Computer Graphics

Computational geometry: some applications in industry.

Prof. Eric Biagioli

Today

- Geometry simplification. (patented by Amazon Web Services in the Patents Office of USA).
- *Spatial index*, Amazon Redshift.
- An optimization for the KNN clustering algorithm. Amazon Redshift.

References for today

References for the class of today (part of the first partial exam)

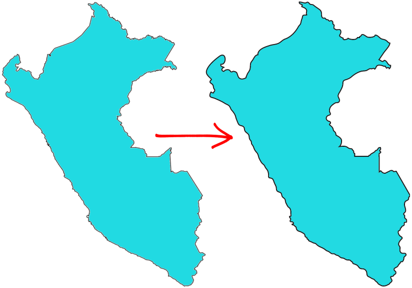
- BERG, M. D., CHEONG, O., KREVELD, M. V., AND OVERMARS, M.
  Computational Geometry: Algorithms and Applications, 3rd ed. ed.
  Springer-Verlag TELOS, Santa Clara, CA, USA, 2008.
- CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., AND STEIN, C.
  Introduction to Algorithms, Third Edition, 3rd ed.
  The MIT Press, 2009.
- MANOLOPOULOS, Y., NANOPOULOS, A., PAPADOPOULOS, A. N., AND THEODORIDIS, Y.
  R-Trees: Theory and Applications.
  Springer Publishing Company, Incorporated, 2005.

Today

- **Geometry simplification. (patented by Amazon Web Services in the Patents Office of USA).**
- *Spatial index*, Amazon Redshift.
- An optimization for the KNN clustering algorithm. Amazon Redshift.

Today

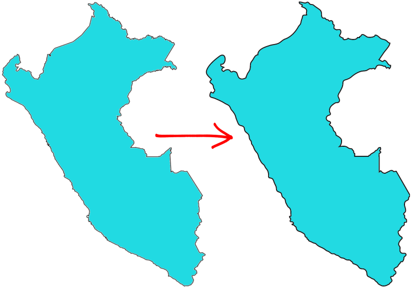
Geometry simplification



≈ 350 Mb $\qquad < 1$ Kb

Today

Geometry simplification

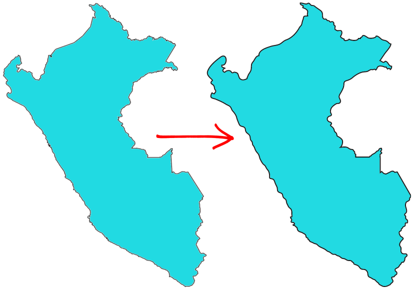


≈ 350 Mb $\quad\quad < 1$ Kb

- **Motivation**: need to ingest very large shapefiles in *GEOMETRY COLUMNS* of limited size, in order to allow efficient approximate operations.

Today

Geometry simplification



≈ 350 Mb < 1 Kb

- **Motivation**: need to ingest very large shapefiles in *GEOMETRY COLUMNS* of limited size, in order to allow efficient approximate operations.
- **Context**: GIS support in Amazon Redshift (first), and geo-spatial support in Snowflake (later).

Today

Motivation (more details):

- ingest very large shapefiles in *GEOMETRY COLUMNS* of limited size, in order to allow efficient approximate operations.

Today

Motivation (more details):

- ingest very large shapefiles in *GEOMETRY COLUMNS* of limited size, in order to allow efficient approximate operations.
- Efficient approximate computation of the area of intersection of polygons.

Today

Motivation (more details):

- ingest very large shapefiles in *GEOMETRY COLUMNS* of limited size, in order to allow efficient approximate operations.
- Efficient approximate computation of the area of intersection of polygons.
- Visualization pipelines.

Today

Motivation (more details):

- ingest very large shapefiles in *GEOMETRY COLUMNS* of limited size, in order to allow efficient approximate operations.
- Efficient approximate computation of the area of intersection of polygons.
- Visualization pipelines.
- Are *similar* two given polygons?

Today

Motivation (more details):

- ingest very large shapefiles in *GEOMETRY COLUMNS* of limited size, in order to allow efficient approximate operations.
- Efficient approximate computation of the area of intersection of polygons.
- Visualization pipelines.
- Are *similar* two given polygons?
- Efficient approximate computation of the center of mass of a polygon.

Today

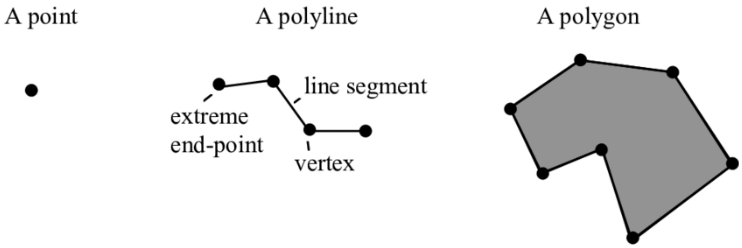
Motivation (more details):

- ingest very large shapefiles in *GEOMETRY COLUMNS* of limited size, in order to allow efficient approximate operations.
- Efficient approximate computation of the area of intersection of polygons.
- Visualization pipelines.
- Are *similar* two given polygons?
- Efficient approximate computation of the center of mass of a polygon.
- Efficient approximate computation of the perimeter of a very large polygon (hundreds of thousands of millions of vertices).

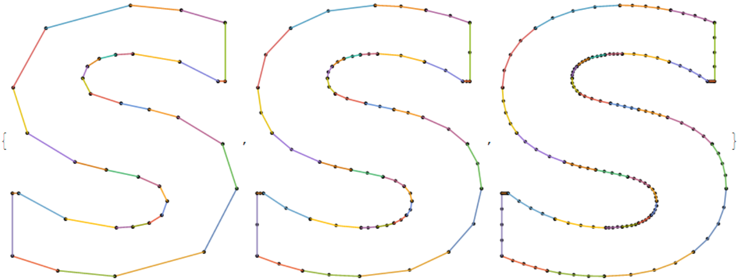
Geometry simplification

- **Part I:** Douglas-Peucker.
- **Part II:** Variant of Douglas-Peucker currently implemented by Amazon Redshit, globally available. Designed by me, implemented jointly by me and my team. Protected and patented by AWS in the Office of Patents of USA.

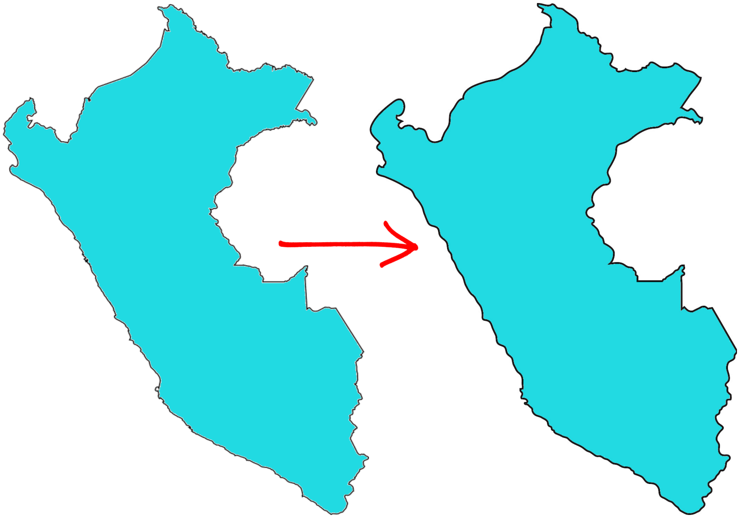
Context and vocabulary.



Douglas-Peucker: problem description.



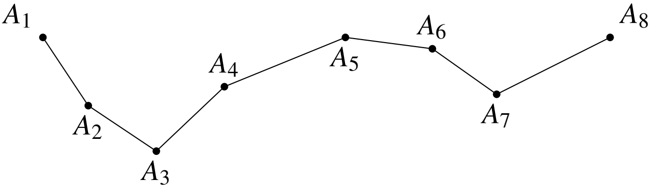
Douglas-Peucker: problem description.



Douglas-Peucker.

Let $\overline{A_1 A_2 \ldots A_n}$ be a *poly-líne* of n $(n \geq 2)$ vertices. Let ε be the *tolerance*.

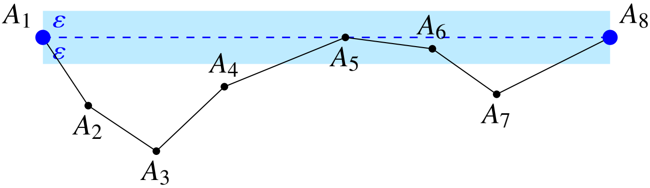
Let A_i $(1 < i < n)$ be such that $D_i = \text{dist}(A_i, \overline{A_1 A_n})$ is maximum. If $D_i > \varepsilon$, repeat the process with the polylines $\overline{A_1 \ldots A_i}$ and $\overline{A_i \ldots A_n}$. Otherwise, simplify the path by removing the vertices $A_2 \ldots A_{n-1}$.



Douglas-Peucker.

Let $\overline{A_1A_2\ldots A_n}$ be a *poly-líne* of n $(n \geq 2)$ vertices. Let ε be the *tolerance*.

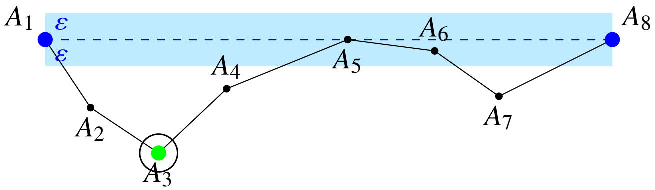
Let A_i $(1 < i < n)$ be such that $D_i = \text{dist}(A_i, \overline{A_1A_n})$ is maximum. If $D_i > \varepsilon$, repeat the process with the polylines $\overline{A_1\ldots A_i}$ and $\overline{A_i\ldots A_n}$. Otherwise, simplify the path by removing the vertices $A_2\ldots A_{n-1}$.



Douglas-Peucker.

Let $\overline{A_1A_2\ldots A_n}$ be a *poly-líne* of n $(n \geq 2)$ vertices. Let ε be the *tolerance*.

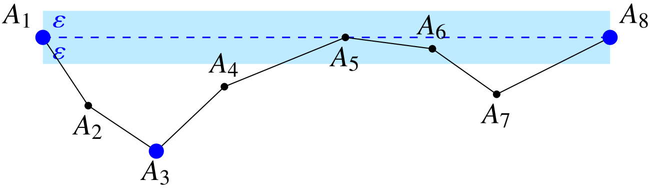
Let A_i $(1 < i < n)$ be such that $D_i = \text{dist}(A_i, \overline{A_1A_n})$ is maximum. If $D_i > \varepsilon$, repeat the process with the polylines $\overline{A_1\ldots A_i}$ and $\overline{A_i\ldots A_n}$. Otherwise, simplify the path by removing the vertices $A_2\ldots A_{n-1}$.



Douglas-Peucker.

Let $\overline{A_1A_2\ldots A_n}$ be a *poly-líne* of n $(n \geq 2)$ vertices. Let ε be the *tolerance*.

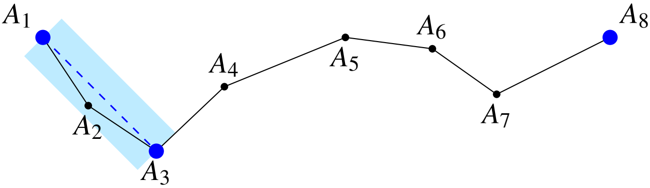
Let A_i $(1 < i < n)$ be such that $D_i = \text{dist}(A_i, \overline{A_1A_n})$ is maximum. If $D_i > \varepsilon$, repeat the process with the polylines $\overline{A_1\ldots A_i}$ and $\overline{A_i\ldots A_n}$. Otherwise, simplify the path by removing the vertices $A_2\ldots A_{n-1}$.



Douglas-Peucker.

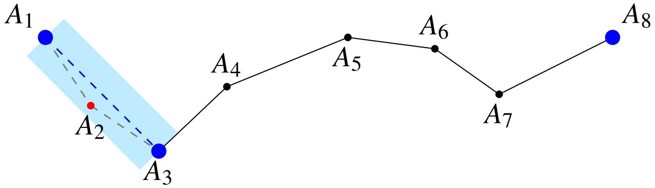
Let $\overline{A_1 A_2 \ldots A_n}$ be a *poly-líne* of n $(n \geq 2)$ vertices. Let ε be the *tolerance*.

Let A_i $(1 < i < n)$ be such that $D_i = \text{dist}(A_i, \overline{A_1 A_n})$ is maximum. If $D_i > \varepsilon$, repeat the process with the polylines $\overline{A_1 \ldots A_i}$ and $\overline{A_i \ldots A_n}$. Otherwise, simplify the path by removing the vertices $A_2 \ldots A_{n-1}$.



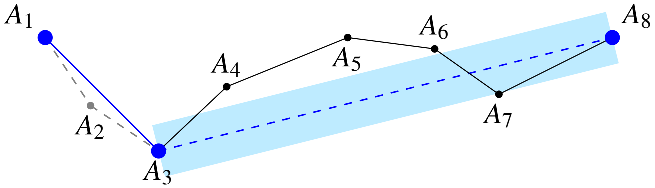
Douglas-Peucker.

Let $\overline{A_1 A_2 \ldots A_n}$ be a *poly-líne* of n $(n \geq 2)$ vertices. Let ε be the *tolerance*.
Let A_i $(1 < i < n)$ be such that $D_i = \text{dist}(A_i, \overline{A_1 A_n})$ is maximum. If $D_i > \varepsilon$, repeat the process with the polylines $\overline{A_1 \ldots A_i}$ and $\overline{A_i \ldots A_n}$. Otherwise, simplify the path by removing the vertices $A_2 \ldots A_{n-1}$.



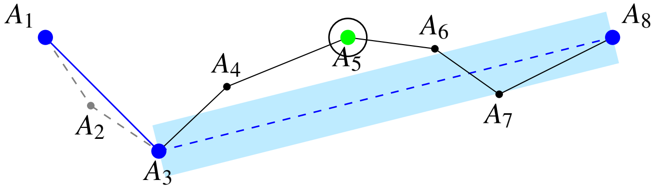
Douglas-Peucker.

Let $\overline{A_1 A_2 \ldots A_n}$ be a *poly-líne* of n $(n \geq 2)$ vertices. Let ε be the *tolerance*.
Let A_i $(1 < i < n)$ be such that $D_i = \text{dist}(A_i, \overline{A_1 A_n})$ is maximum. If $D_i > \varepsilon$, repeat the process with the polylines $\overline{A_1 \ldots A_i}$ and $\overline{A_i \ldots A_n}$.
Otherwise, simplify the path by removing the vertices $A_2 \ldots A_{n-1}$.



Douglas-Peucker.

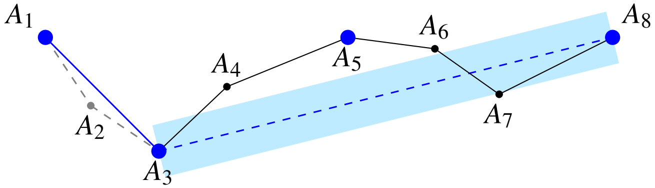
Let $\overline{A_1A_2\ldots A_n}$ be a *poly-líne* of n $(n \geq 2)$ vertices. Let ε be the *tolerance*.
Let A_i $(1 < i < n)$ be such that $D_i = \text{dist}(A_i, \overline{A_1A_n})$ is maximum. If $D_i > \varepsilon$, repeat the process with the polylines $\overline{A_1\ldots A_i}$ and $\overline{A_i\ldots A_n}$. Otherwise, simplify the path by removing the vertices $A_2\ldots A_{n-1}$.



Douglas-Peucker.

Let $\overline{A_1 A_2 \ldots A_n}$ be a *poly-líne* of n $(n \geq 2)$ vertices. Let ε be the *tolerance*.

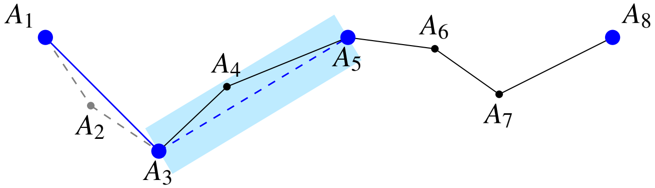
Let A_i $(1 < i < n)$ be such that $D_i = \text{dist}(A_i, \overline{A_1 A_n})$ is maximum. If $D_i > \varepsilon$, repeat the process with the polylines $\overline{A_1 \ldots A_i}$ and $\overline{A_i \ldots A_n}$. Otherwise, simplify the path by removing the vertices $A_2 \ldots A_{n-1}$.



Douglas-Peucker.

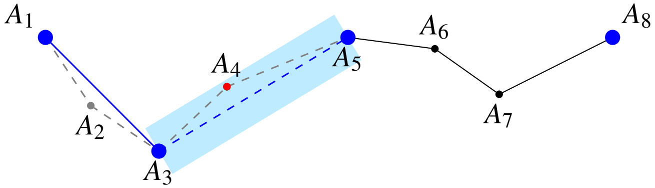
Let $\overline{A_1 A_2 \ldots A_n}$ be a *poly-líne* of n $(n \geq 2)$ vertices. Let ε be the *tolerance*.

Let A_i $(1 < i < n)$ be such that $D_i = \text{dist}(A_i, \overline{A_1 A_n})$ is maximum. If $D_i > \varepsilon$, repeat the process with the polylines $\overline{A_1 \ldots A_i}$ and $\overline{A_i \ldots A_n}$. Otherwise, simplify the path by removing the vertices $A_2 \ldots A_{n-1}$.



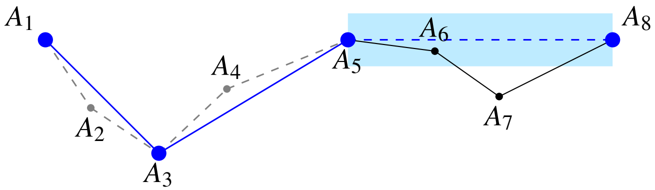
Douglas-Peucker.

Let $\overline{A_1A_2\ldots A_n}$ be a *poly-líne* of n $(n \geq 2)$ vertices. Let ε be the *tolerance*.
Let A_i $(1 < i < n)$ be such that $D_i = \text{dist}(A_i, \overline{A_1A_n})$ is maximum. If $D_i > \varepsilon$, repeat the process with the polylines $\overline{A_1\ldots A_i}$ and $\overline{A_i\ldots A_n}$. Otherwise, simplify the path by removing the vertices $A_2\ldots A_{n-1}$.



Douglas-Peucker.

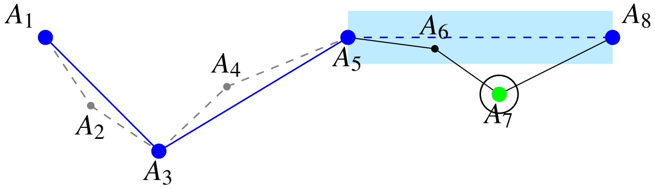
Let $\overline{A_1 A_2 \ldots A_n}$ be a *poly-líne* of n $(n \geq 2)$ vertices. Let ε be the *tolerance*.
Let A_i $(1 < i < n)$ be such that $D_i = \text{dist}(A_i, \overline{A_1 A_n})$ is maximum. If $D_i > \varepsilon$, repeat the process with the polylines $\overline{A_1 \ldots A_i}$ and $\overline{A_i \ldots A_n}$.
Otherwise, simplify the path by removing the vertices $A_2 \ldots A_{n-1}$.



Douglas-Peucker.

Let $\overline{A_1A_2\ldots A_n}$ be a *poly-líne* of n $(n \geq 2)$ vertices. Let ε be the *tolerance*.

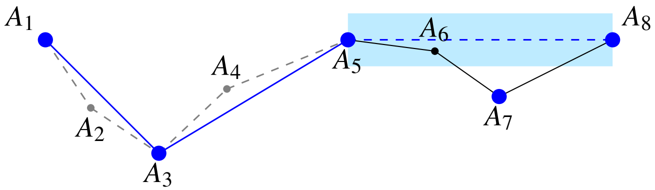
Let A_i $(1 < i < n)$ be such that $D_i = \text{dist}(A_i, \overline{A_1A_n})$ is maximum. If $D_i > \varepsilon$, repeat the process with the polylines $\overline{A_1\ldots A_i}$ and $\overline{A_i\ldots A_n}$. Otherwise, simplify the path by removing the vertices $A_2\ldots A_{n-1}$.



Douglas-Peucker.

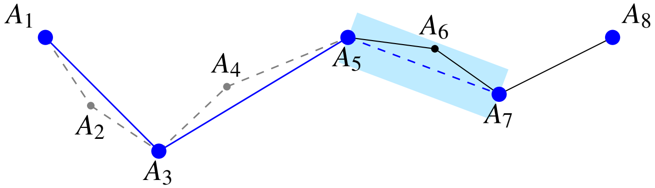
Let $\overline{A_1 A_2 \ldots A_n}$ be a *poly-líne* of n $(n \geq 2)$ vertices. Let ε be the *tolerance*.

Let A_i $(1 < i < n)$ be such that $D_i = \text{dist}(A_i, \overline{A_1 A_n})$ is maximum. If $D_i > \varepsilon$, repeat the process with the polylines $\overline{A_1 \ldots A_i}$ and $\overline{A_i \ldots A_n}$. Otherwise, simplify the path by removing the vertices $A_2 \ldots A_{n-1}$.



Douglas-Peucker.

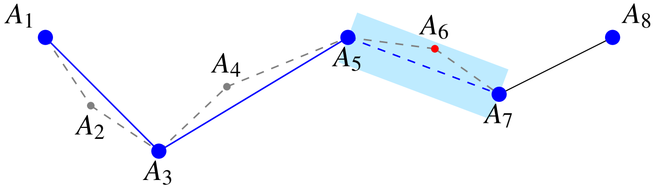
Let $\overline{A_1A_2\ldots A_n}$ be a *poly-líne* of n $(n \geq 2)$ vertices. Let ε be the *tolerance*.
Let A_i $(1 < i < n)$ be such that $D_i = \text{dist}(A_i, \overline{A_1A_n})$ is maximum. If $D_i > \varepsilon$, repeat the process with the polylines $\overline{A_1\ldots A_i}$ and $\overline{A_i\ldots A_n}$. Otherwise, simplify the path by removing the vertices $A_2\ldots A_{n-1}$.



Douglas-Peucker.

Let $\overline{A_1 A_2 \ldots A_n}$ be a *poly-líne* of n $(n \geq 2)$ vertices. Let ε be the *tolerance*.

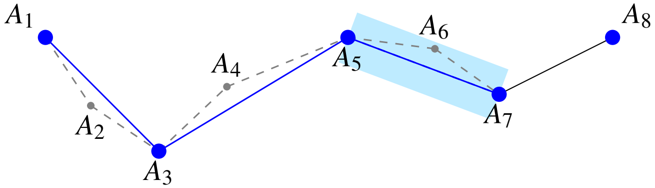
Let A_i $(1 < i < n)$ be such that $D_i = \text{dist}(A_i, \overline{A_1 A_n})$ is maximum. If $D_i > \varepsilon$, repeat the process with the polylines $\overline{A_1 \ldots A_i}$ and $\overline{A_i \ldots A_n}$. Otherwise, simplify the path by removing the vertices $A_2 \ldots A_{n-1}$.



Douglas-Peucker.

Let $\overline{A_1 A_2 \ldots A_n}$ be a *poly-líne* of n $(n \geq 2)$ vertices. Let ε be the *tolerance*.

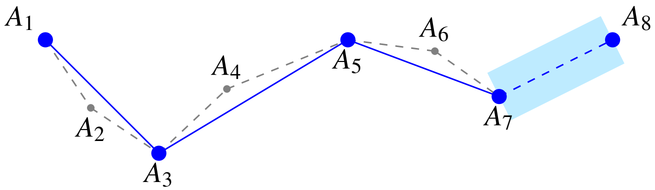
Let A_i $(1 < i < n)$ be such that $D_i = \text{dist}(A_i, \overline{A_1 A_n})$ is maximum. If $D_i > \varepsilon$, repeat the process with the polylines $\overline{A_1 \ldots A_i}$ and $\overline{A_i \ldots A_n}$. Otherwise, simplify the path by removing the vertices $A_2 \ldots A_{n-1}$.



Douglas-Peucker.

Let $\overline{A_1 A_2 \ldots A_n}$ be a *poly-líne* of n $(n \geq 2)$ vertices. Let ε be the *tolerance*.

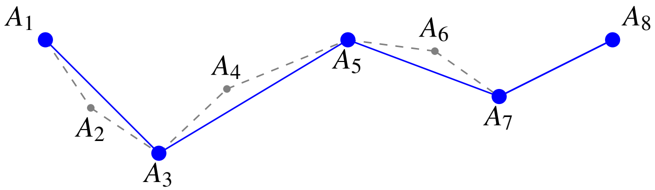
Let A_i $(1 < i < n)$ be such that $D_i = \text{dist}(A_i, \overline{A_1 A_n})$ is maximum. If $D_i > \varepsilon$, repeat the process with the polylines $\overline{A_1 \ldots A_i}$ and $\overline{A_i \ldots A_n}$. Otherwise, simplify the path by removing the vertices $A_2 \ldots A_{n-1}$.



Douglas-Peucker.

Let $\overline{A_1 A_2 \ldots A_n}$ be a *poly-líne* of n $(n \geq 2)$ vertices. Let ε be the *tolerance*.

Let A_i $(1 < i < n)$ be such that $D_i = \text{dist}(A_i, \overline{A_1 A_n})$ is maximum. If $D_i > \varepsilon$, repeat the process with the polylines $\overline{A_1 \ldots A_i}$ and $\overline{A_i \ldots A_n}$. Otherwise, simplify the path by removing the vertices $A_2 \ldots A_{n-1}$.



Douglas-Peucker.

- The size of the output if function of the tolerance. The larger is the tolerance, the smaller is the size of the output.
- If we know the limit for the size of the output, how can we choose the value for the tolerance that exploits this limit in the best posible way?

Douglas-Peucker.

- The size of the output if function of the tolerance. The larger is the tolerance, the smaller is the size of the output.
- If we know the limit for the size of the output, how can we choose the value for the tolerance that exploits this limit in the best posible way?
- First idea: binary search on the tolerance.

Douglas-Peucker.

Two considerations.

- The users are not people coming from computational geometry. The parameter *tolerance* is not intuitive. The users have a map of several gigabytes (maybe even terabytes) and need to represent it in the best possible way in a much smaller size, which is known.
- The binary-search-based process runs Douglas-Peucker several times.

An impreved approach.

The approach that I will propose:

- computes the same simplification as Douglas-Peucker *with the tolerance that maximizes the usage of the available space, without exceeding it.* This is: it computes the *best possible simplification in terms of exploitation of the available space*.
- Executes *just one iteration of Douglas-Peucker*, differently to the approach mentioned earlier, in which several iterations were needed.

An impreved approach.

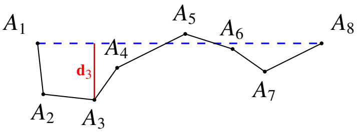
Idea

- When does the recursion in Douglas-Peucker *end*?
- We start with infinite tolerance, and we reduce it iteratively until the *next maximum tolerance that would change the stop condition of some of the steps in which the decision was to stop*.

An impreved approach.

Each node has 4 values: FROM, TO, MAXDIST, MAXDISTARG.
If MaxDist $> \varepsilon$ we keep subdividing. If $\leq \varepsilon$, we stop.

When Douglas-Peucker *ends*, we set ε to be the largest MaxDist among the nodes in which the decision was *to stop*, and we continue as if that one was the initial value of the tolerance.

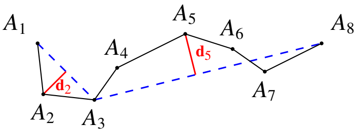


1. $\varepsilon = \infty$. Simplification: A_1A_8.
   Distancias frenantes: $\{d_3\}$.

An impreved approach.

Each node has 4 values: FROM, TO, MAXDIST, MAXDISTARG.
If MaxDist $> \varepsilon$ we keep subdividing. If $\leq \varepsilon$, we stop.

When Douglas-Peucker *ends*, we set ε to be the largest MaxDist among the nodes in which the decision was *to stop*, and we continue as if that one was the initial value of the tolerance.

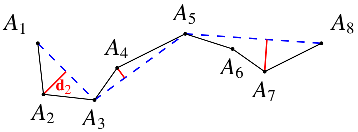


1. $\varepsilon = \infty$. Simplification: A_1A_8. Distancias frenantes: $\{d_3\}$.
2. $\varepsilon = d_3$. Simplification: $A_1A_3A_8$. Distancias frenantes: $\{d_2, d_5\}$(Nota: $d_5 > d_2$).

An impreved approach.

Each node has 4 values: FROM, TO, MAXDIST, MAXDISTARG.
If MaxDist $> \varepsilon$ we keep subdividing. If $\leq \varepsilon$, we stop.

When Douglas-Peucker *ends*, we set ε to be the largest MaxDist among the nodes in which the decision was *to stop*, and we continue as if that one was the initial value of the tolerance.

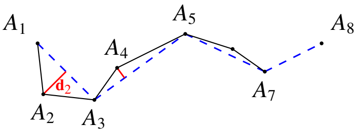


1. $\varepsilon = \infty$. Simplification: A_1A_8.
   Distancias frenantes: $\{d_3\}$.
2. $\varepsilon = d_3$. Simplification: $A_1A_3A_8$.
   Distancias frenantes:
   $\{d_2, d_5\}$(Nota: $d_5 > d_2$).

An impreved approach.

Each node has 4 values: FROM, TO, MAXDIST, MAXDISTARG.
If MaxDist $> \varepsilon$ we keep subdividing. If $\leq \varepsilon$, we stop.

When Douglas-Peucker *ends*, we set ε to be the largest MaxDist among the nodes in which the decision was *to stop*, and we continue as if that one was the initial value of the tolerance.



1. $\varepsilon = \infty$. Simplification: A_1A_8. Distancias frenantes: $\{d_3\}$.
2. $\varepsilon = d_3$. Simplification: $A_1A_3A_8$. Distancias frenantes: $\{d_2, d_5\}$(Nota: $d_5 > d_2$).
3. $\varepsilon = d_5$. Simplification: $A_1A_3A_5A_8$. Distancias f

Aditional considerations.

- Stop condition:
    - the target number of points was reached
    - it is not possible to keep simplifying
- The adaption of this method to polygons, multi-polygons and geometry-collections requires simply the use of a *forest* structure instead of a tree, and apply the same ideas.

Today

- Geometry simplification. (patented by Amazon Web Services in the Patents Office of USA).
- ***Spatial index*, Amazon Redshift.**
- An optimization for the KNN clustering algorithm. Amazon Redshift.

The *spatial index* of Amazon Redshift

Problem description, motivation

- Queries like *select all the restaurants within 100 meters of a given street, in a region of a city described by a polygon P in a map* use approached based in NESTED JOINs, resulting runtimes so large that are not acceptable.
- Many examples of queries involving geometric data and selection of entries based on operations on those geometric data.
- Context: Amazon Redshift is one of largest heavily-distributed databases, with all the implications in term of engineering that it means.

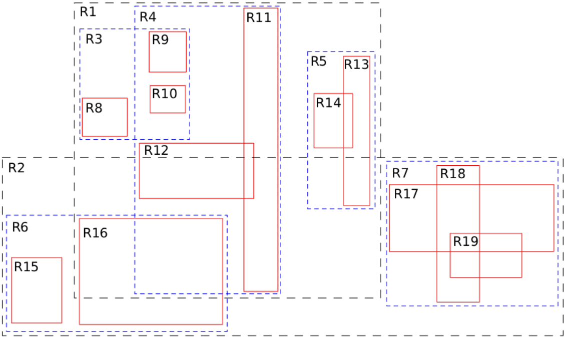
The *spatial index* of Amazon Redshift

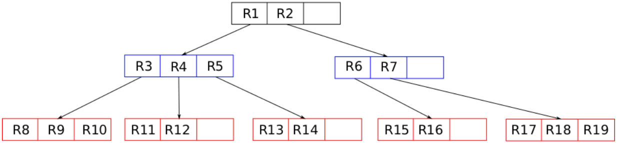
Implemented idea, results

- Index based in R-Trees.
- The transition from public preview to global availability was smooth and without unexpected surprises.
- The runtimes of geometric queries relevant to the index reduced dramatically. Approximatedly 50 times fastrer. The index transformed minutes into seconds. It made *possible* queries that were *not possible*.

The *spatial index* of Amazon Redshift

Generalities about R-Trees





Today

- Geometry simplification. (patented by Amazon Web Services in the Patents Office of USA).
- *Spatial index*, Amazon Redshift.
- **An optimization for the KNN clustering algorithm. Amazon Redshift.**

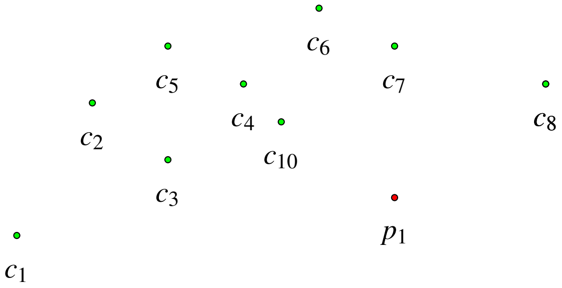
Simplified KNN. Amazon Redshift

Problem description

Given k green points (*centroids*, from now on) and n red points (*points*, from now on), compute one association that allows to know, for each point, which is the closest centroid.

Simplified KNN. Amazon Redshift

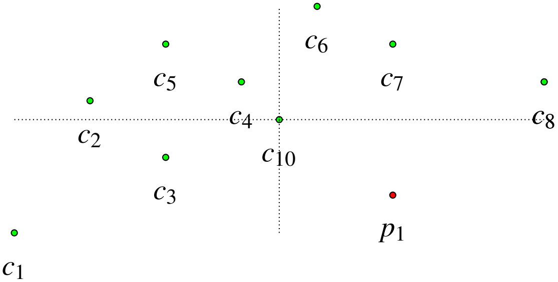
Problem description



- *Trivial approach*: do nk operations and use memory proportional to k.
- Let's assume k centroids and 1 point (this process will be repeated n times).

Simplified KNN. Amazon Redshift

Problem description

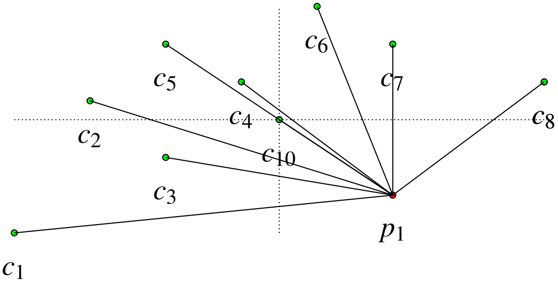


- *Trivial approach*: do nk operations and use memory proportional to k.
- Let's assume k centroids and 1 point (this process will be repeated n times).

Choose (in time proportional to $\log k$) the centroid that is in *as in the middle as possible* (c_{10})

Simplified KNN. Amazon Redshift

Problem description

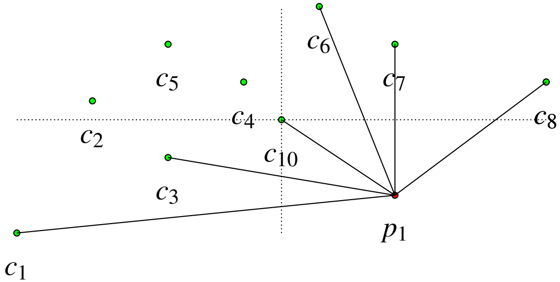


- *Trivial approach*: do nk operations and use memory proportional to k.
- Let's assume k centroids and 1 point (this process will be repeated n times).

Instead of comparing these values...

Simplified KNN. Amazon Redshift

Problem description



- *Trivial approach*: do nk operations and use memory proportional to k.
- Let's assume k centroids and 1 point (this process will be repeated n times).

We compare these ones.

KNN Simplificado. Amazon Redshift

Result

- Significant impact. Improved the performance of the geometric clustering of Amazon Redshift by approx. 20%.

Exercises

- How do we find in linear time the median of a list?
- How do we find in linear time the point that is *as in the middle as possible*, described in the previous slide?
- Which are the best and worse cases for the R-tree described for the spatial index?