Graph Visualization

0710006 盧可瑜

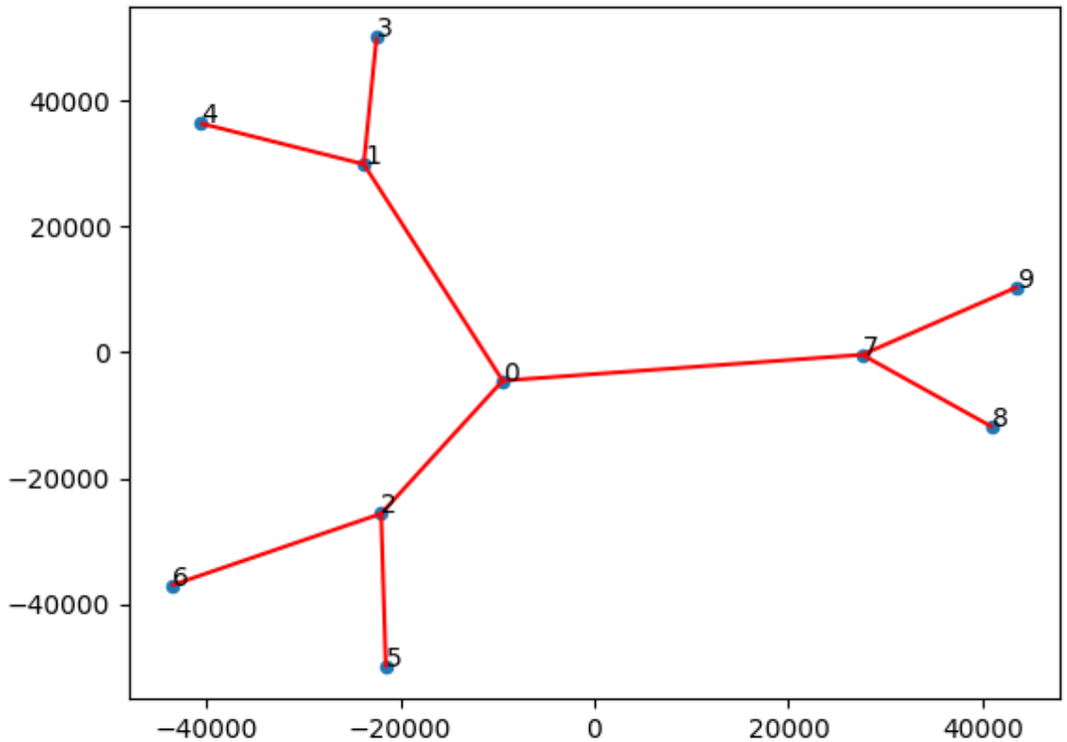
https://github.com/Luke2336/graph-visualization

What is visualization?

- Edges:
  {0, 1} {0, 2} {0, 7}
  {1, 3} {1, 4} {2, 5}
  {2, 6} {7, 8} {7, 9}

What is visualization?

- Edges:
  {0, 1} {0, 2} {0, 7}
  {1, 3} {1, 4} {2, 5}
  {2, 6} {7, 8} {7, 9}

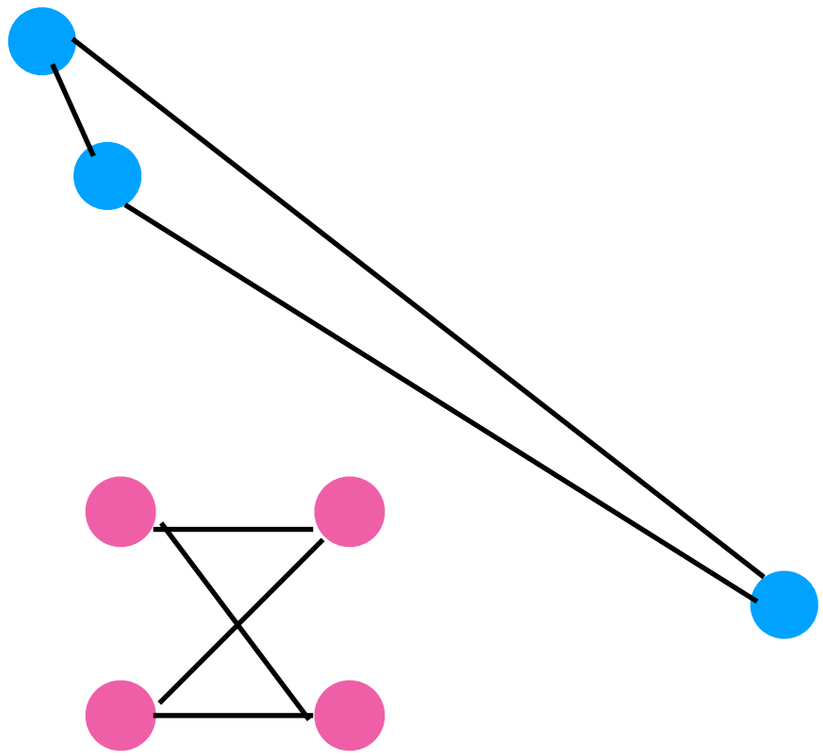


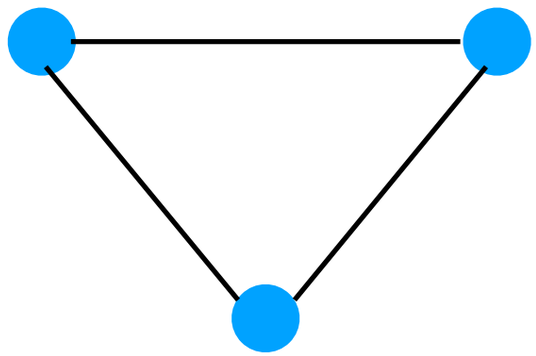
- Why we need to visualize a graph?
  - To find some properties of the graph, such as chromatic number, connectivity, diameter, ……
  - To know the relationship between each node. For example, the friendship on facebook or the inheritance diagram, …

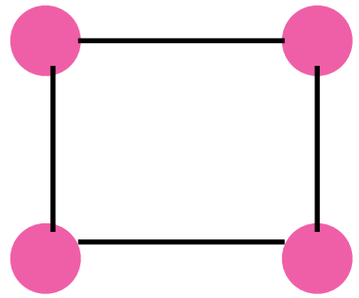
Objective

- To visualize all undirected simple graphs with less than 1000 vertices.

- What is important when we draw a graph?
    - We hope that the number of edge crossings as small as possible.
    - We hope that the nodes are not too crowded or too sparse.

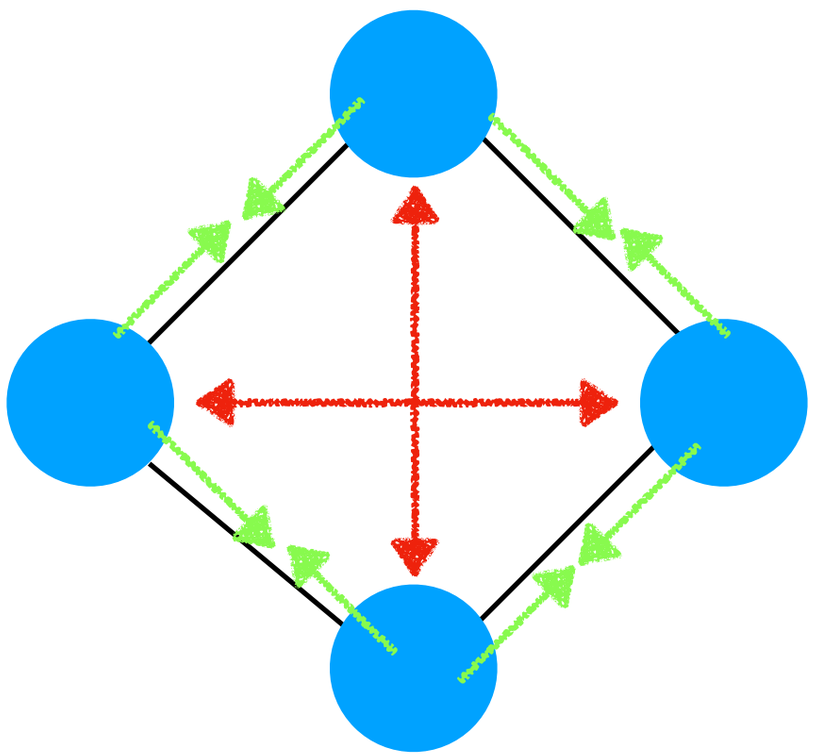






Algorithm

- It is a force-directed method.
- For a pair of vertices, we construct a repulsive force.
- For two incident vertices, we construct an attractive force.



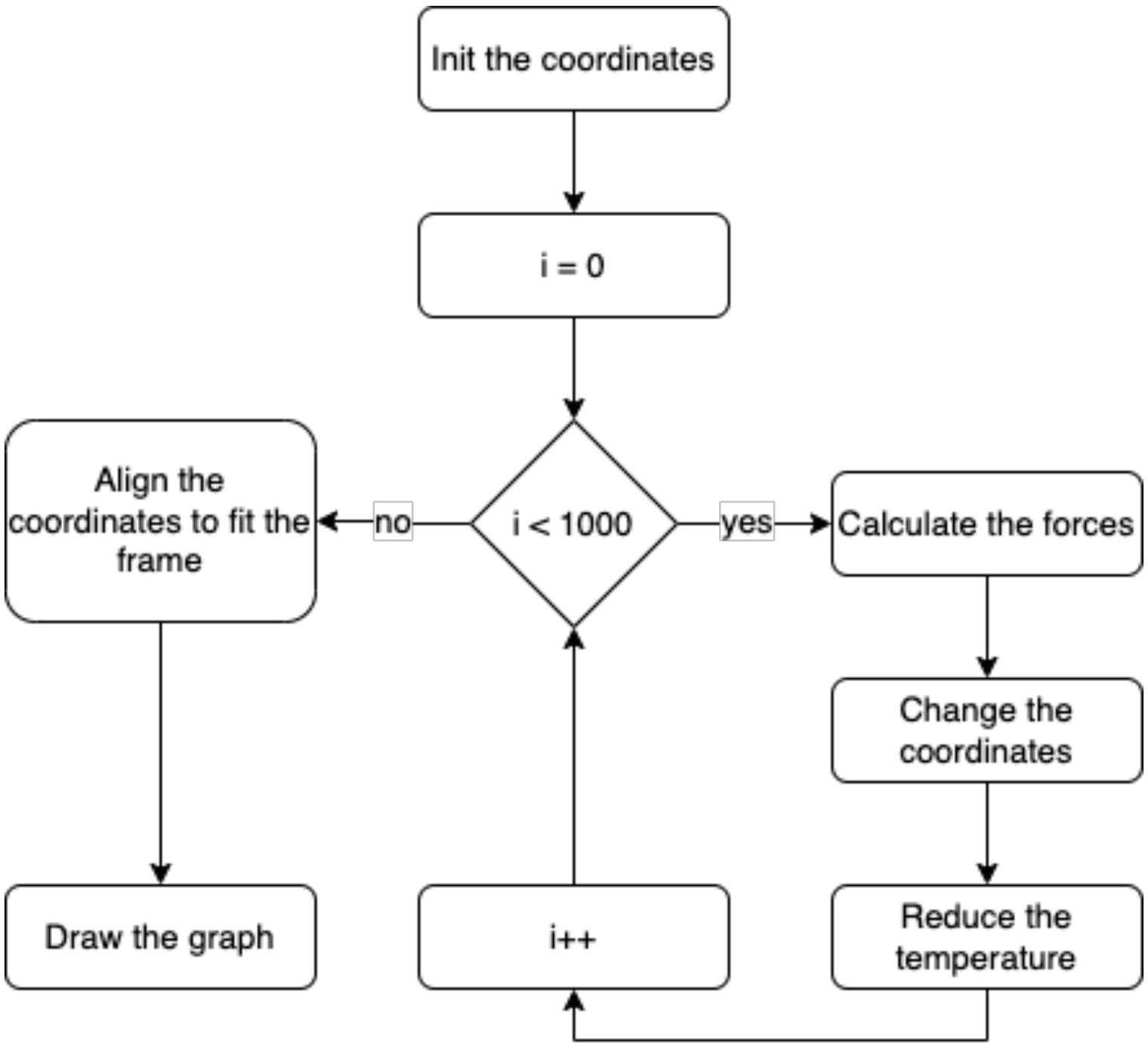
Algorithm

- Repulsive force → Coulomb's law
- Attractive force → Hooke's law

Algorithm

- We iteratively change the positions of vertices.
- Too let it converge, we use simulated annealing.
  - The displacement $\propto$ the temperature
  - Each iteration, we reduce the temperature

Flow Chart

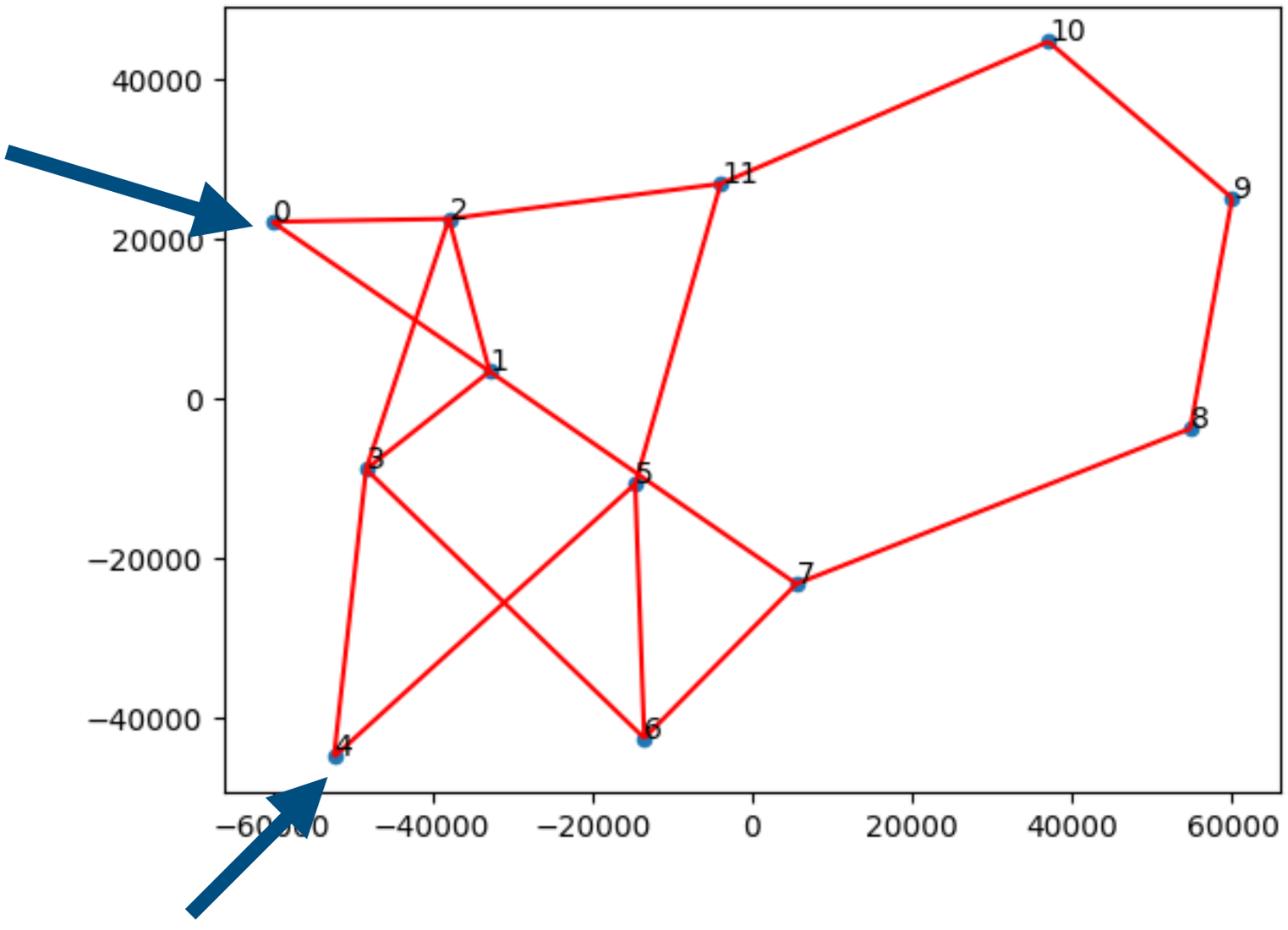

Init the coordinates
i = 0
i < 1000
no
yes
Align the coordinates to fit the frame
Calculate the forces
Change the coordinates
Reduce the temperature
i++
Draw the graph

Challenges

- We should avoid two segments overlapping.
    - To solve this problem, the initial positions should be well-designed. In my code, I scatter the nodes as a circle, which can avoid three nodes collinear.
- It is hard to find some constants to calculate the forces, which depends on the number of nodes.
- Not optimal solution, only local minima.

Challenges

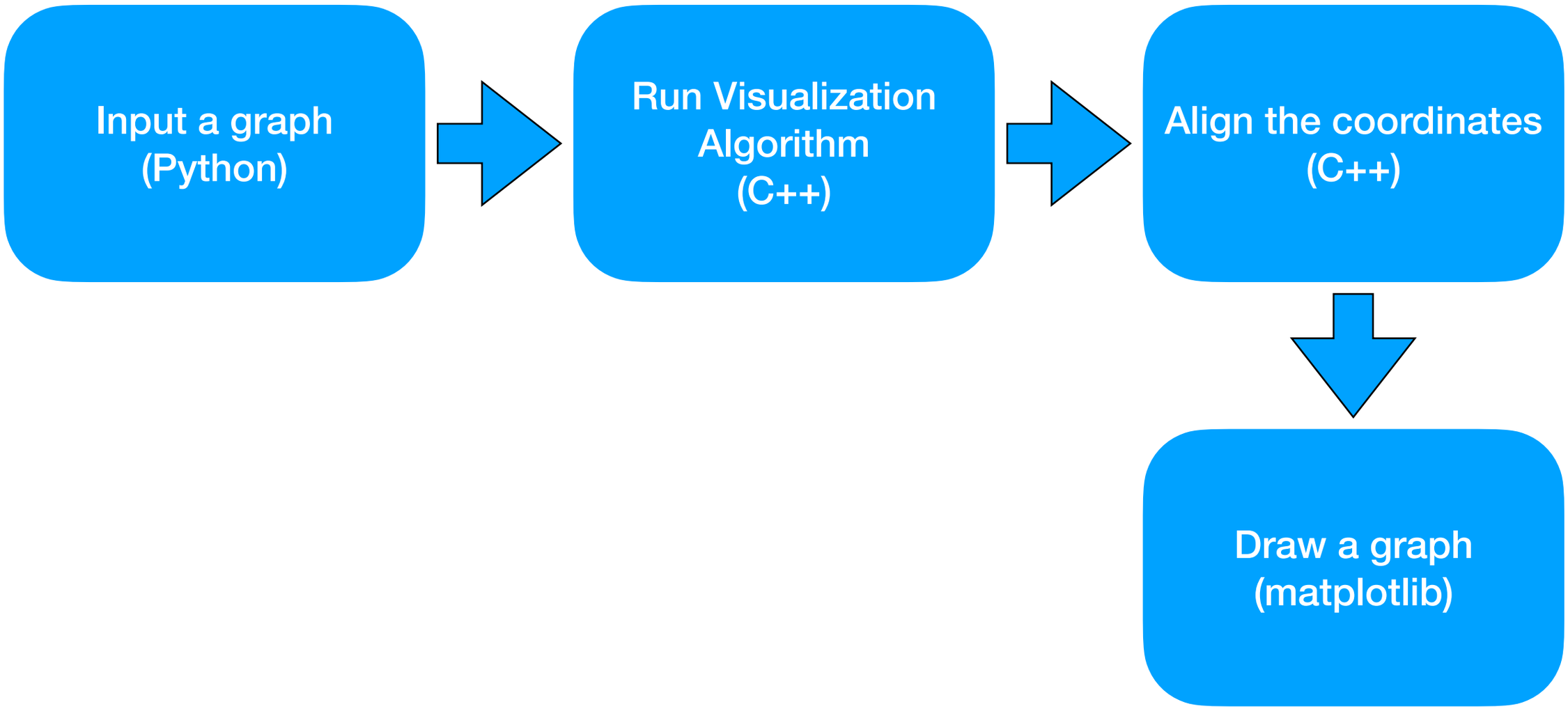
- Edges still cross in some planar graphs.



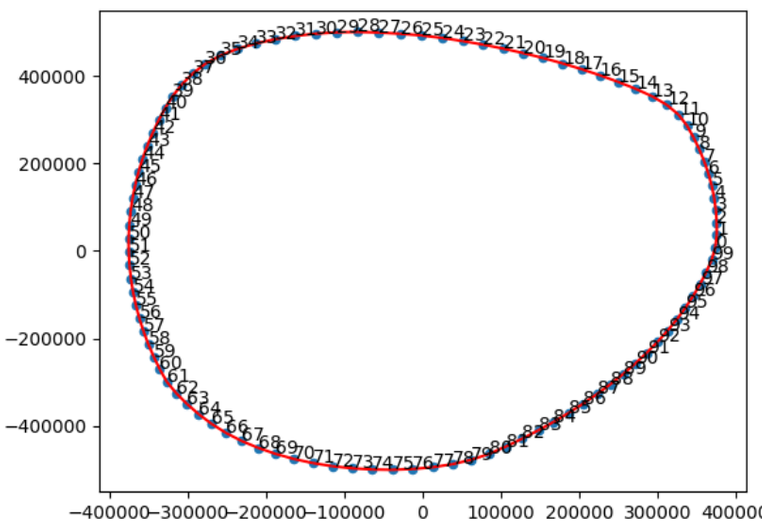
Python API

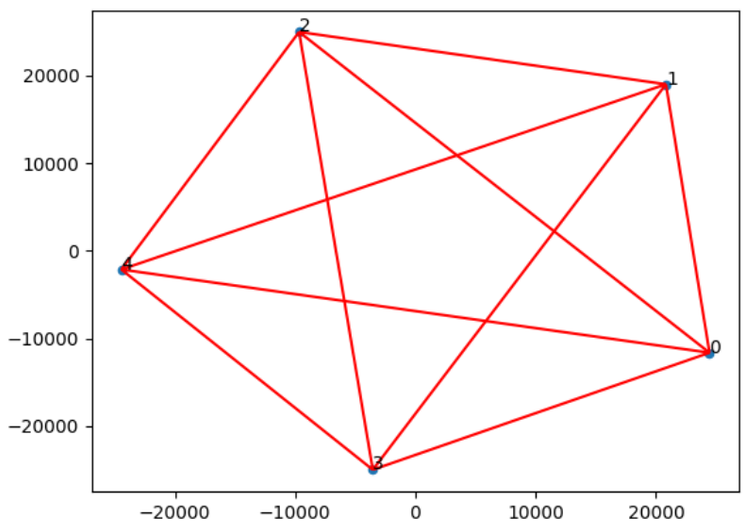
- `Visualization(numNode, numEdge, Edges)`
  - `numNode`: number of nodes
  - `numEdge`: number of edges
  - `Edges`: a list of edges

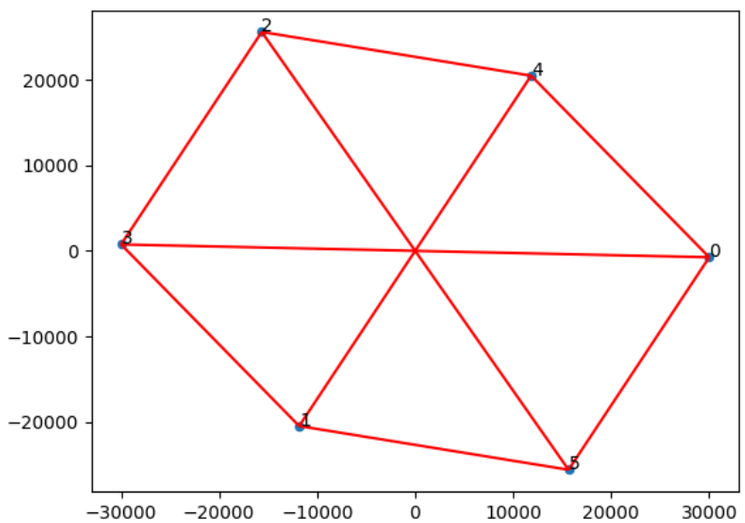
Architecture

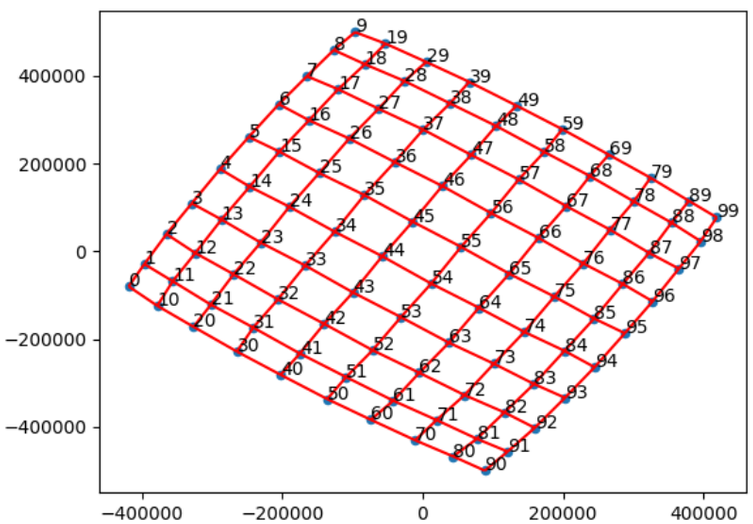


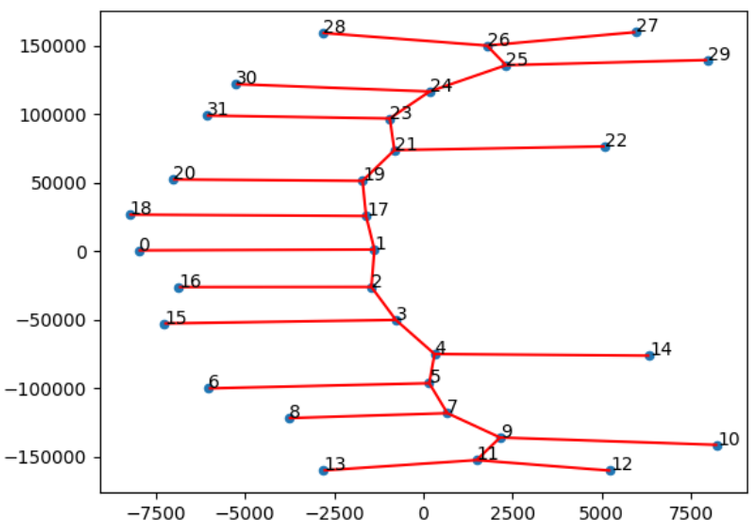
Result

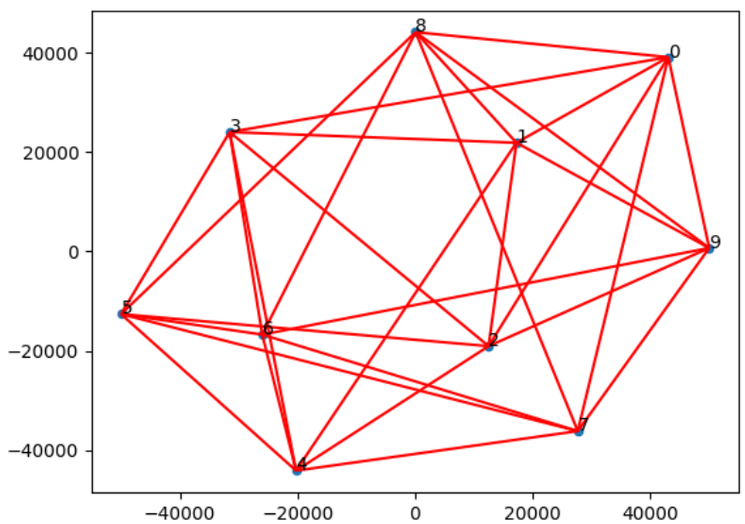












Testing

- To make sure the implementation is correct, I adjust the number of iterations, and found it would be more convergent when there are more iterations.

- It take 4.33s to calculate the coordinates of a complete graph of 1000 nodes.

Reference

- If you feel interested in this algorithm, you can search “Fruchterman-Reingold”.

- https://dcc.fceia.unr.edu.ar/sites/default/files/uploads/materias/fruchterman.pdf

Q & A