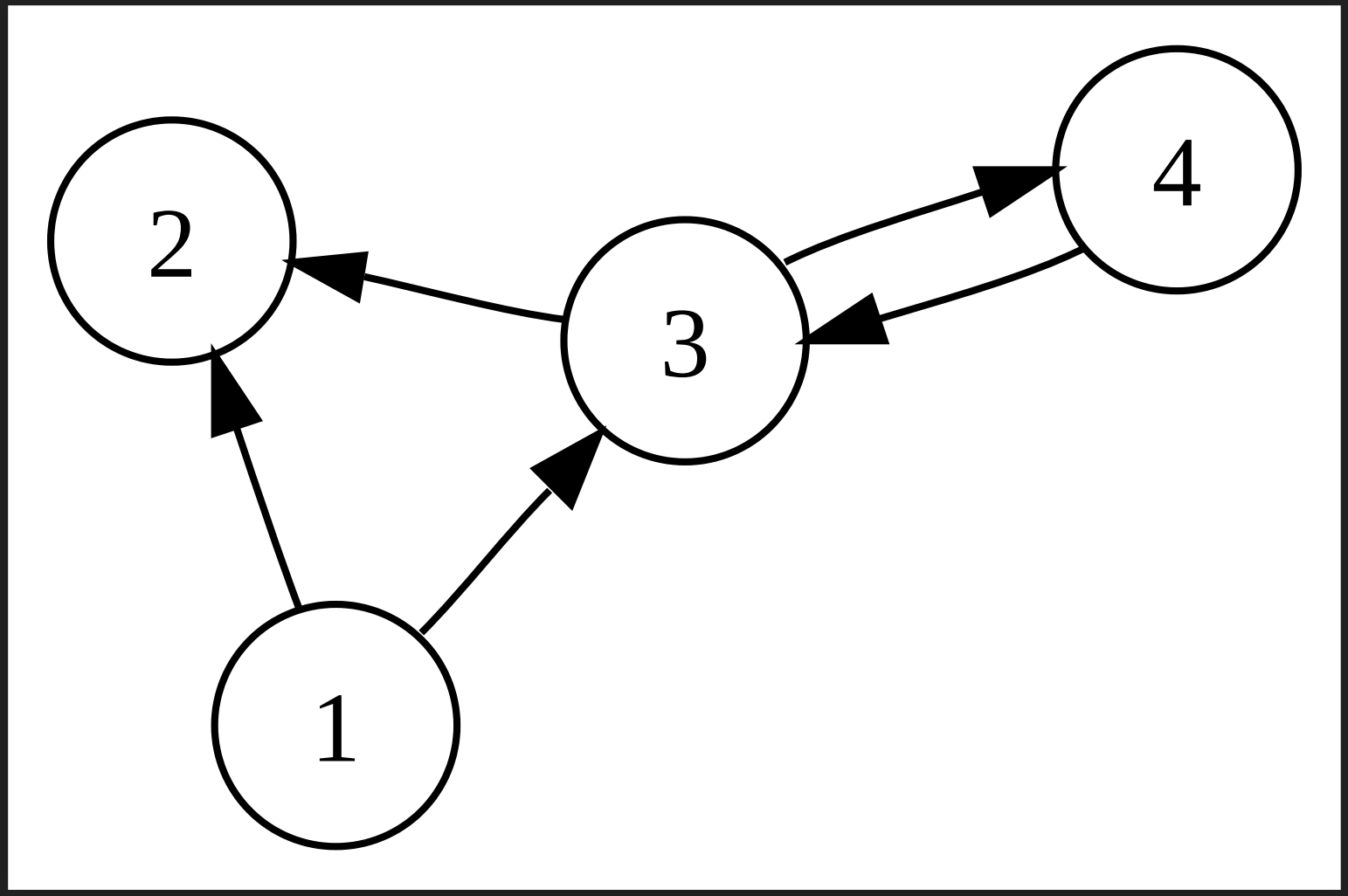
GRAPHS & GEOMETRIC ALGORITHMS

Graph is a collection of things (vertices) that have relationships between them (edges).



GRAPHS ARE SUPER USEFUL, BECAUSE THEY REPRESENT MANY PRACTICAL PROBLEMS:

- Maps (vertex = city, edge = road)
  - How can I get from Ostrava to Brno quickly? (GPS navigation)
- Social networks (vertex = person, edge = friend/follower)
  - Who's the most popular kid on the block?
- Computer networks (vertex = router, edge = Ethernet cable)
  - Which path to choose for a packet from Czech Republic to USA?
- Data structures (trees are graphs - binary, B, AVL, red-black etc.)
- Also used in: graph databases, compilers, language processing, sociology, biology (DNA matching), scheduling, ...

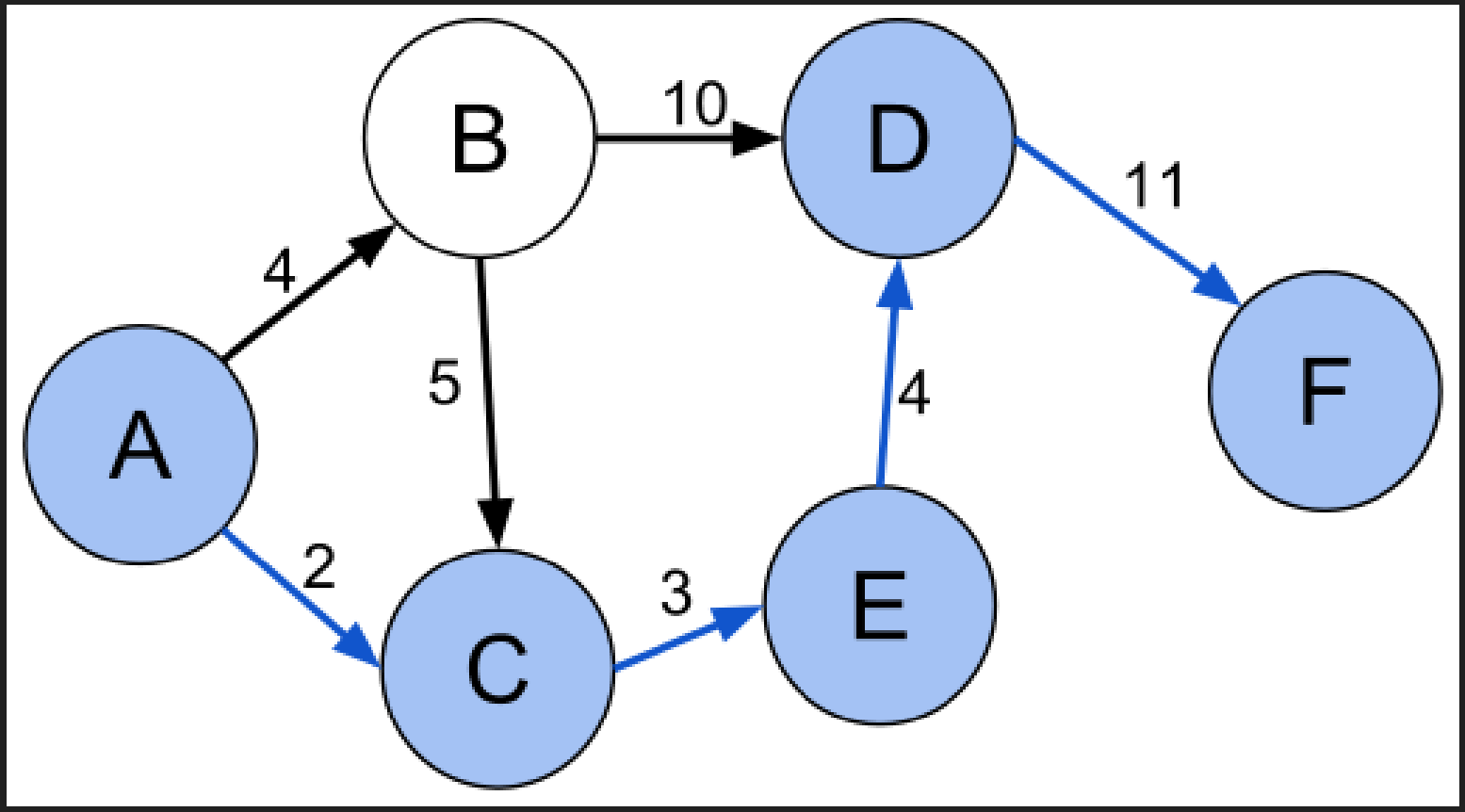
FINDING SHORTEST PATHS IN GRAPHS

- Traffic navigation - TomTom, GPS etc.
- AI pathfinding - robotics, drones, games (how to move NPC from point A to B)
- Network routing - routing protocols used by routers
- and many other uses

Single source shortest paths (SSSP)

- Find shortest path from vertex A to all other vertices
- Easy when edges have same length - BFS (breadth-first search)
- (Slightly) harder with variable edge lengths - Dijkstra's algorithm
- You should know this from ALG I/II and DIM

Shortest path from A to F: length 20, vertices [A, C, E, D, F]



D is a predecessor of F on the shortest path

E is a predecessor of D etc.

BFS REMINDER (PYTHON PSEUDOCODE)

```
def BFS(graph, SRC, DEST):
  put SRC into queue

  while queue not empty:
    V = get item from queue
    for all neighbours N of V:
      if N is not visited:
        put N into queue
        mark N as visited
        save somewhere that V is a predecessor of N

  if DEST not visited, return NOT_FOUND
  path = reconstruct predecessors from DEST to SRC
  return reversed path
```

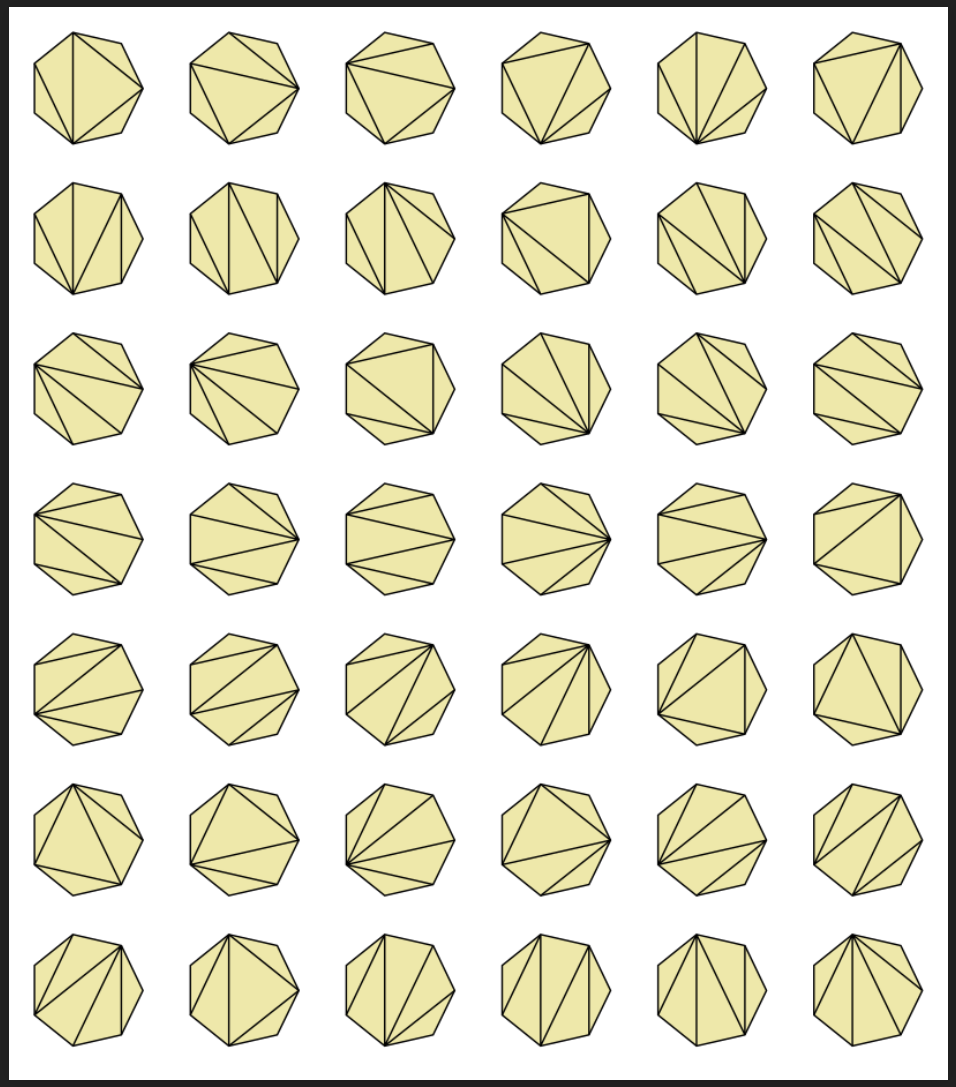
When a vertex is removed from the queue, a shortest path to it has been found.
Therefore if it's the DEST vertex, you can stop the search.

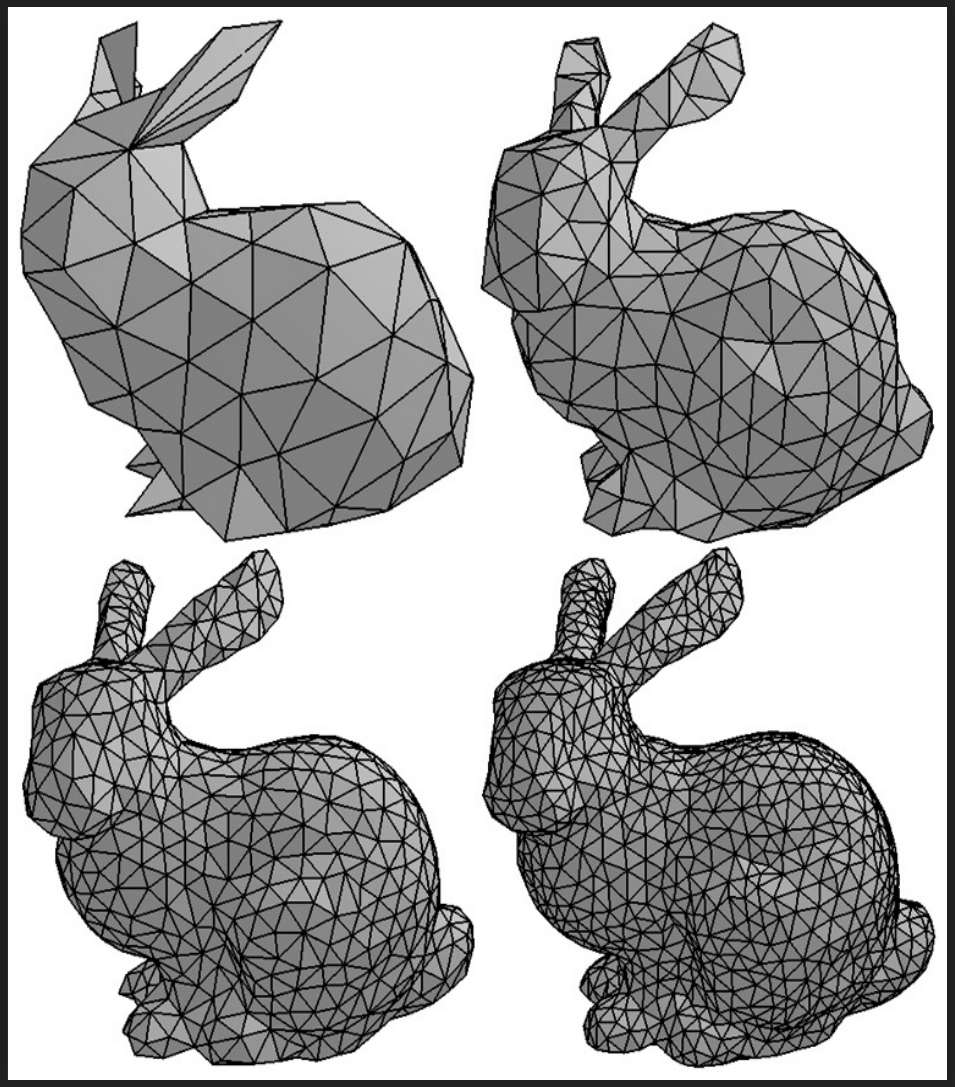
GEOMETRIC ALGORITHMS (COMPUTATIONAL GEOMETRY)

- Family of algorithms that use geometric formulations
- Terrain modelling, map operations, 3D rendering
- Intersect lines, query points in a map, triangulate a region
- More in AGU

TRIANGULATION

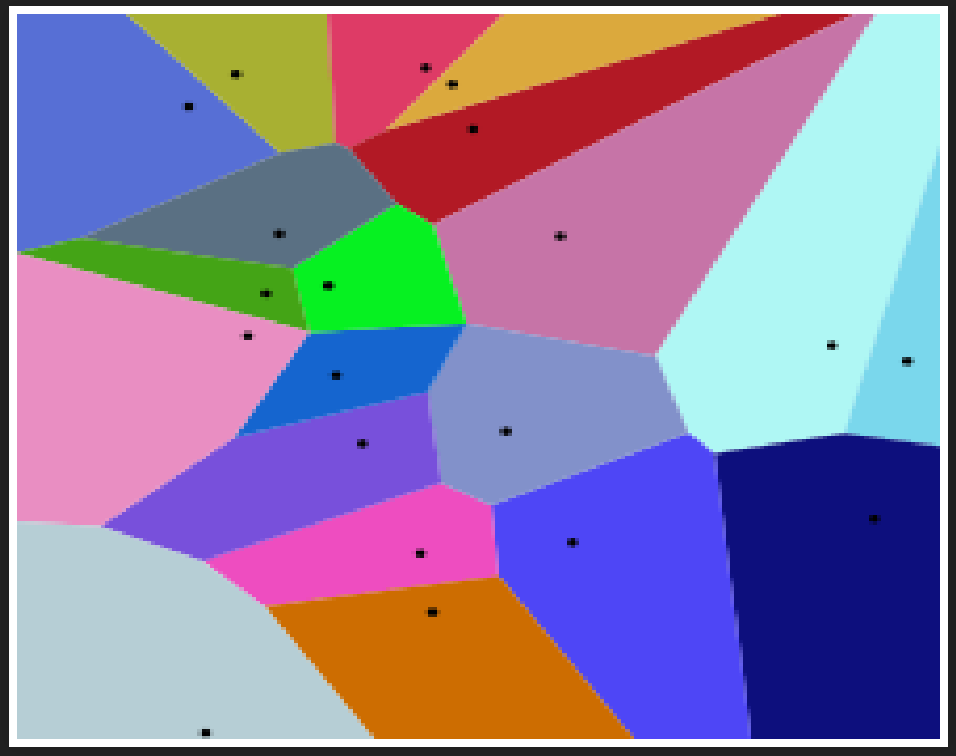
Split a set of points (polygon) into a set of triangles

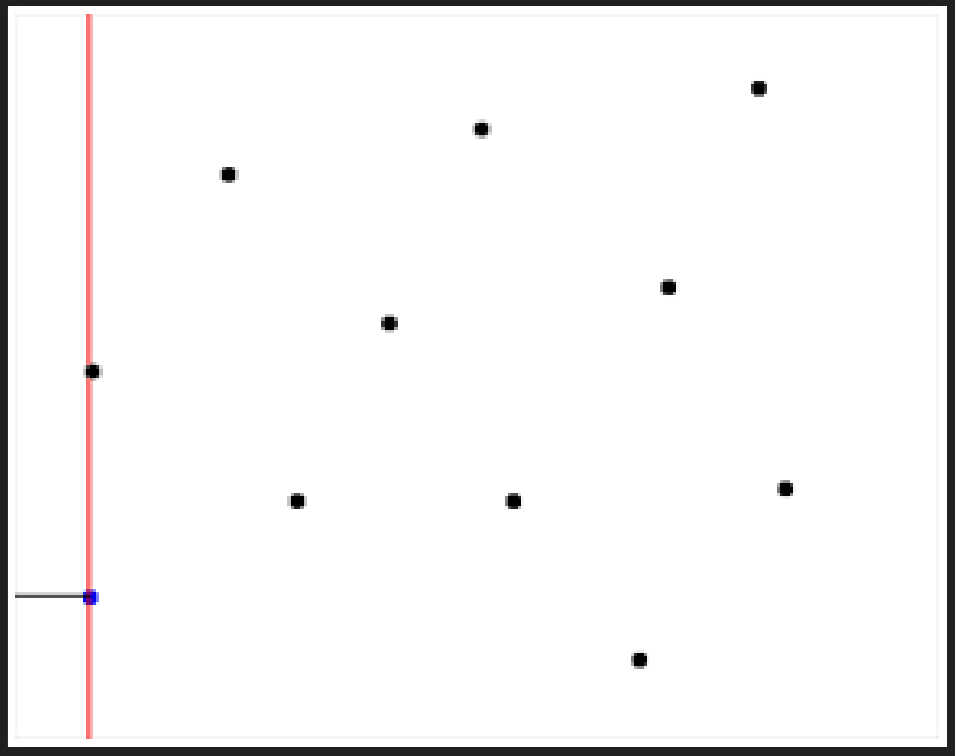




VORONOI DIAGRAM

Partition a plane into regions that are closest to each input point

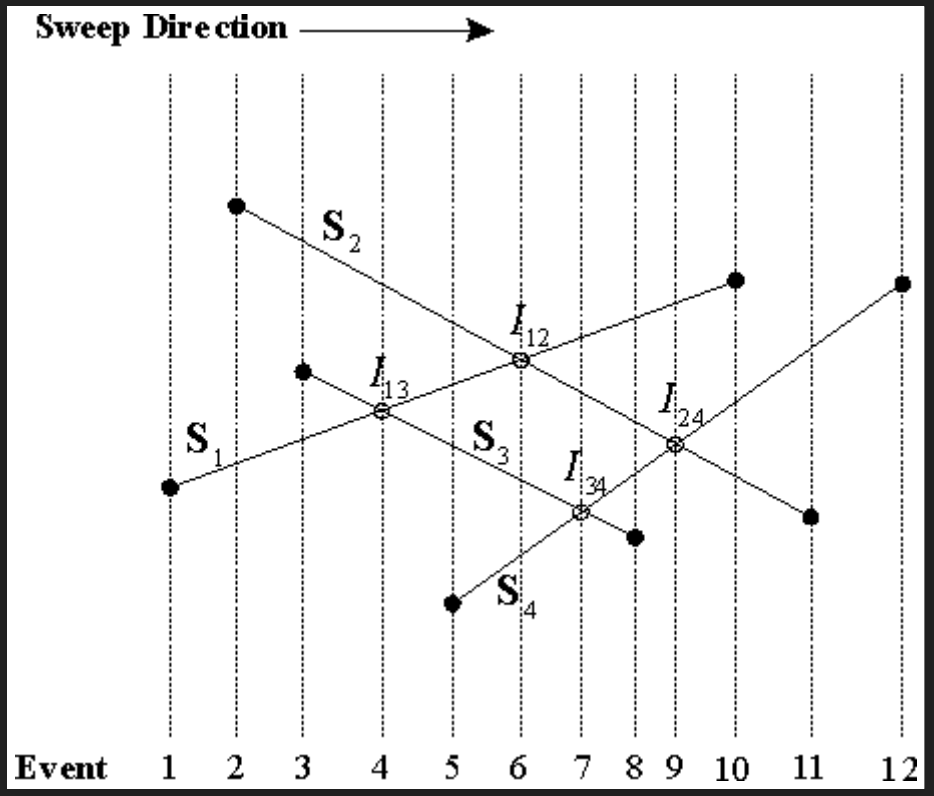




http://alexbeutel.com/webgl/voronoi.html

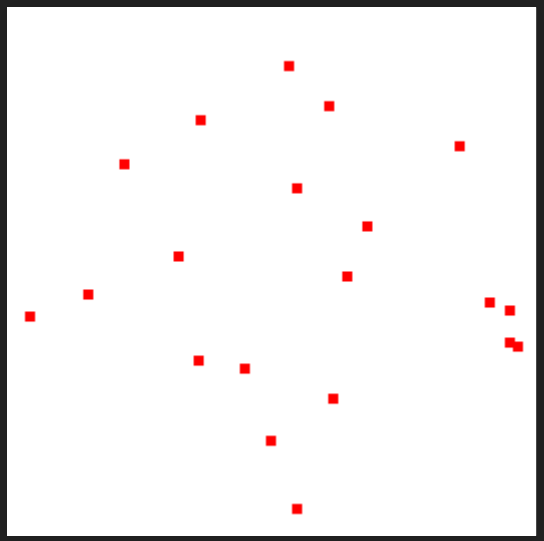
LINE SEGMENT INTERSECTION

Quickly calculate all intersections of a set of lines/segments



CONVEX HULL

Find a convex envelope of a set of points



GIFT WRAPPING (JARVIS MARCH)

- Input: a list of points (tuples of float (x, y))
- Output: a list of points that form the convex hull

Algorithm

GIFT WRAPPING (JARVIS MARCH)

- Input: a list of points (tuples of float (x, y))
- Output: a list of points that form the convex hull

1. Add the leftmost point to the convex hull
2. Traverse all points and find the one with the steepest angle from the last point added to CH
3. Add that point to CH
4. If that point was the first point added to CH, end
5. Otherwise go to 2)

Algorithm