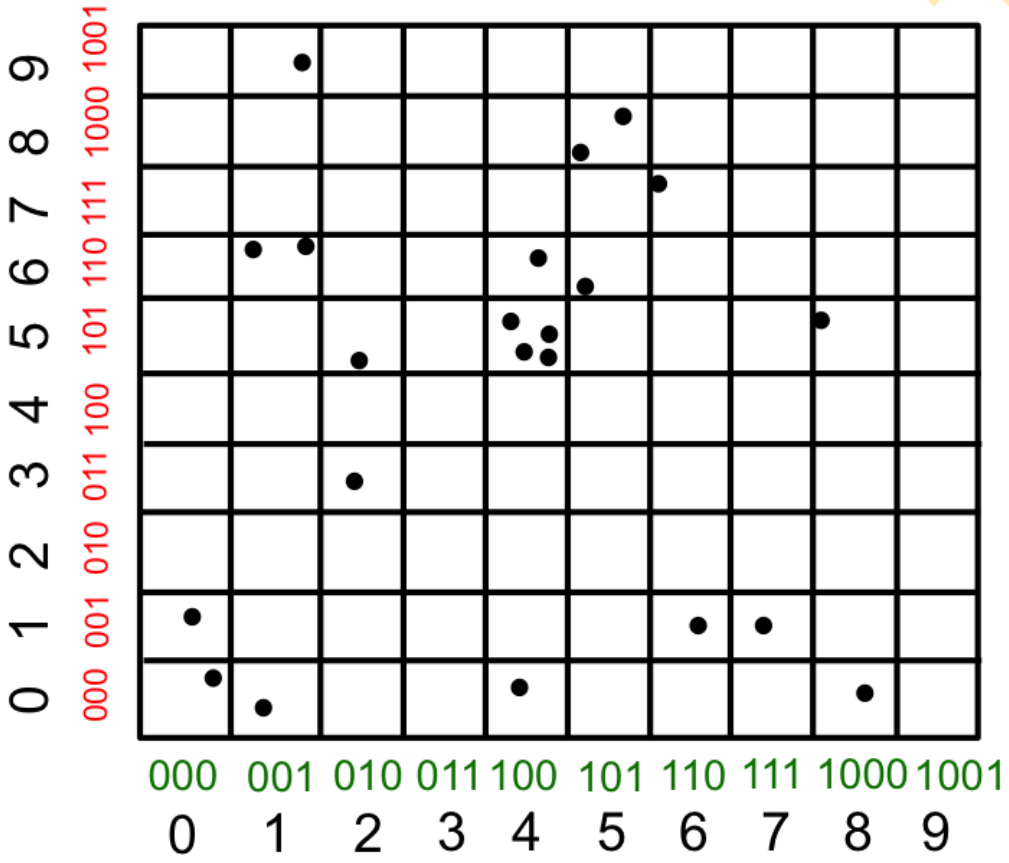
- Grids are usually sparse, resulting in many empty cells
- O(n^2) space in the index file
- The disk pages for each cell are not necessarily stored near each other on the disk. Disk is a one-dimensional structure, we are storing two-dimensional structure
- Adjacent cells in a grid index cannot be adjacent on disk

Grid Indexing

- How to insert a new point or data element?
- How to delete a point or data element from a grid index?
- How to perform range search with grid-based index?
- How to perform KNN search with grid-based index?

Z-Ordering

- Maps 2 or higher dimensional ordering to a lower dimensional ordering
- Works by interleaving the bit representation of a cell address
- An identifier for each cell is generated by interleaving bit addresses of longitude and latitude. Even bits from longitude and odd bits from latitude
- Produces a linear ordering of cells



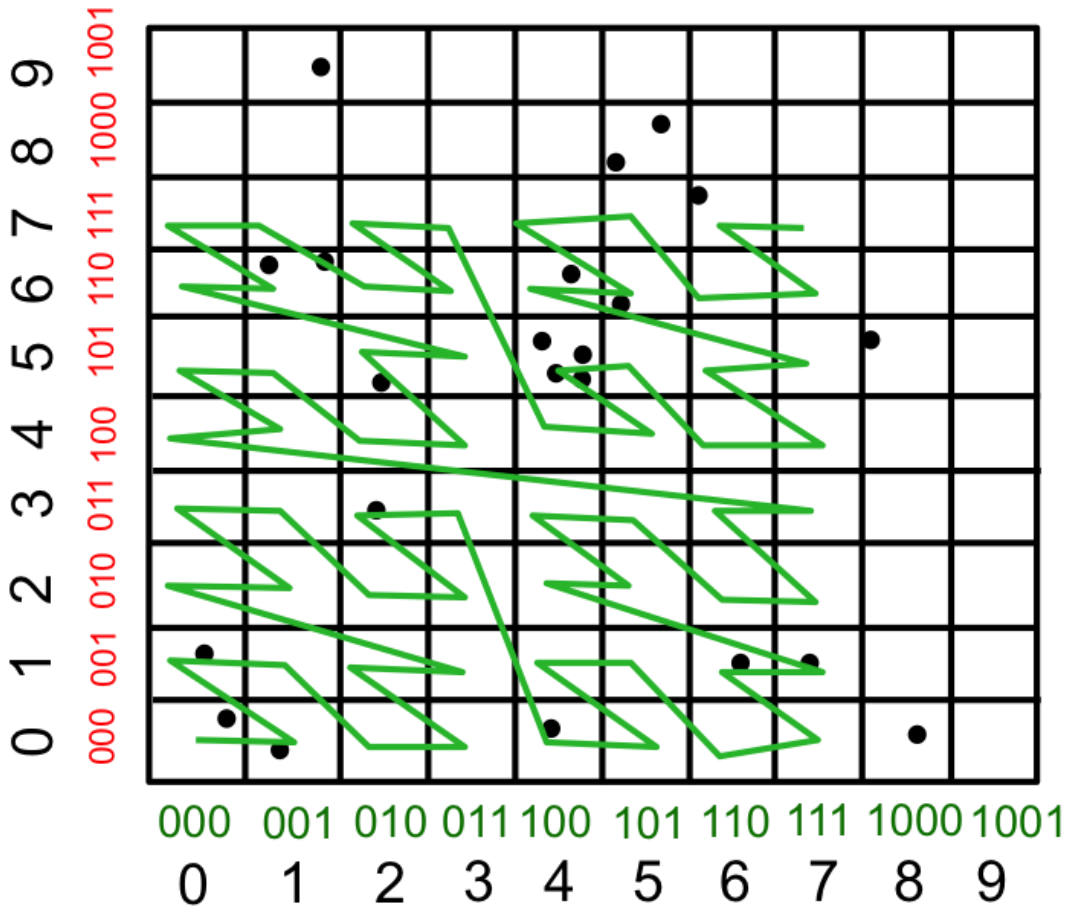
Representing Cells (1, 0) and (2, 1)

0 0 0 0 1 0 = (1, 0)

0 0 0 1 1 0 = (2, 1)

Z-Ordering

Linear ordering of cells produced by cell identifiers has a shape similar to Z



Z-Ordering

Advantages

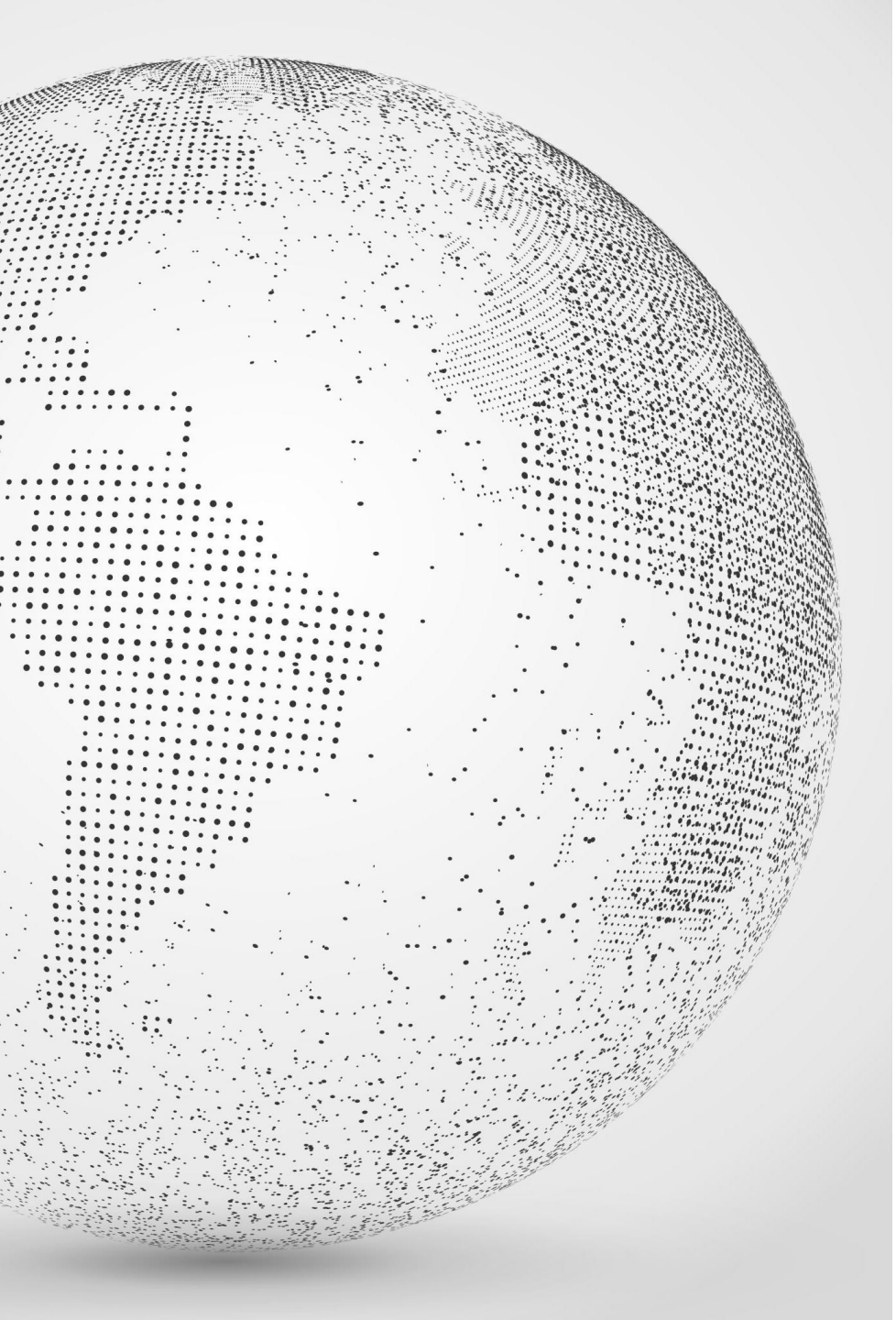
- Instead of random cell placement on disk, follow some kind of ordering
- Helps placing neighboring cells close to each other on disk
- Improve seek time of disk pages

Disadvantages

- Cannot solve the sparsity problem of grid cells
- Cells are static, sensitive to the data skew. A lot of data might exist in a small area.

Z-Ordering

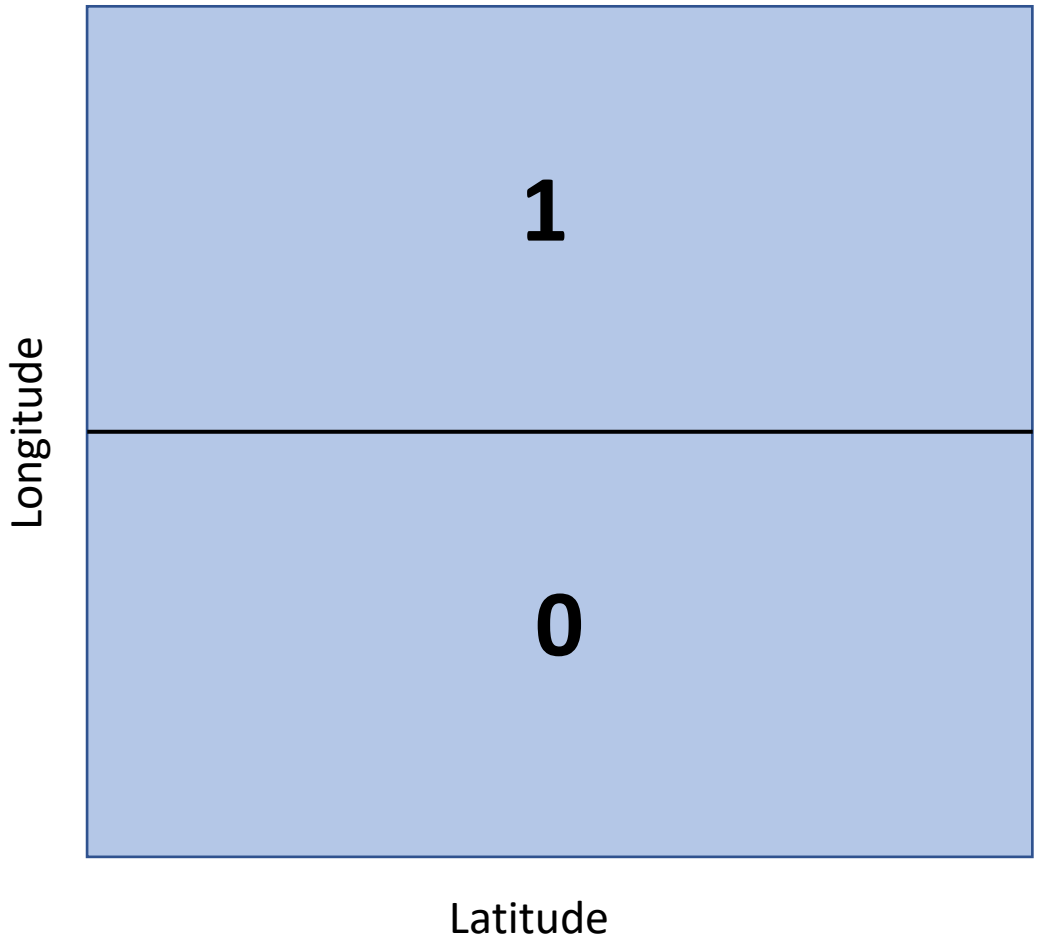
- How to insert a new point or data element?
- How to delete a point or data element from a grid index?
- How to perform range search with Z-Ordering index?
- How to perform KNN search with Z-Ordering index?

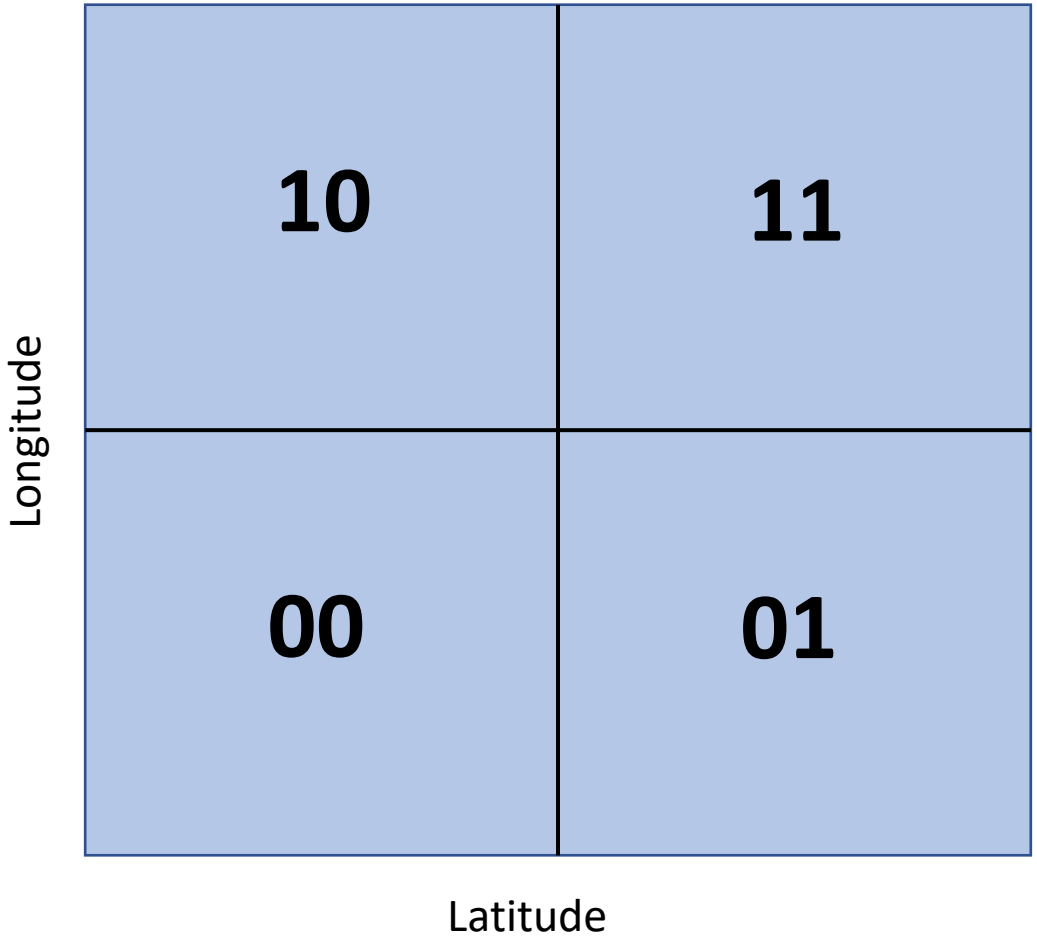


Geohashing

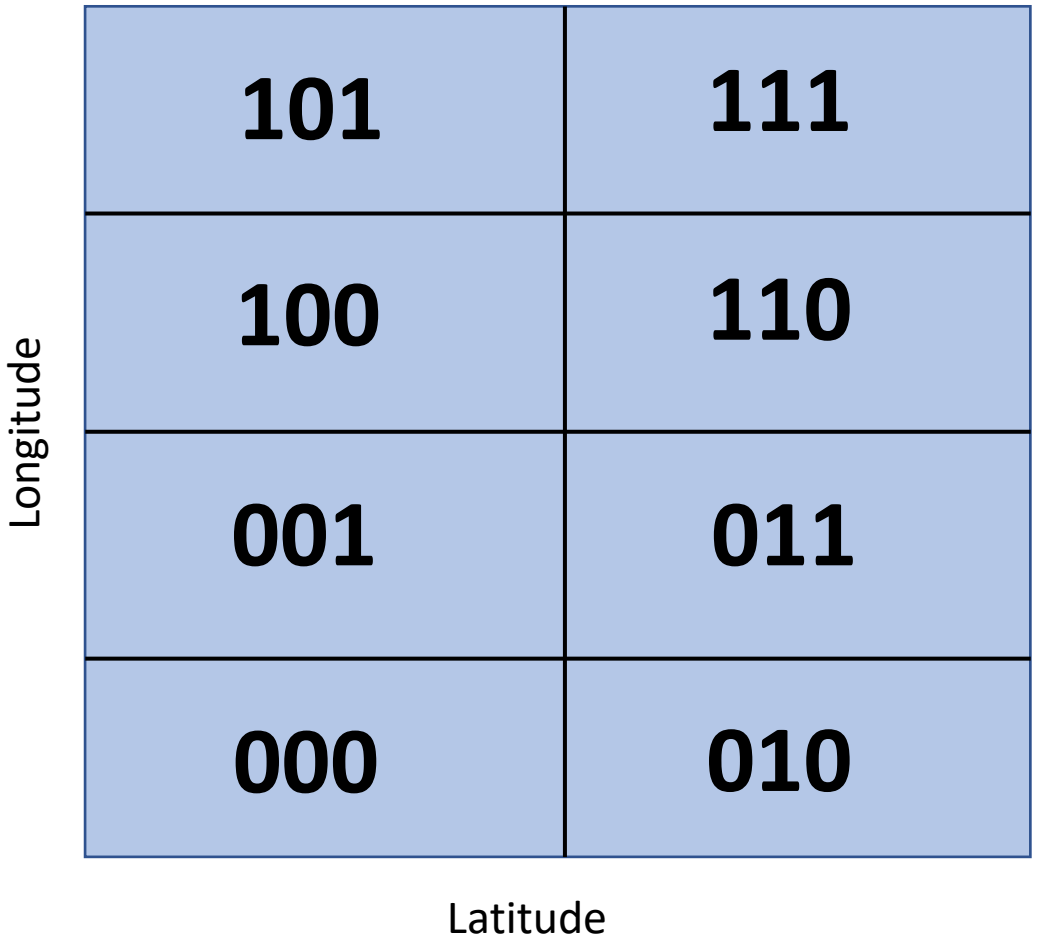
- An application Z-Ordering of space
- Reduces a two-dimensional latitude and longitude pair into a single alpha numeric string by employing a base-32 encoding
- Each additional bit adds precision to the location. Precision factor determines the size of the cell
- The world is recursively divided into smaller and smaller grids with each additional bit
- The first bit bisects the longitude, the next one latitude, the next longitude, etc.

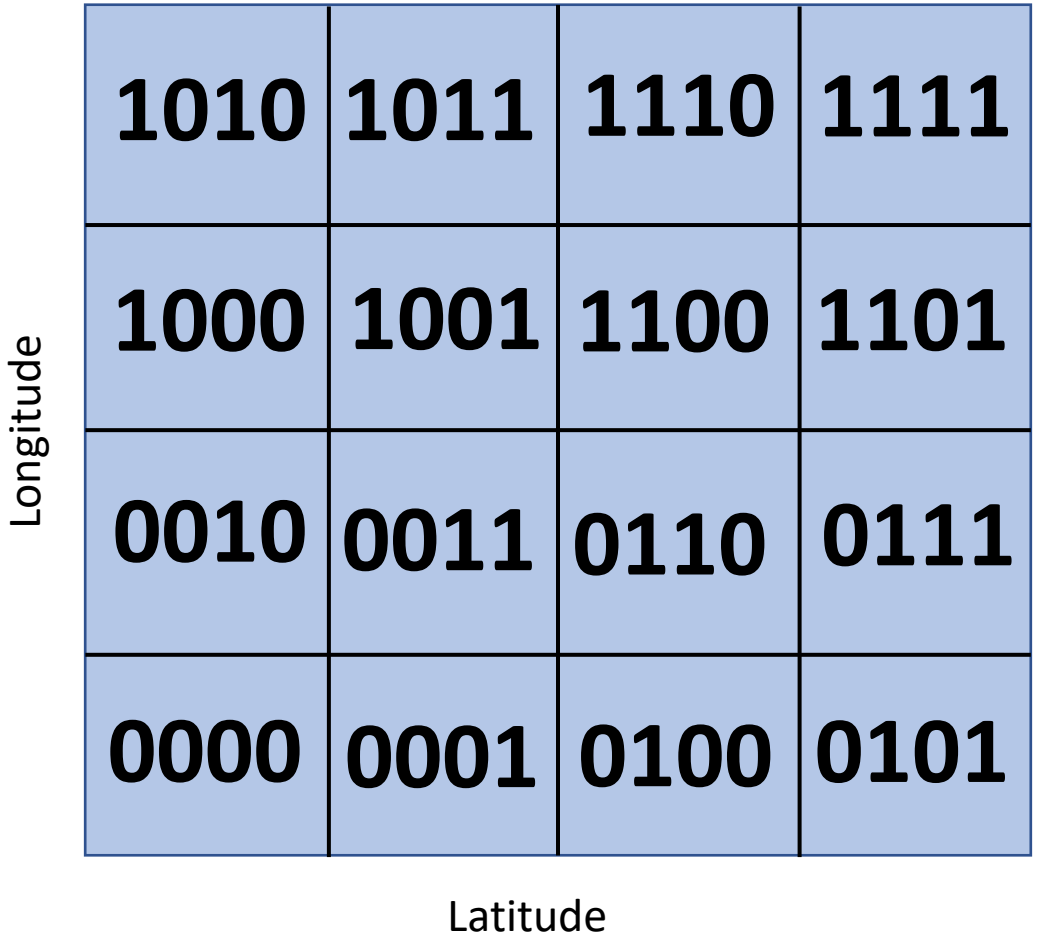
Constructing Geohashes





Constructing Geohashes





Constructing Geohashes

- Convert the binary representation of grid cells into base-32 strings

Binary	01001	10110	01000	11110	11110
Decimal	9	22	8	30	30
Base 32	9	q	8	y	y

Decoding a Geohashes

- Think of geohash string as interleaved longitude (even bits) and latitude (odd bits).

Base 32	9	q	8	y	y
Decimal	9	22	8	30	30
Binary	01001	10110	01000	11110	11110
Longitude	0-0-1	-0-1-	0-0-0	-1-1-	1-1-0
Latitude	-1-0-	1-1-0	-1-0-	0-1-0	-1-1-

Finding Neighbors with Geohashes

- Locations starting with the same prefix are within the same region and neighbors, but the converse is not true
- Nearby locations can be found through simple string matching

City	Geohash	Latitude	Longitude
San Francisco	9q8yym901hw	37.77926	-122.41923
Oakland	9q9p1d5zfks	37.80531	-122.27258
Berkeley	9q9p3tvj8uf	37.86947	-122.27093
Los Angeles	9q5ctr60zyr	34.05366	-118.24276
New York City	dr5regw2z6y	40.71273	-74.00599
London	gcpvn0ntjut	51.50479	-0.07871
Greenwich	u10hb5403uy	51.47651	0.00283

How to find 8 neighbors from a Geohash Code?

Finding 8 Neighbors of a Geohash Code

b	**c**	**f**	**g**	**u**	**v**	**y**	**z**
8	9	d	e	s	t	w	x
2	3	6	7	k	m	q	r
o	1	4	5	h	j	n	p

How to search locations within a distance using Geohash?

Disadvantages of Geohashing?

K-d Tree

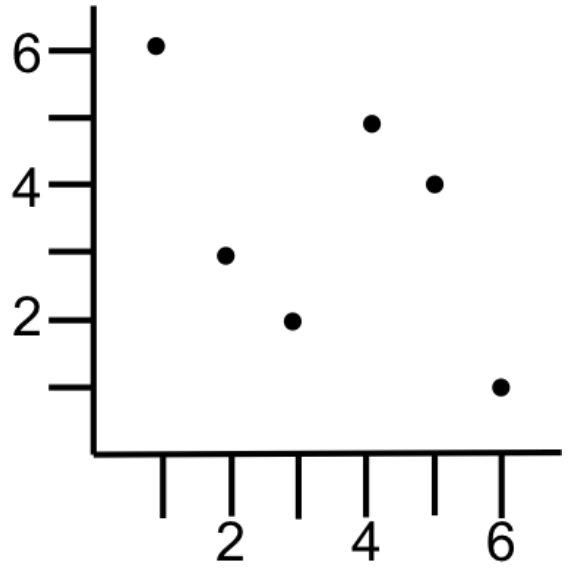
- Stands for k dimensional tree
- An adaptive data structure that will adjust to fit the data
- It is a binary tree in which every node represents a point in one of k dimensions
- Non-leaf nodes divide a hyperplane into two half spaces

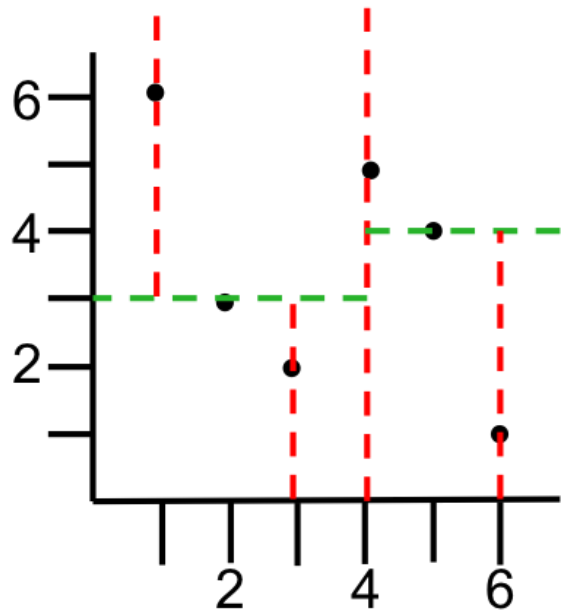
Constructing K-d Tree

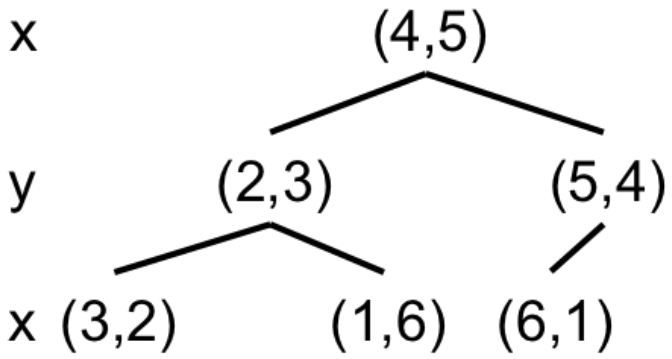
1. Start with one of k dimensions
2. Find the median point along the selected dimension
3. Add the points located in the left of median to the left subtree and points in the right to the right subtree
4. Repeat Steps 2 and 3 for other dimensions one after another
5. Repeat steps 1-4 until no points are left

Build a K-d tree with the points (1, 6), (2, 3), (3, 2), (4, 5), (5, 4), (61)

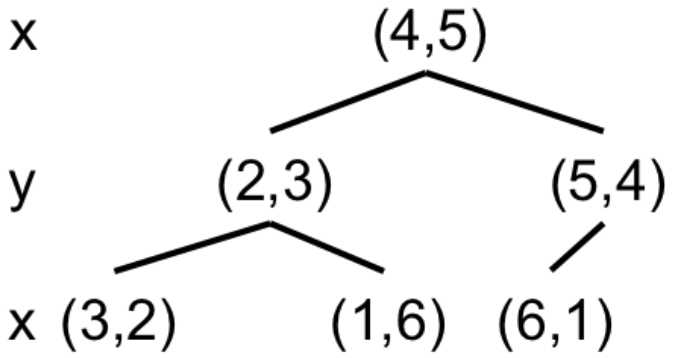
Constructing K-d Tree







Find Rectangular Region containing a Point with K-d Tree



Find the rectangle containing the point (3.5, 3.5)

- Less than (4, 5) on x split, so the rectangle (lat1, lat2, lon1, lon2) = (lat1, 4, lon1, lon2)
- Greater than (2, 3) on y split, so the rectangle (lat1, lat2, lon1, lon2) = (lat1, 4, 3, lon2)
- Greater than (1, 6) on x split, so the rectangle (lat1, lat2, lon1, lon2) = (1, 4, 3, lon2)

The containing rectangle range is from 1 to 4 in x direction, and from 3 to +infinity in the y direction

Insertion and Deletion with K-d Tree

Insertion

- Follow the same binary search tree procedure to find the insertion point
- Insertions occur at leaves

Deletion

- Find the node to delete
- Rebuild the tree consisting of all children of the deleted node
- Do a binary search tree style delete

What is the time complexity of building K-d tree?

K-d Tree

Advantages

- Adaptive to the distribution of data
- Logarithmic depth makes it efficient for search and KNN operations

Disadvantages

- Not balanced, data updates may require to rebuild the whole tree
- Curse of dimensionality occurs for high dimensional data. Finding the nearest point is an O(logn) operation on average. The algorithm needs to visit too many branches in high dimensional data
- A node can hold only one element/point