Chapter 8

Computational geometry

-Niraj K.C.

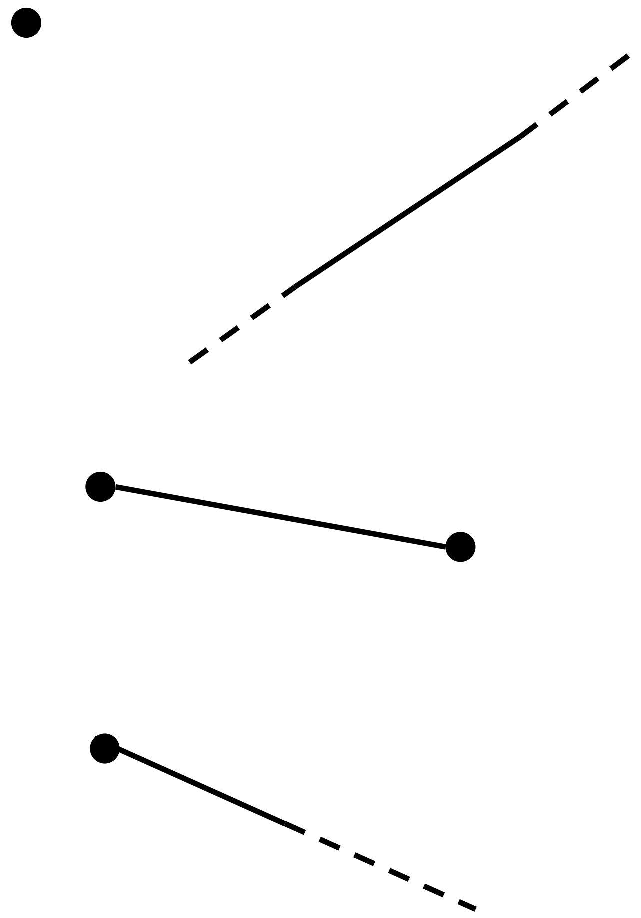
Table of Content

What is Computational Geometry?

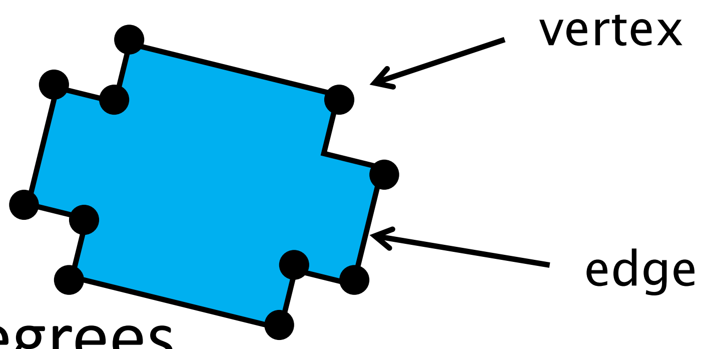
- ➢Deals with geometrical structures
    - ▪ Points, lines, line segments, vectors, planes, etc.

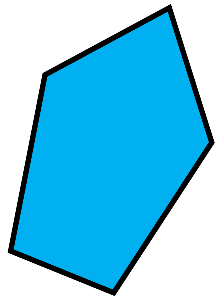
Basic definitions

- ➢Point
  - Specified by two coordinates (x, y)
- ➢Line
  - Extends to infinity in both directions
- ➢Line segment
  - Specified by two endpoints
- ➢Ray
  - Extends to infinity in one direction



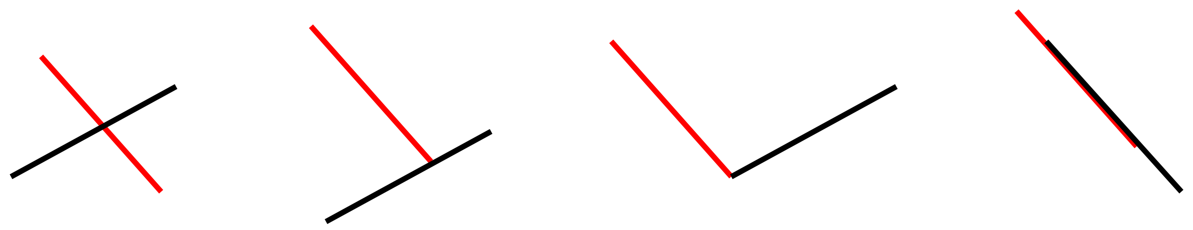
- Polygon
  - We assume edges do not cross
- Convex polygon
  - Every interior angle is at most 180 degrees
  - Precise definition of *convex*: For any two points inside the polygon, the line segment joining them lies entirely inside the polygon



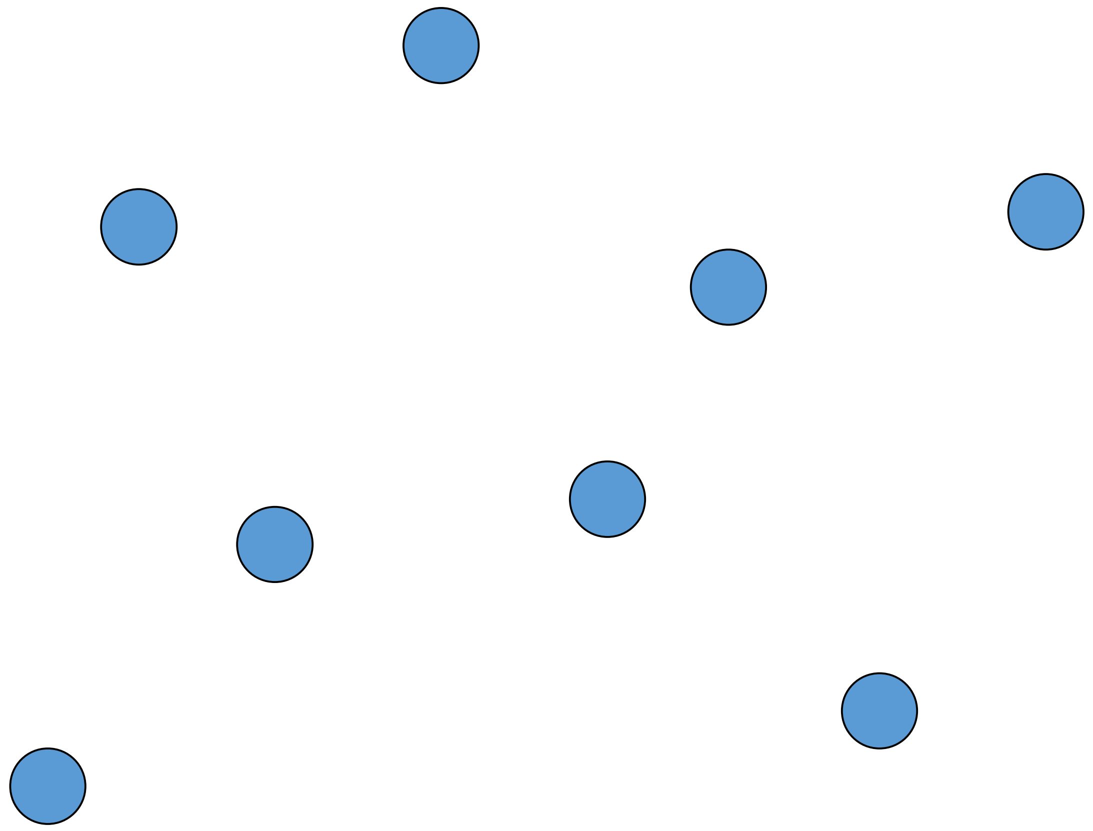


What makes CoGeom problems so annoying?

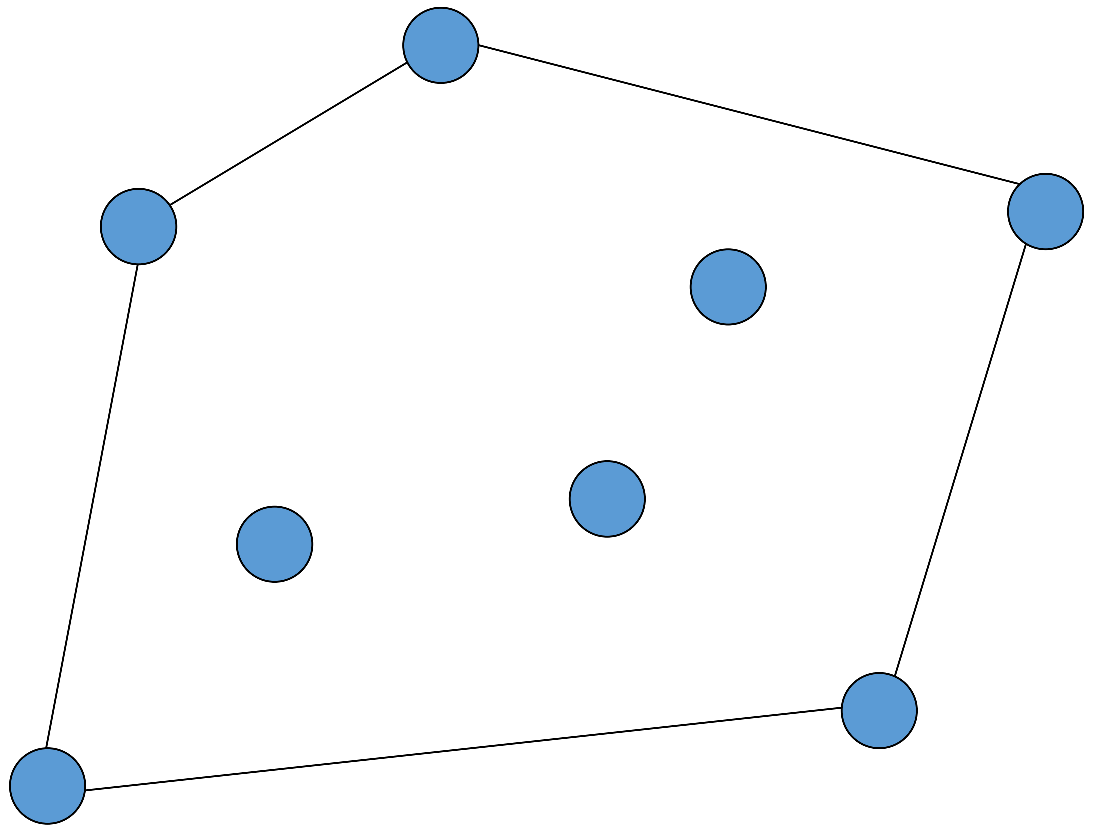
- Precision error
  - Avoid floating-point computations whenever possible.
- Degeneracy
  - Boundary cases
  - For example, imagine how two line segments can intersect



Convex Hull



Convex Hull



Convex Hull: Identifying

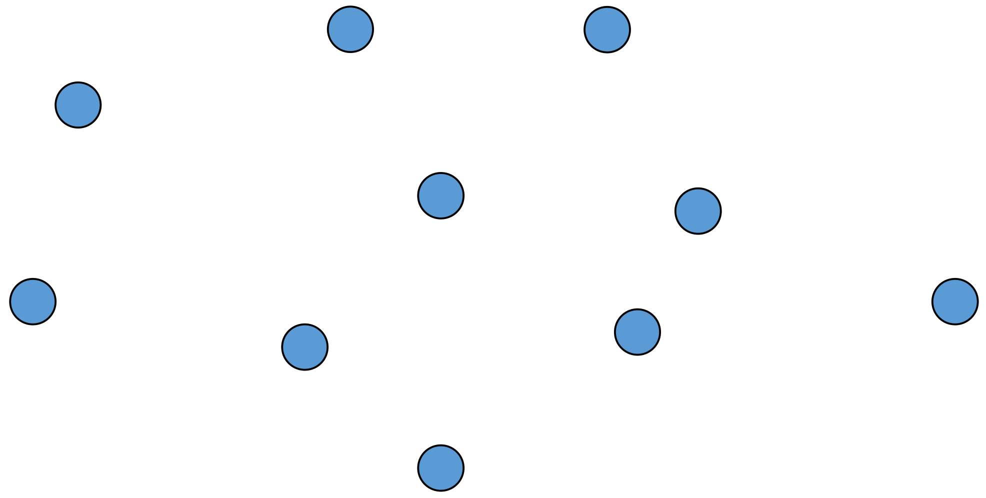
- Two points farthest distance from each other.
- Minimum set of points enclosing…
- Ideas of fencing, encompassing or enclosing
- Diameter, farthest pair, longest distance between…

Algorithms

- Jarvis March
- Graham Scan

Jarvis March

- Pick a point that you know to be on the convex hull.
- To determine the next point on the hull, loop through all points and find the one that forms the minimum sized anticlockwise angle off the horizontal axis from the previous point.
- Continue until you encounter the first point



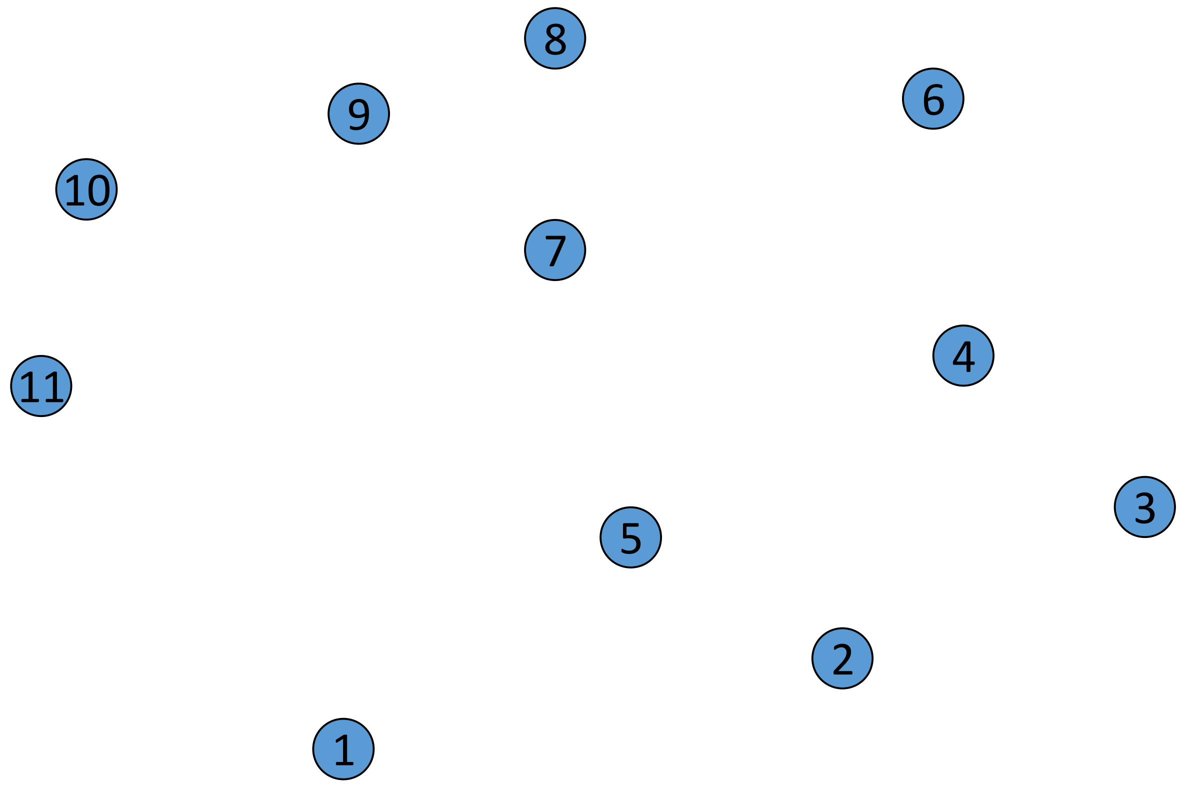
Pros and Cons

- Pros
  - Easy to program
- Cons
  - Slow!

Graham Scan

- Find a point you know to be on the convex hull. (Lower y coordinate)
- Sort all other points angularly around this point (anti clockwise), by calculating the angle that each point makes with the x axis (within the range 0 to 360 degrees)
- Add the first two points to the hull.
- For every other point except the last point
- Make it the next point in the convex hull
- Check to see if the angle it forms with the previous two points is greater than 180 degrees
    - As long as the angle formed with the last two points is greater than 180 degrees, remove the previous point
- To add the last point
    - Perform the deletion above,
    - Check to see if the angle the last point forms with the previous point and the first point is greater than 180 degrees or if the angle formed with the last point and the first two points is greater than 180 degrees.
    - If the first case is true, remove the last point, and continue checking with the next-to-last point.
    - If the second case is true, remove the first point and continue checking.
    - Stop when neither case is true.

Graham Scan



Pros and Cons

- Pros
  - Fast
  - Relatively Easy to Program
- Cons
  - Bit fiddly to program

Robust Geometric Primitives

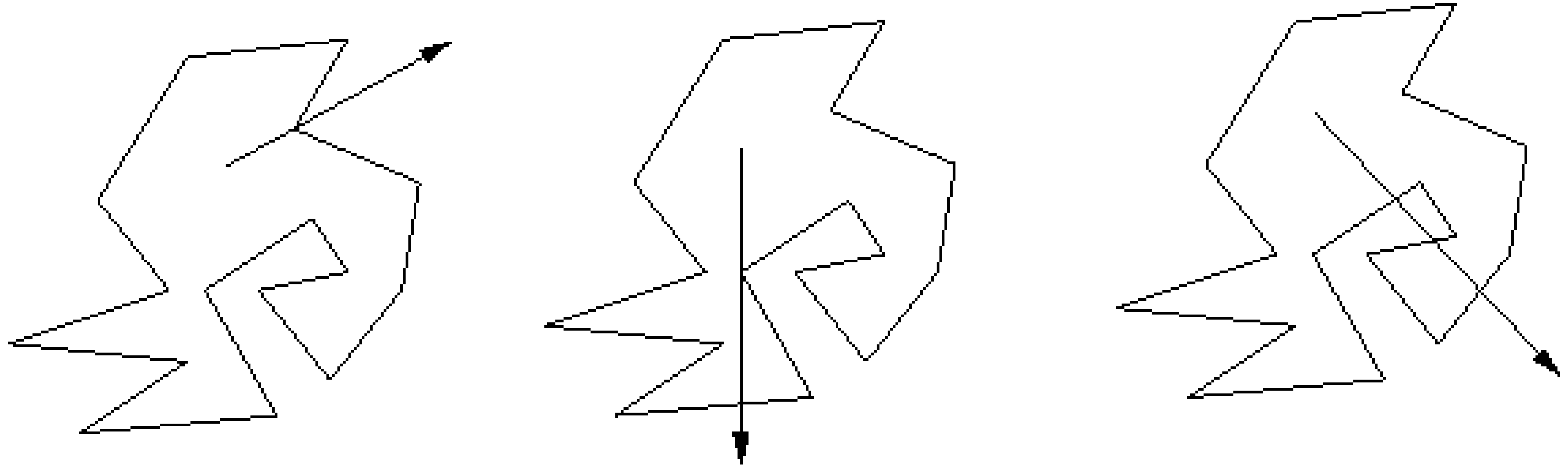
- ➢Area of a Triangle
- ➢Point in Polygon
- ➢Area of a Polygon
- ➢Pitfalls, Arithmetic Accuracies, Exceptions.

Area of a Triangle

- Heron’s Formula
  - Area = Sqrt(s(s-a)(s-b)(s-c)), where s = (a+b+c)/2
  - Determinant / Cross Product

Point in Polygon

- Extend the point in a random direction forming a ray.
- Find the number of times the ray intersects an edge of the polygon.
- Even number of intersections = outside
- Odd number of intersection = inside



Algorithm generalizes to 3 dimensions!

Area of Polygon

Area of polygon
The area of a polygon with vertices (*x* 1, *y* 1), ..., (*x* n, *y* n) is equal to the determinant:

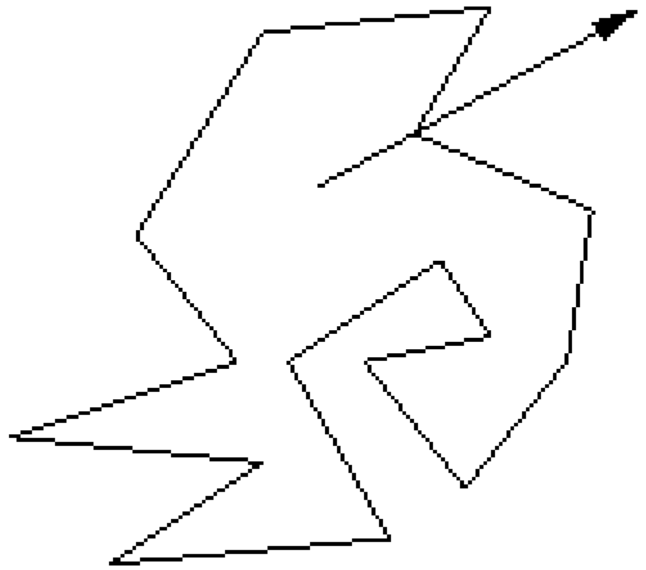
```
 1   | x1 x2 . . . xn  |
---  |                 |
 2   |  y1 y2 . . . yn |
```

This formula generalizes to compute d! times the volume of a simplex in d dimensions.

Areas and volumes are signed!

Area of a Polygon

- Triangulation
- Monte Carlo Methods



Data Structures & Algorithms

- ➢Implementation of spatial algebra in an integrated manner with the DBMS query processing.
- ➢Not just simply implementing atomic operations using computational geometry algorithms, but consider the use of the predicates within set-oriented query processing **Spatial indexing** or access methods, and **spatial join** algorithms

Data Structures

- ➢Representation of a value of a SDT must be compatible with two different views:

1. DBMS perspective:
    - Same as attribute values of other types with respect to generic operations
    - Can have varying and possibly large size
    - Reside permanently on disk page(s)
    - Can efficiently be loaded into memory
    - Offers a number of type-specific implementations for generic operations needed by the DBMS (e.g., transformation functions from/to ASCII or graphic)

2. Spatial algebra implementation perspective, the representation:

- ➢Is a value of some programming language data type
- ➢Is some arbitrary data structure which is possibly quite complex
- ➢Supports efficient computational geometry algorithms for spatial algebra operations
- ➢Is not geared only to one particular algorithm but is balanced to support many operations well enough

- From both perspectives, the representation should be mapped by the compiler into a single or perhaps a few contiguous areas (to support DBMS paging). Also supports:
  - Plane sweep sequence: object’s vertices stored in a specific sweep order (e.g. x-order) to expedite plane-sweep operation.
  - Approximations: stores some approximations as well (e.g. MBR) to speed up operations (e.g. comparison)
  - Stored unary function values: such as perimeter or area be stored once the object is constructed to eliminate future expensive computations.

Spatial Indexing

- To expedite spatial selection (as well as other operations such as spatial joins, …)
- It organizes space and the objects in it in some way so that only parts of the objects need to be considered to answer a query.
- Two main approaches:
    1. Dedicated spatial data structures (e.g. R-tree)
    2. Spatial objects mapped to a 1-D space to utilize standard indexing techniques (e.g. B-tree)

Spatial Indexing: operations

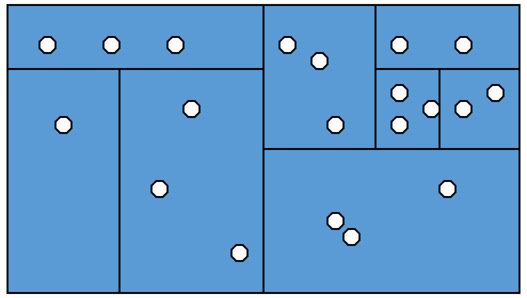
- Spatial data structures either store *points* or *rectangles* (for line or region values)
- Operations on those structures: insert, delete, member
- Query types for points:
  - Range query: all points within a query rectangle
  - Nearest neighbor: point closest to a query point
  - Distance scan: enumerate points in increasing distance from a query point.
- Query types for rectangles:
  - Intersection query
  - Containment query

Spatial Indexing: idea… approximate!

- A fundamental idea: use of approximations as keys
  - continuous (e.g. bounding box)
  - Grid (a geometric entity as a set of cells).
- Filter and refine strategy for query processing:
  - Filter: returns a set of candidate object which is a superset of the objects fulfilling a predicate
  - Refine: for each candidate, the exact geometry is checked

Spatial Indexing: memory organization

- A spatial index structure organizes points into buckets.
- Each bucket has an associated *bucket region*, a part of space containing all objects stored in that bucket.
- For point data structures, the regions are disjoint & partition space so that each point belongs into precisely one bucket.
- For rectangle data structures, bucket regions may overlap.



A kd-tree partitioning of

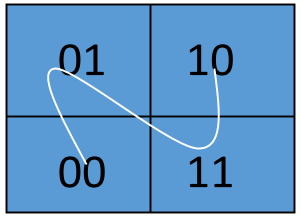
2d-space

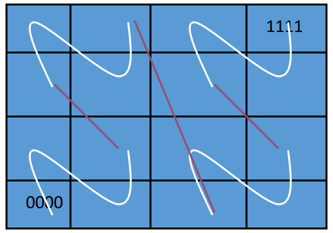
where each bucket can

hold up to 3 points

Spatial Indexing: 1-D Grid approx.

- One dimensional embedding: z-order or bit-interleaving
  - Find a linear order for the cells of the grid while maintaining "locality" (i.e., cells close to each other in space are also close to each other in the linear order)
  - Define this order recursively for a grid that is obtained by hierarchical subdivision of space





Spatial Join

- Traditional join methods such as hash join or sort/merge join are not applicable.
- Filtering cartesian product is expensive.
- Two general classes:
  1. Grid approximation/bounding box
  2. None/one/both operands are presented in a spatial index structure
- Grid approximations and overlap predicate:
  - A parallel scan of two sets of z-elements corresponding to two sets of spatial objects is performed
  - Too fine a grid, too many z-elements per object (inefficient)
  - Too coarse a grid, too many “false hits” in a spatial join

Spatial Join

- Bounding boxes: for two sets of rectangles R, S all pairs *(r,s), r in* R, *s in* S, such that *r* intersects *s*:
  - **No spatial index on R and S**: *bb_join* which uses a computational geometry algorithm to detect rectangle intersection, similar to external merge sorting
  - **Spatial index on either R or S**: *index join* scan the non-indexed operand and for each object, the bounding box of its SDT attribute is used as a search argument on the indexed operand (only efficient if non-indexed operand is not too big or else bb-join might be better)
  - **Both R and S are indexed**: synchronized traversal of both structures so that pairs of cells of their respective partitions covering the same part of space are encountered together.

System Architecture

- Extensions required to a standard DBMS architecture:
    - Representations for the data types of a spatial algebra
    - Procedures for the atomic operations (e.g. overlap)
    - Spatial index structures
    - Access operations for spatial indices (e.g. insert)
    - Filter and refine techniques
    - Spatial join algorithms
    - Cost functions for all these operations (for query optimizer)
    - Statistics for estimating selectivity of spatial selection and join
    - Extensions of optimizer to map queries into the specialized query processing method
    - Spatial data types & operations within data definition and query language
    - User interface extensions to handle graphical representation and input of SDT values

System Architecture

- The only clean way to accommodate these extensions is an integrated architecture based on the use of an extensible DBMS.
- There is no difference in principle between:
  - a standard data type such as a STRING and a spatial data type such as REGION
  - same for operations: concatenating two strings or forming intersection of two regions
  - clustering and secondary index for standard attribute (e.g. B-tree) & for spatial attribute (R-tree)
  - sort/merge join and bounding-box join
  - query optimization (only reflected in the cost functions)

Assignment 7

1. What do you understand by computational geometry? Define point, line , line segment, polygon ,ray and convex polygon with suitable diagram.
2. What makes computational geometry problem so annoying? Define convex Hull and its algorithm in detail.
3. Describe robust geometric primitives in detail.
4. Describe data structures and algorithms for spatial databases.
5. Describe system architecture for SDBMS in detail.