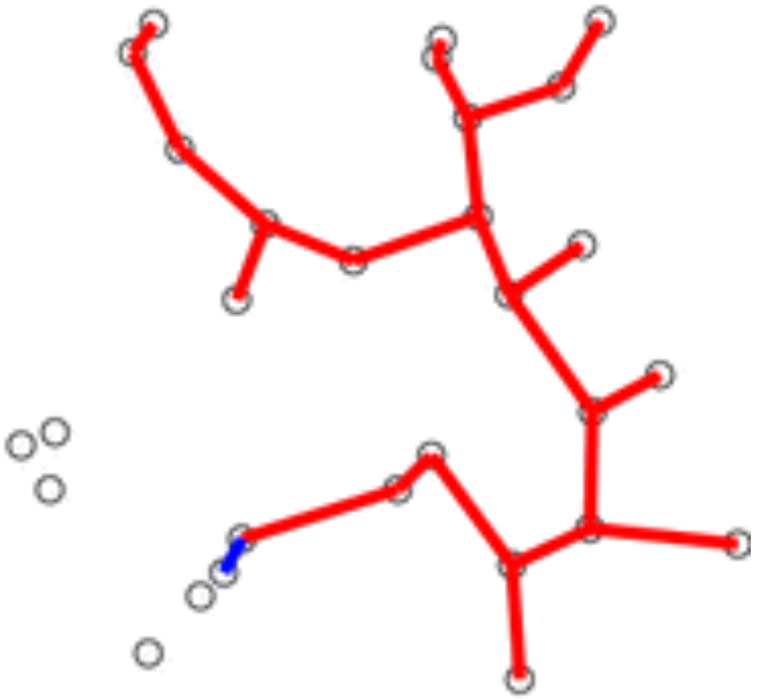
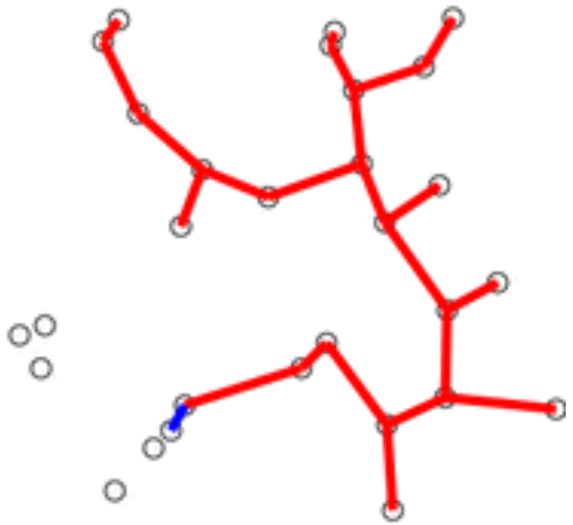
Class 26: Graph algorithms and visualization

Instructor: Michael Szell
Nov 29, 2019

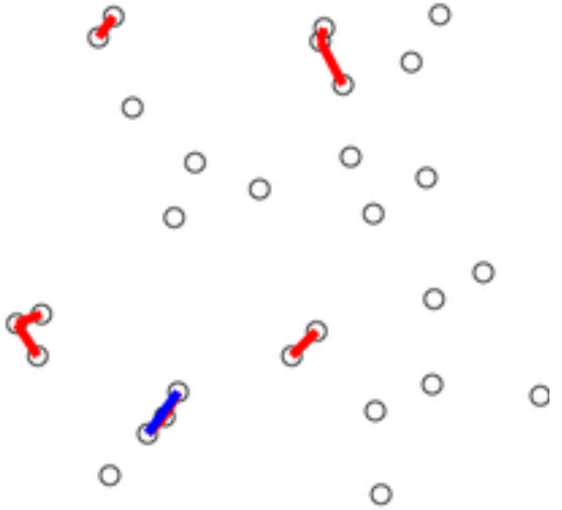


Today you will learn about minimum spanning tree algorithms

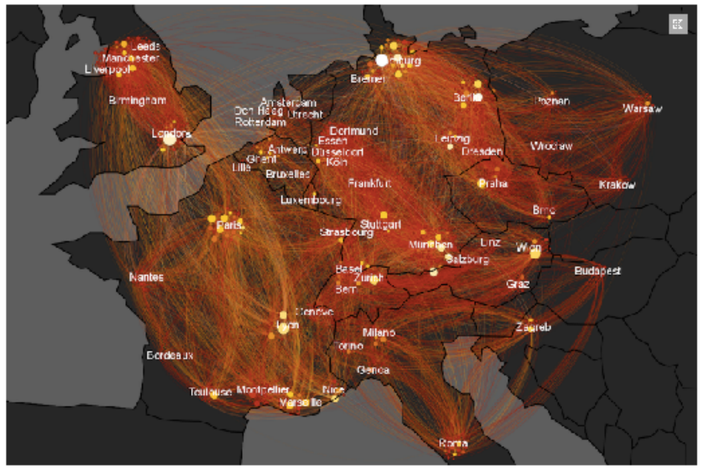
Prim



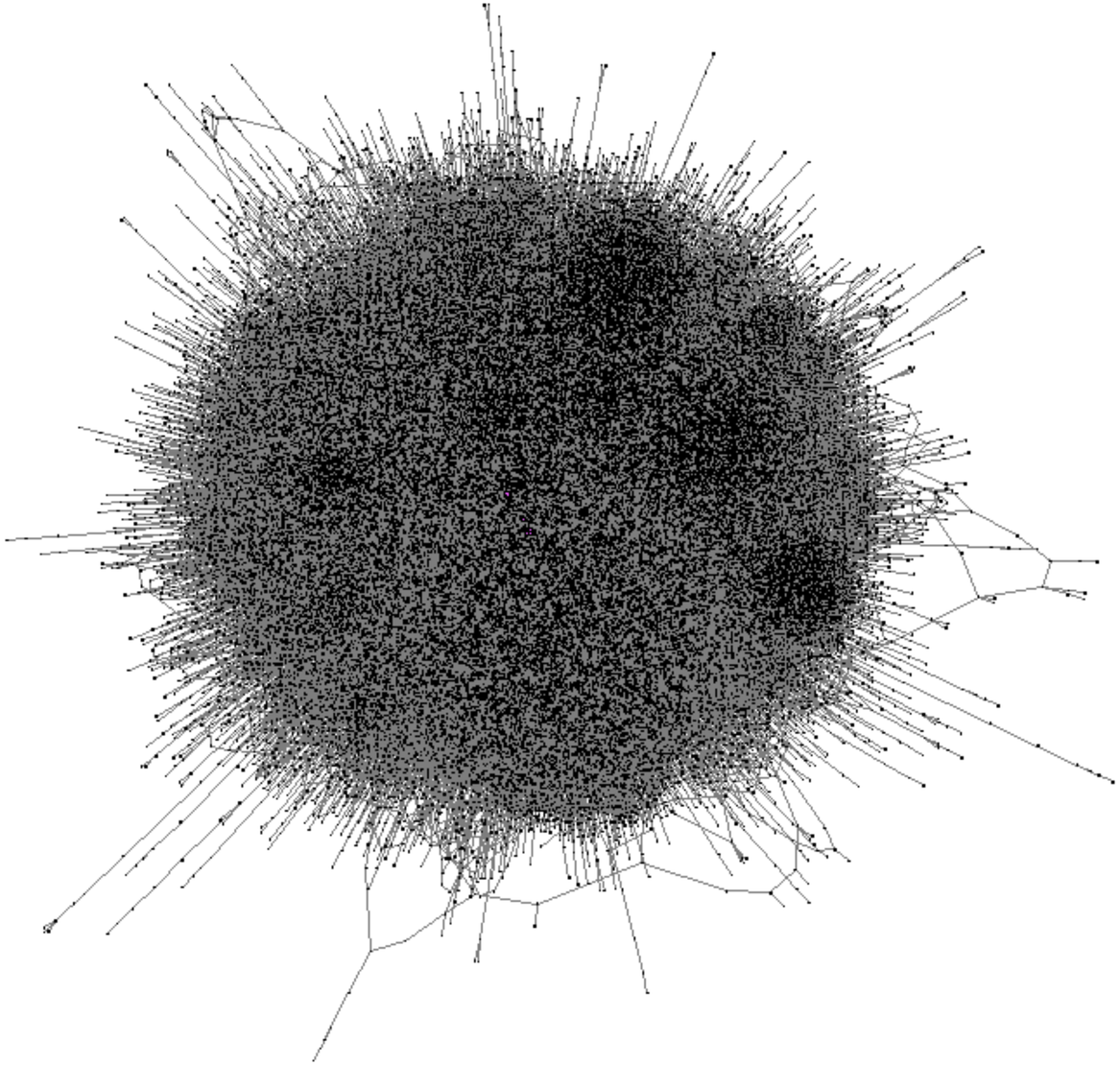
Kruskal



Mapping with Gephi

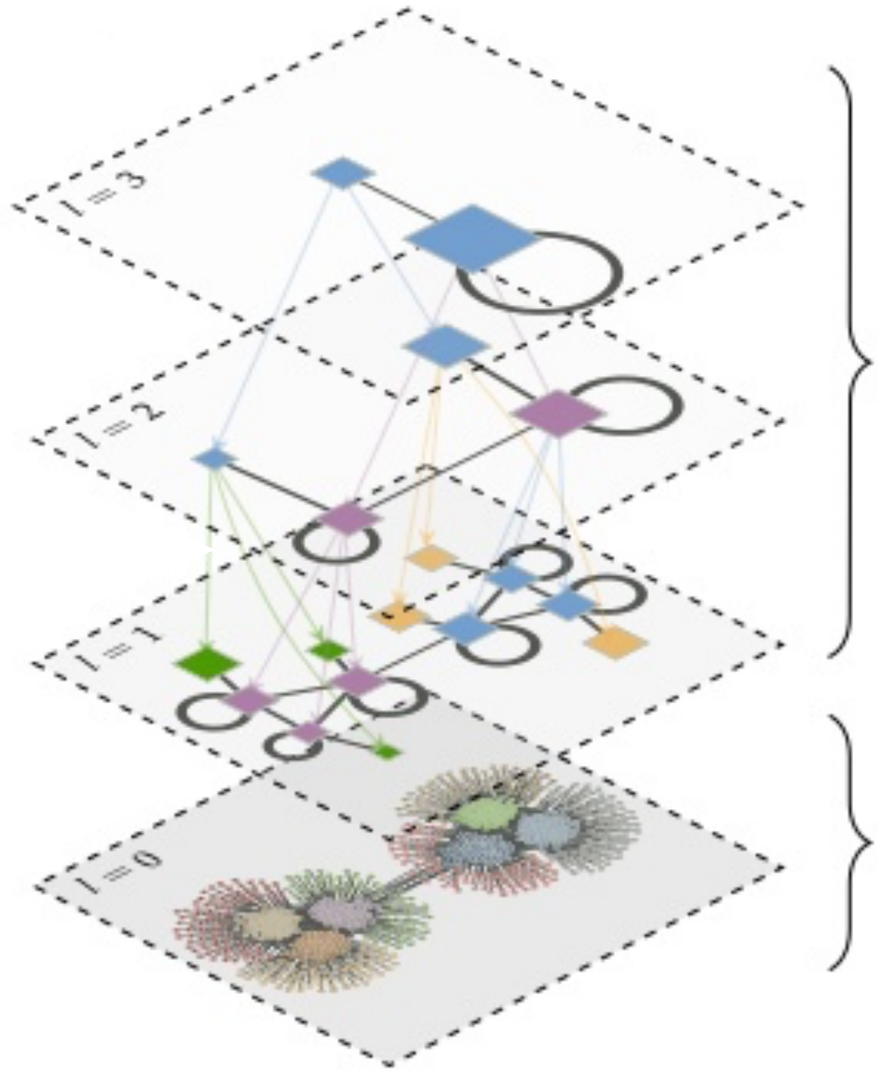


Networks can have a lot of links, even sparse ones



Je suis une hairball

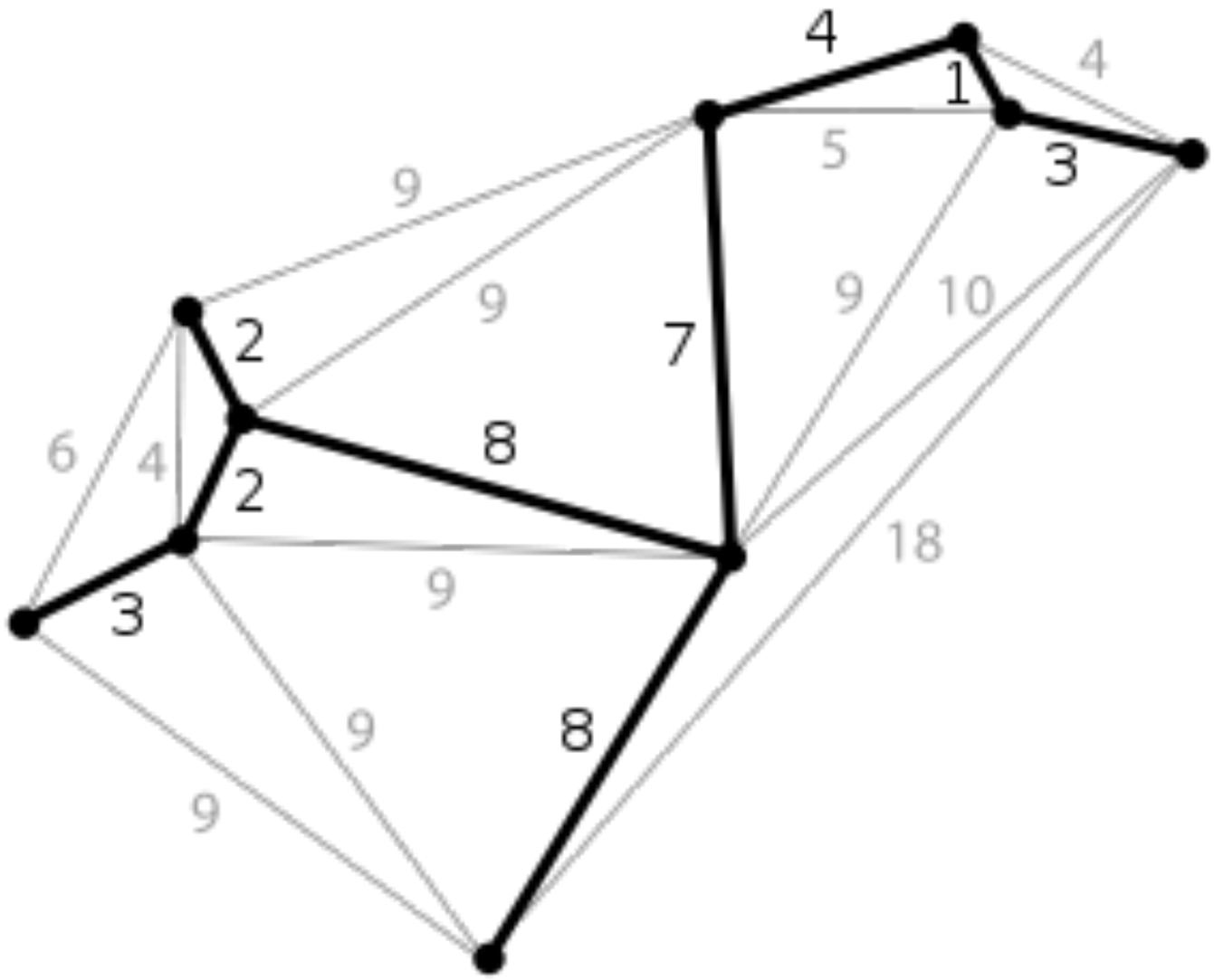
Network filtering constructs a new network with far fewer data, allowing exploring the relevant network features



A simpler network, preserving some relevant features

Original network

The minimum spanning tree identifies the network's "backbone"



The minimum spanning tree (MST) is a special spanning tree

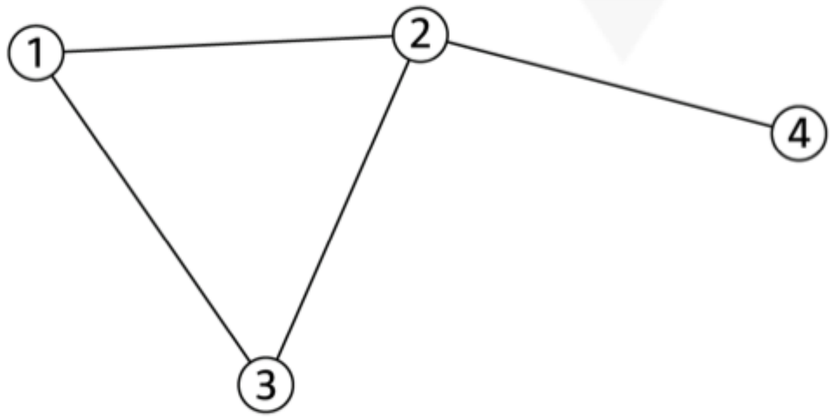
A tree is a connected acyclic graph (= no cycles).

The minimum spanning tree (MST) is a special spanning tree

A tree is a connected acyclic graph (= no cycles).

Cycle: A walk of length 3 or more, starting and ending at the same node, all other nodes being distinct.

$\{n_1, n_2, n_3, n_1\}$



The minimum spanning tree (MST) is a special spanning tree

A tree is a connected acyclic graph (= no cycles).

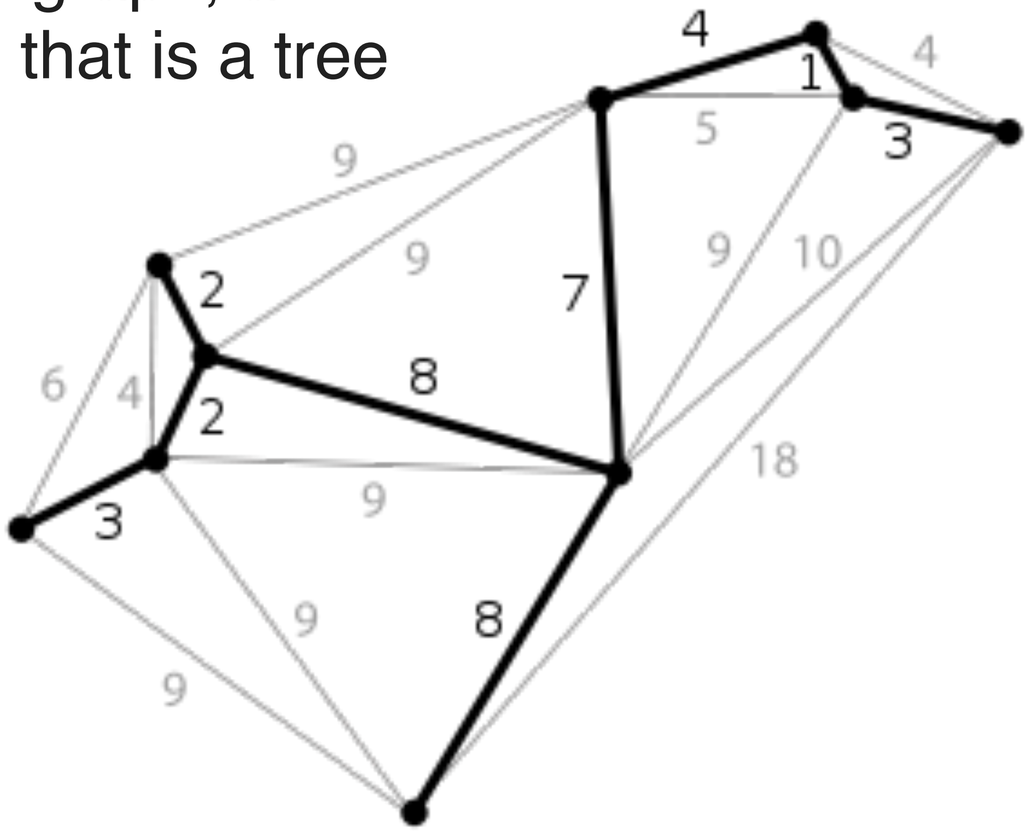
Given a connected, undirected, weighted graph, a spanning tree of the graph is a subgraph that is a tree and connects all the nodes.

The minimum spanning tree (MST) is a special spanning tree

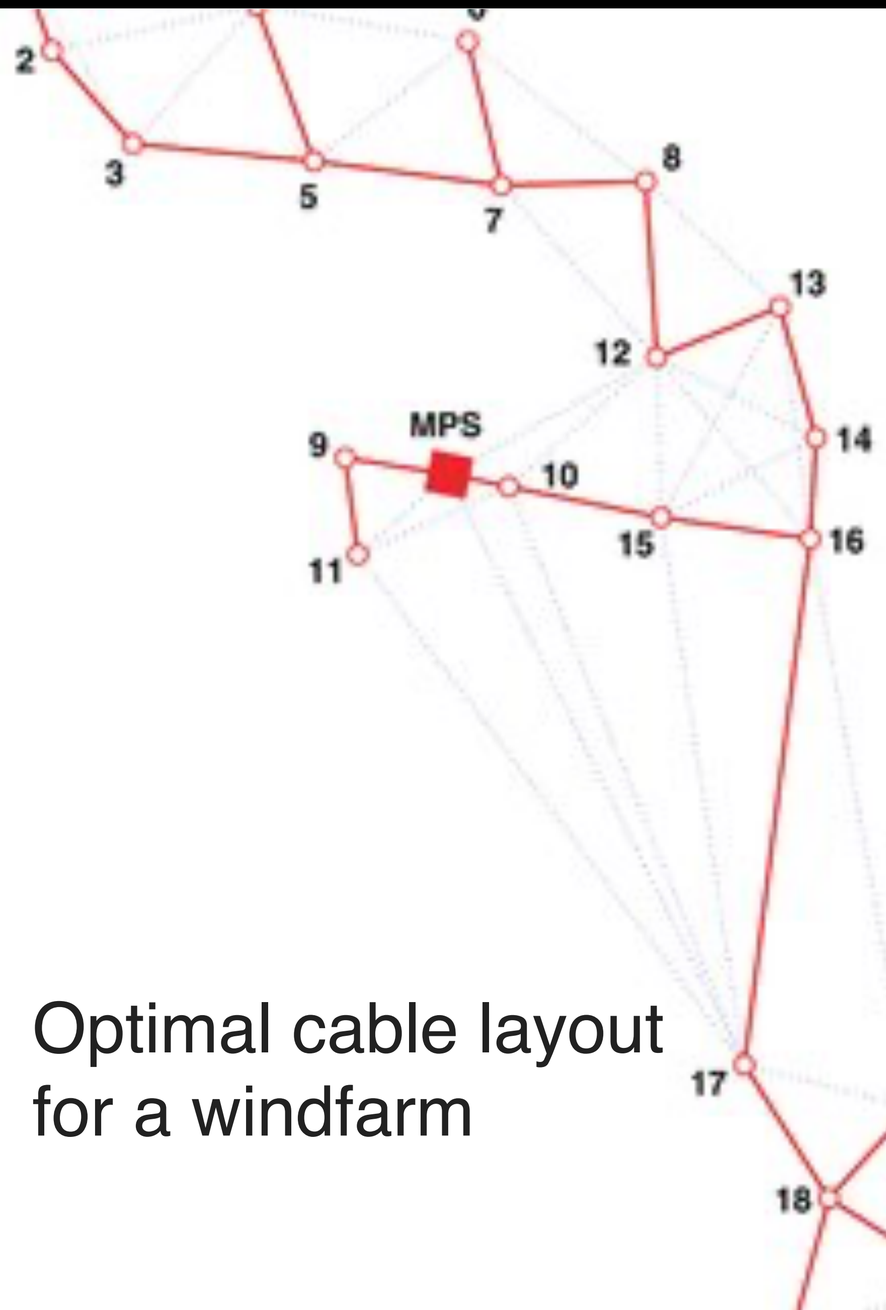
A tree is a connected acyclic graph (= no cycles).

Given a connected, undirected, weighted graph, a spanning tree of the graph is a subgraph that is a tree and connects all the nodes.

A minimum spanning tree (MST) is a spanning tree such that the sum of its link lengths is smaller than or equal to the sum of the link lengths of any other spanning tree.

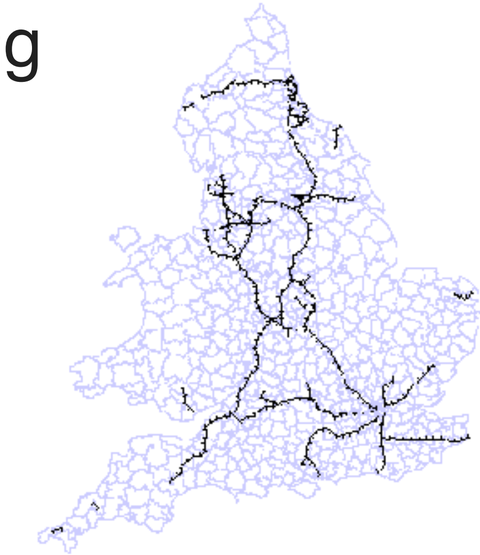


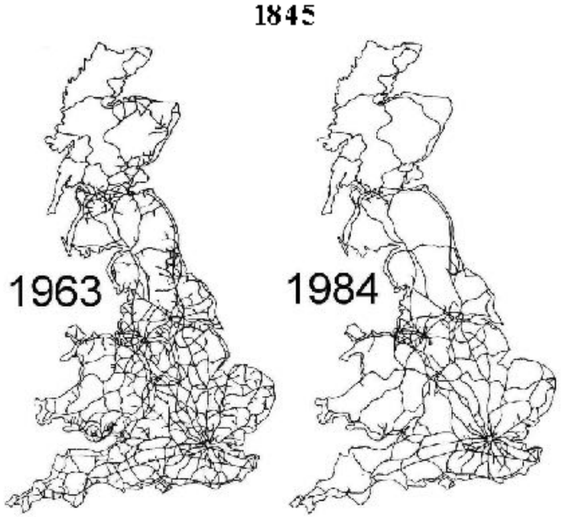
The MST has many applications for spatial networks



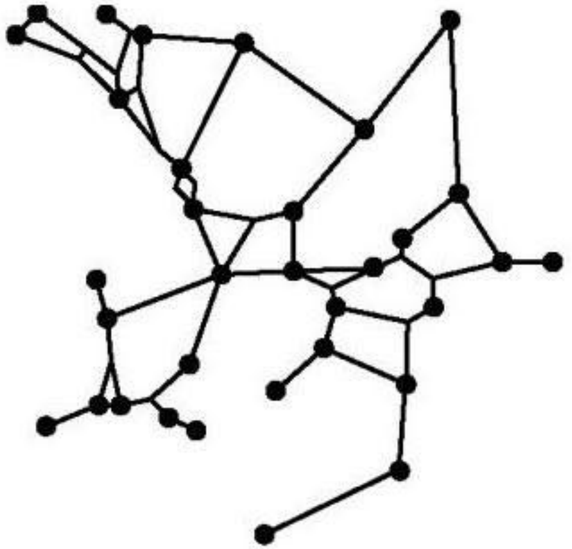
Optimal cable layout for a windfarm

Railroad network planning

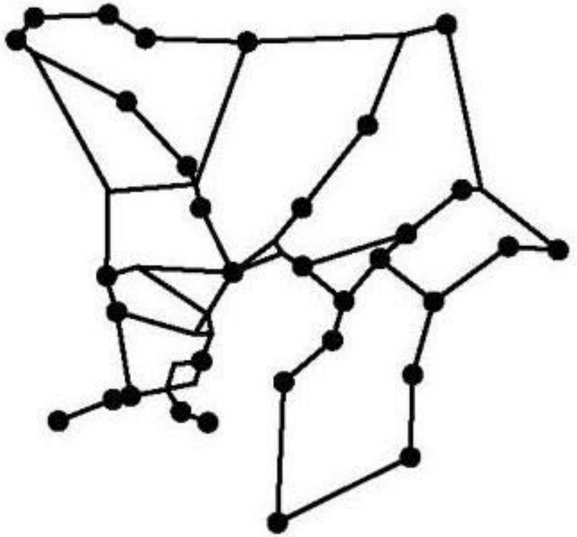




Slime mold network

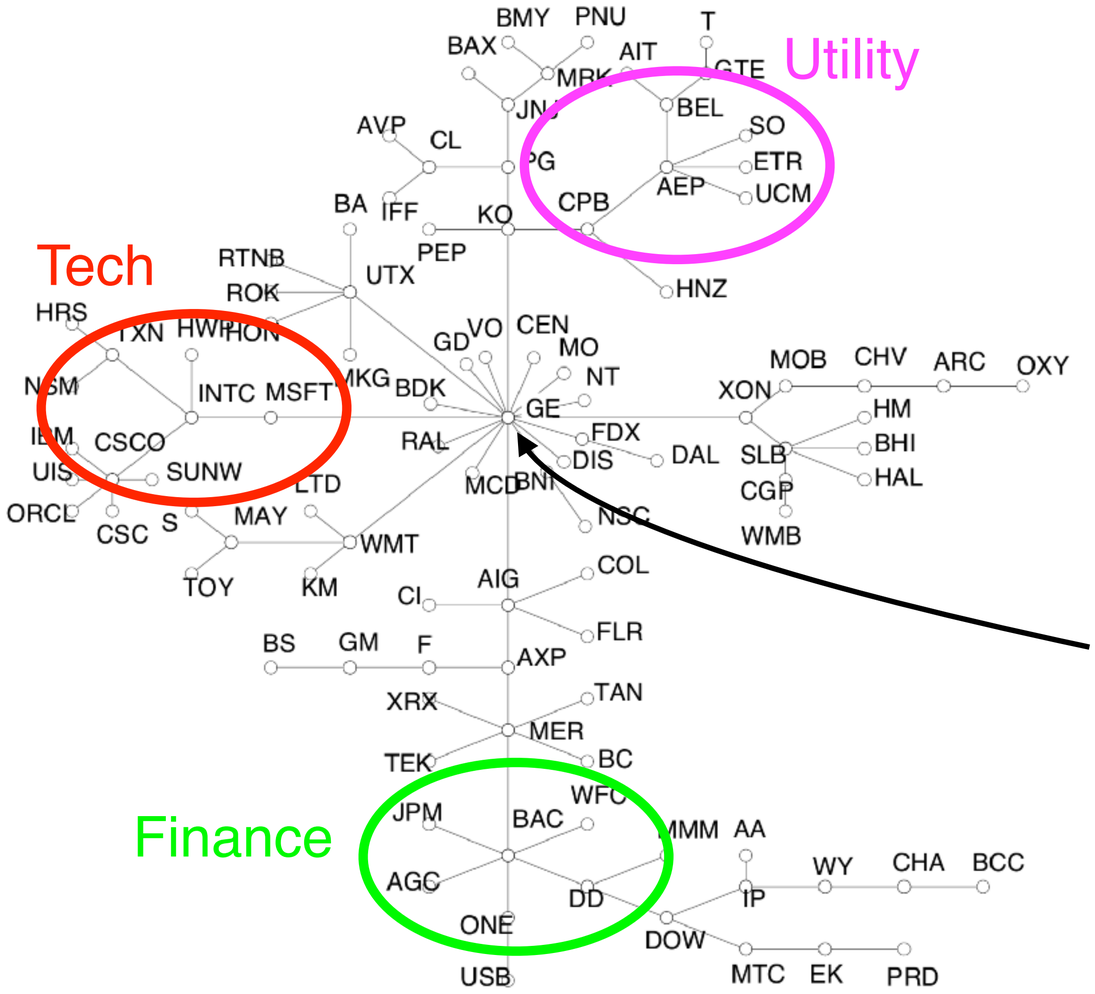


Actual Tokyo railway network



Tero et al. 2010

The MST allows insights into the financial market



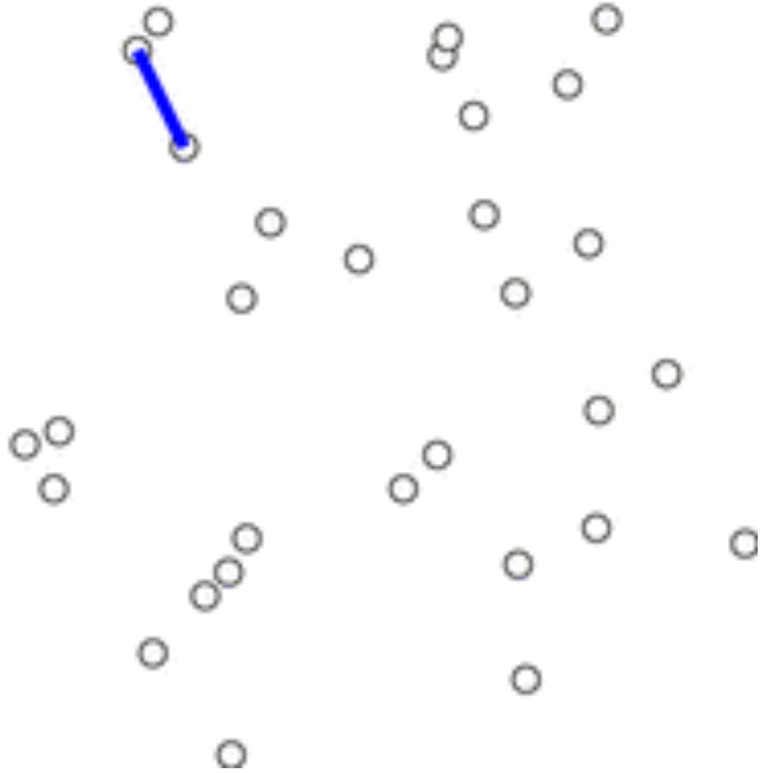
100 stocks (NYSE),
daily returns 1995-1998

GE is a “hierarchical reference stock”

Bonanno, G., Lillo, F., & Mantegna, R. N. (2001). High-frequency cross-correlation in a set of stocks.

Two common algorithms find the MST: Prim, Kruskal

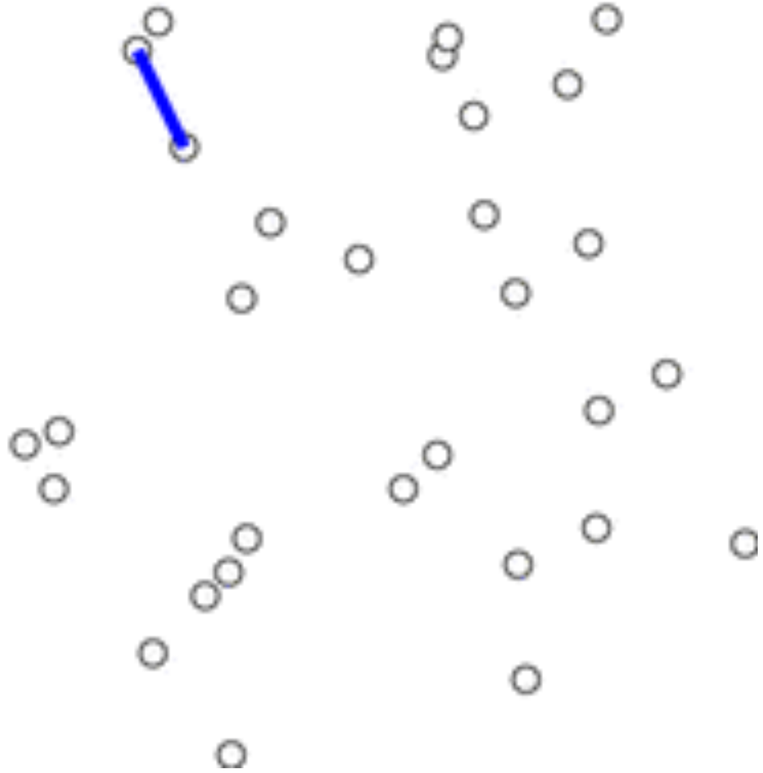
Prim



Grow a tree from a node

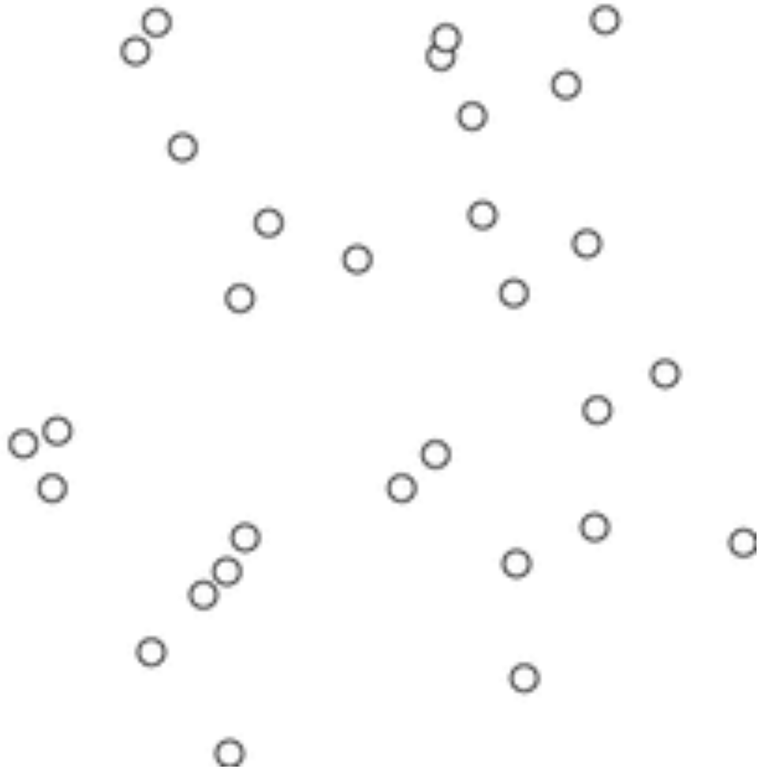
Two common algorithms find the MST: Prim, Kruskal

Prim



Grow a tree from a node

Kruskal



Add lowest-weight edges
if not creating a cycle

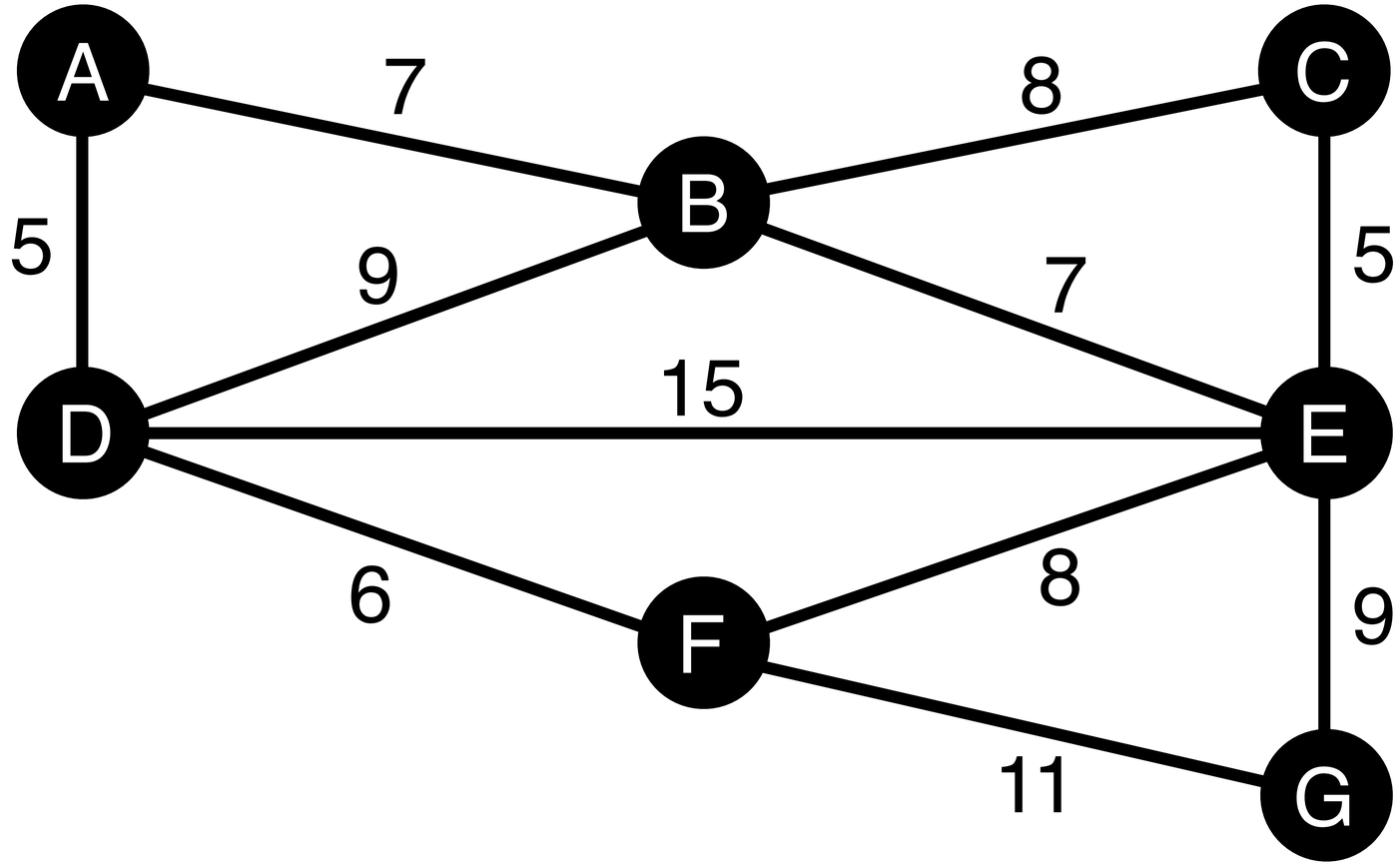
Prim's algorithm:

1. Initialize a tree with a single vertex, chosen arbitrarily from the graph.

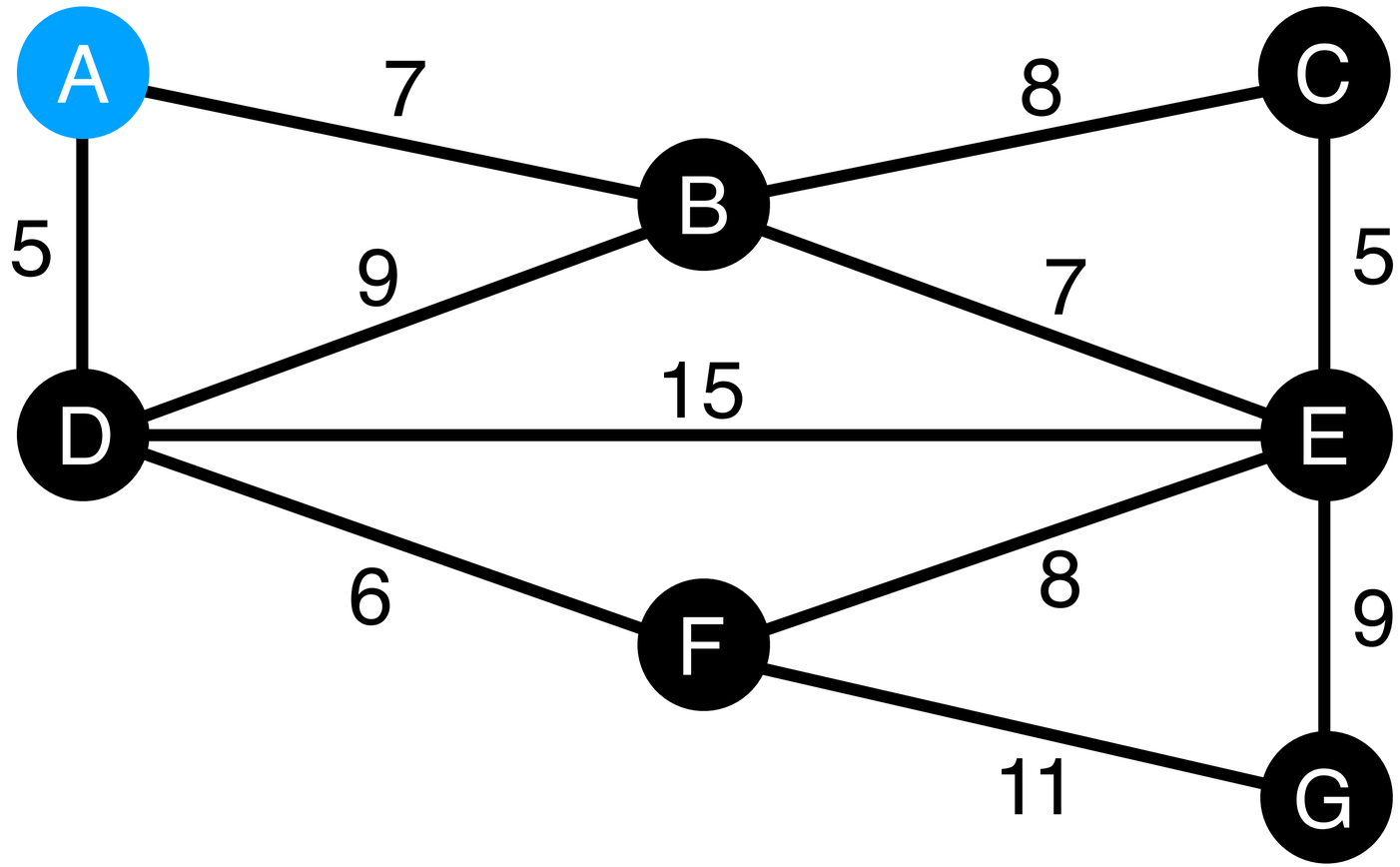
2. Grow the tree by one edge: of the edges that connect the tree to vertices not yet in the tree (the frontier), find the minimum-weight edge, and transfer it to the tree.

3. Repeat step 2 (until all vertices are in the tree).

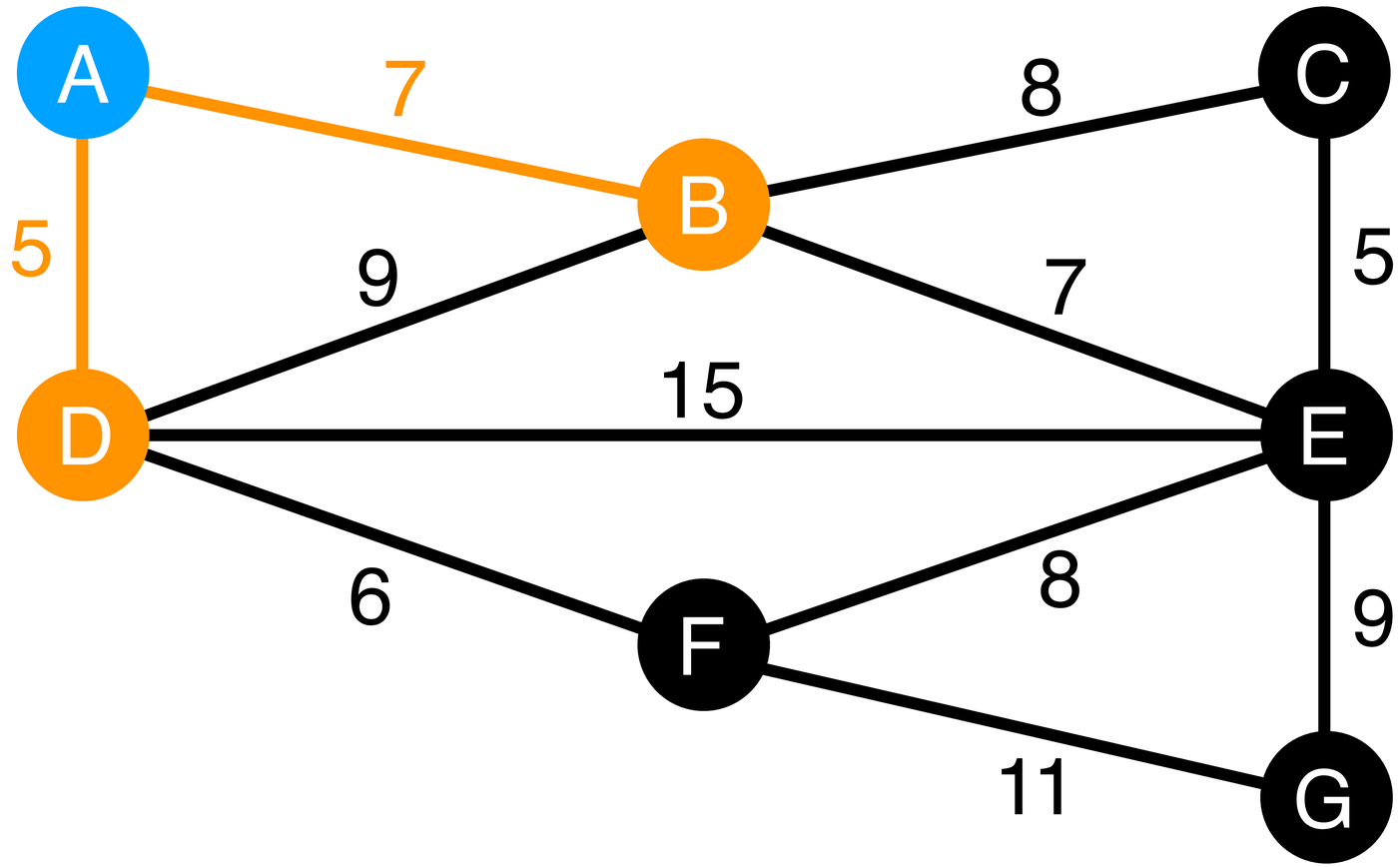
Prim's algorithm example



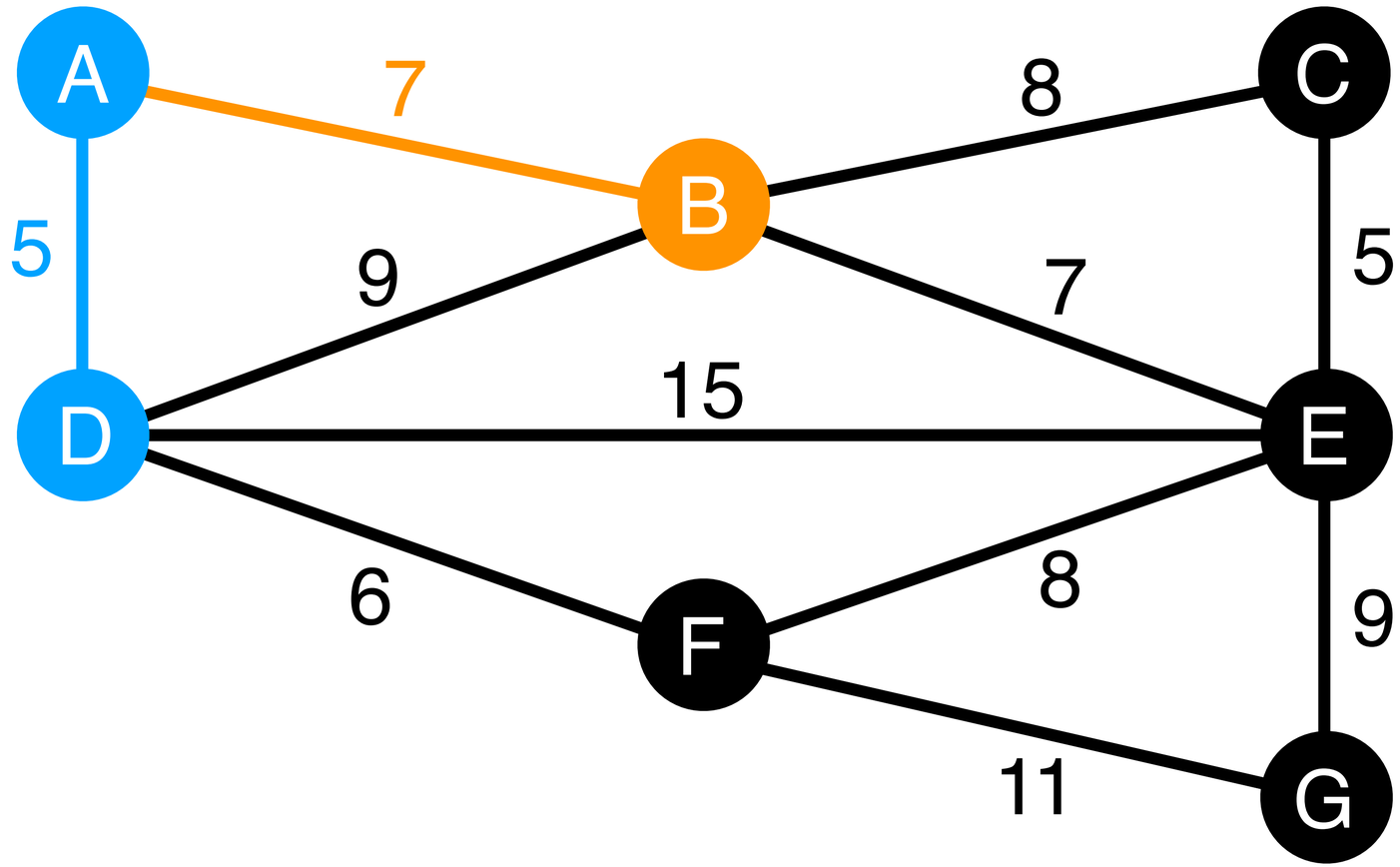
Prim's algorithm example: Choose arbitrary starting point



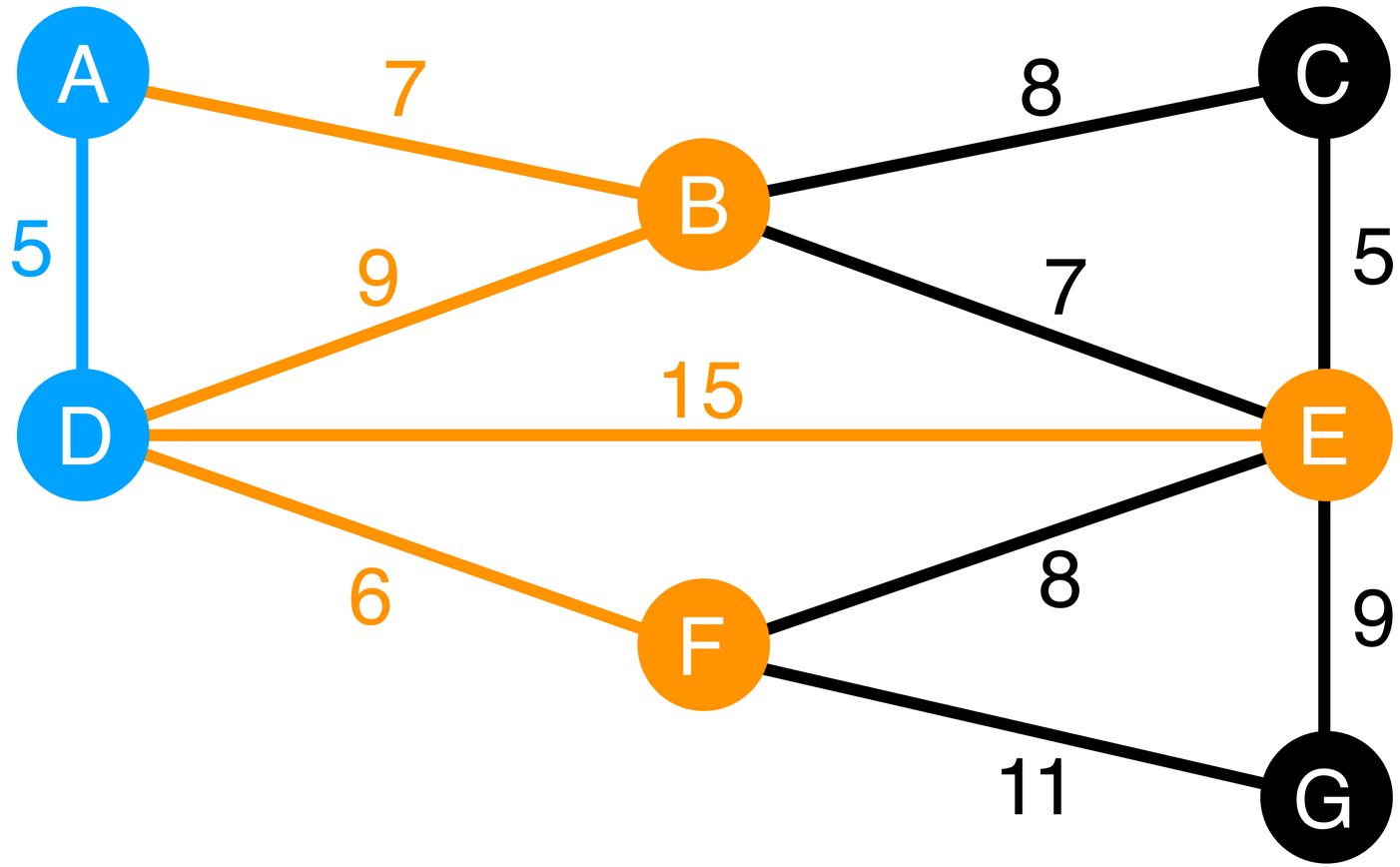
Prim's algorithm example: Update frontier



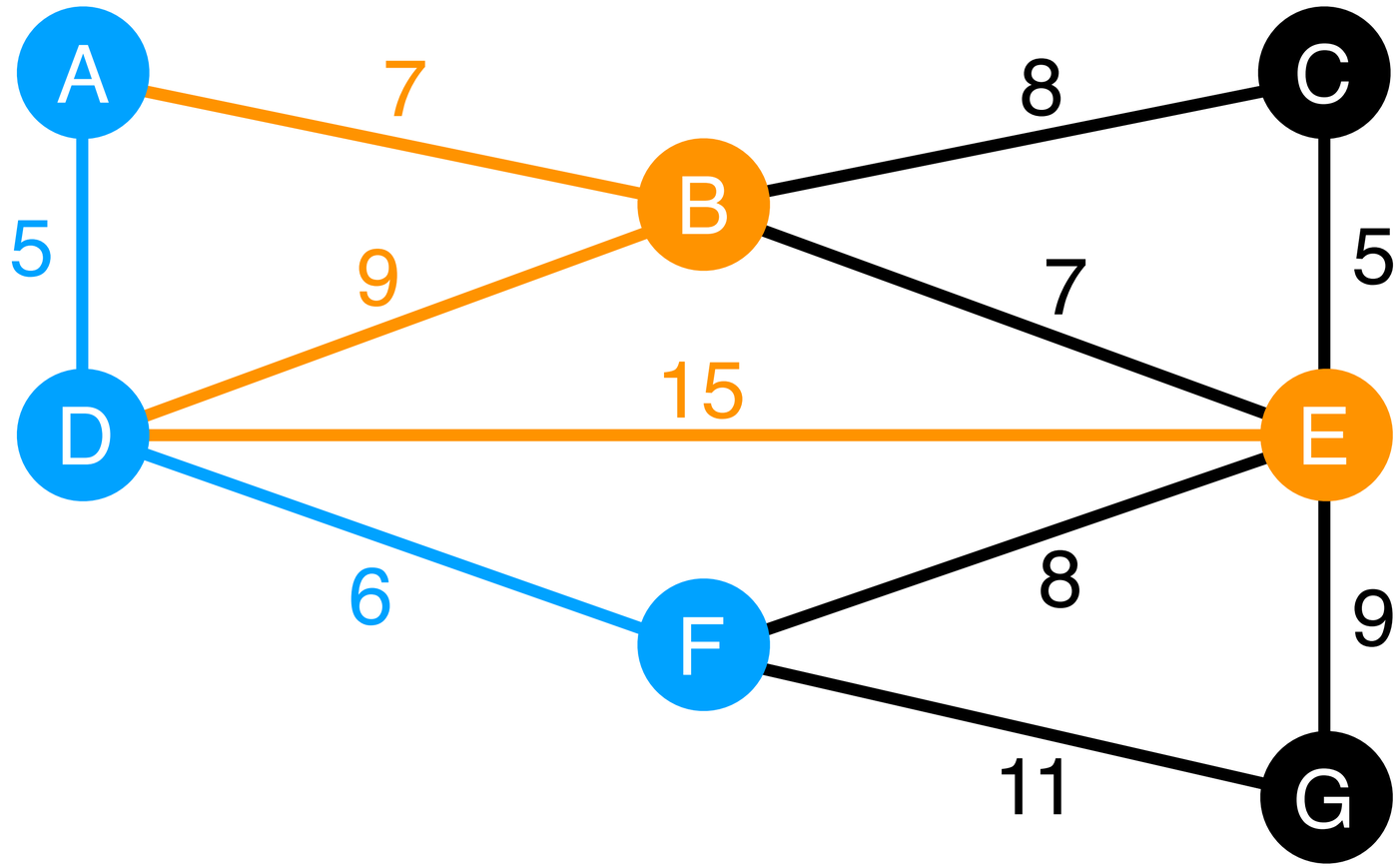
Prim's algorithm example: Choose lowest weight



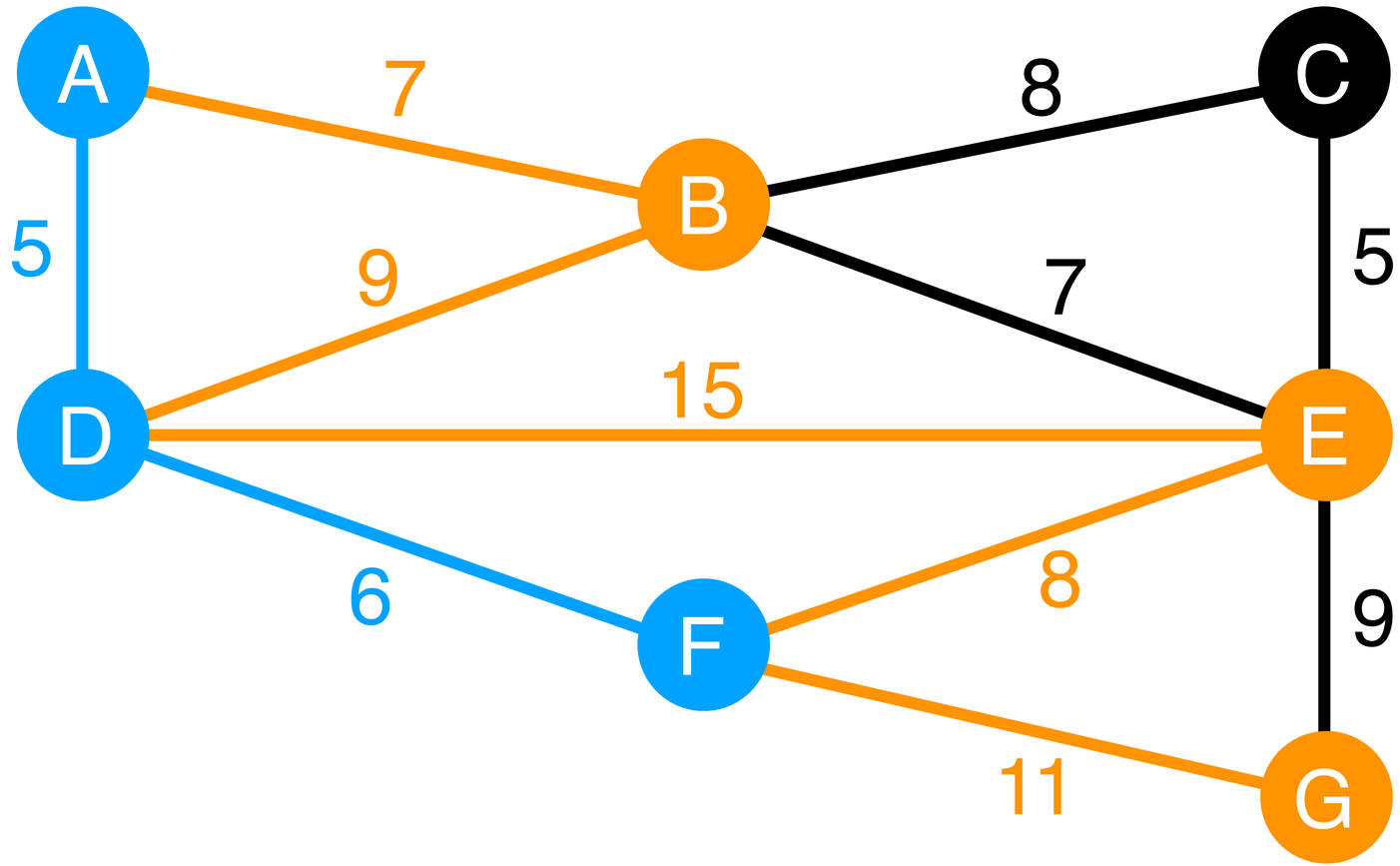
Prim's algorithm example: Update frontier



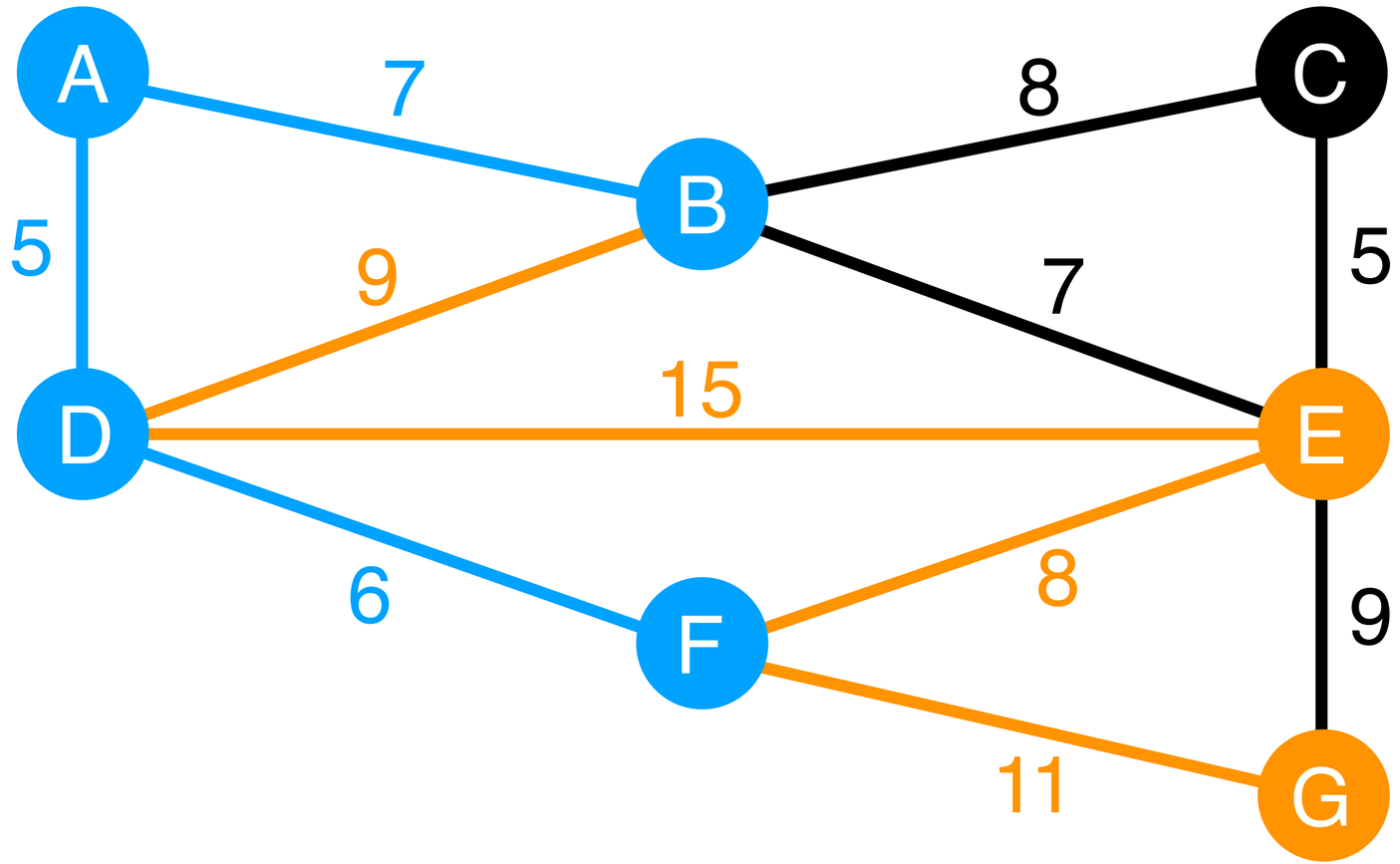
Prim's algorithm example: Choose lowest weight



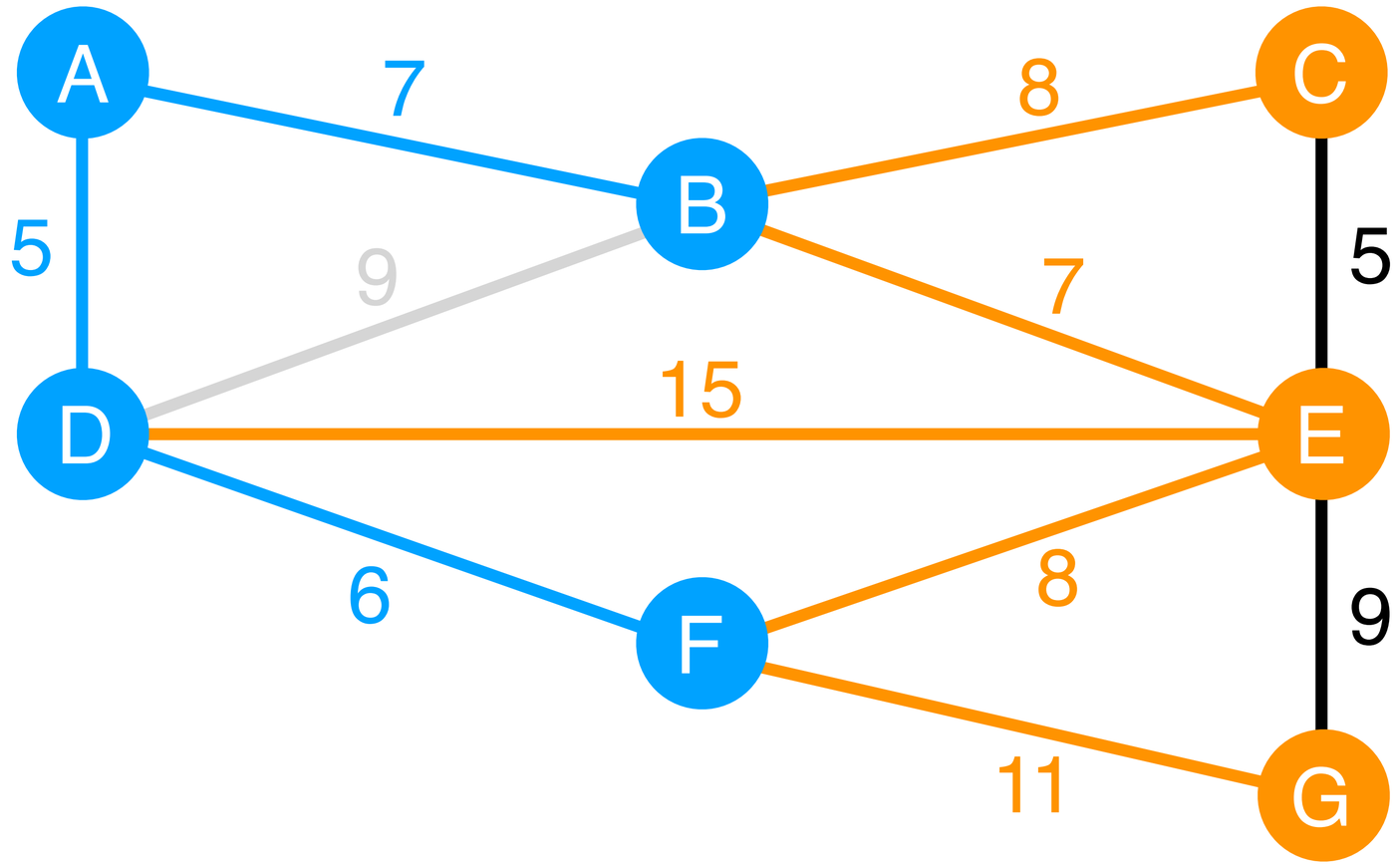
Prim's algorithm example: Update frontier



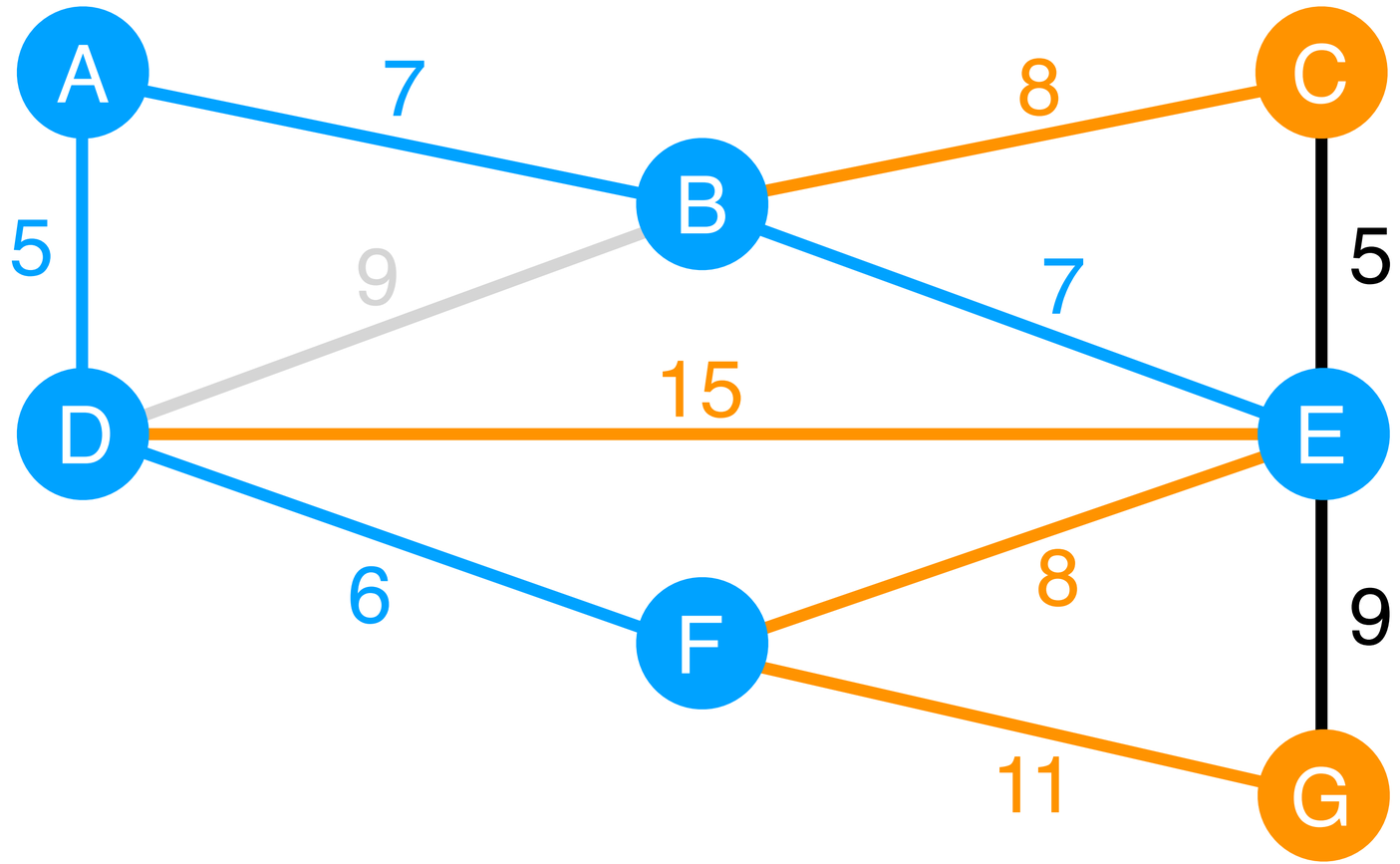
Prim's algorithm example: Choose lowest weight



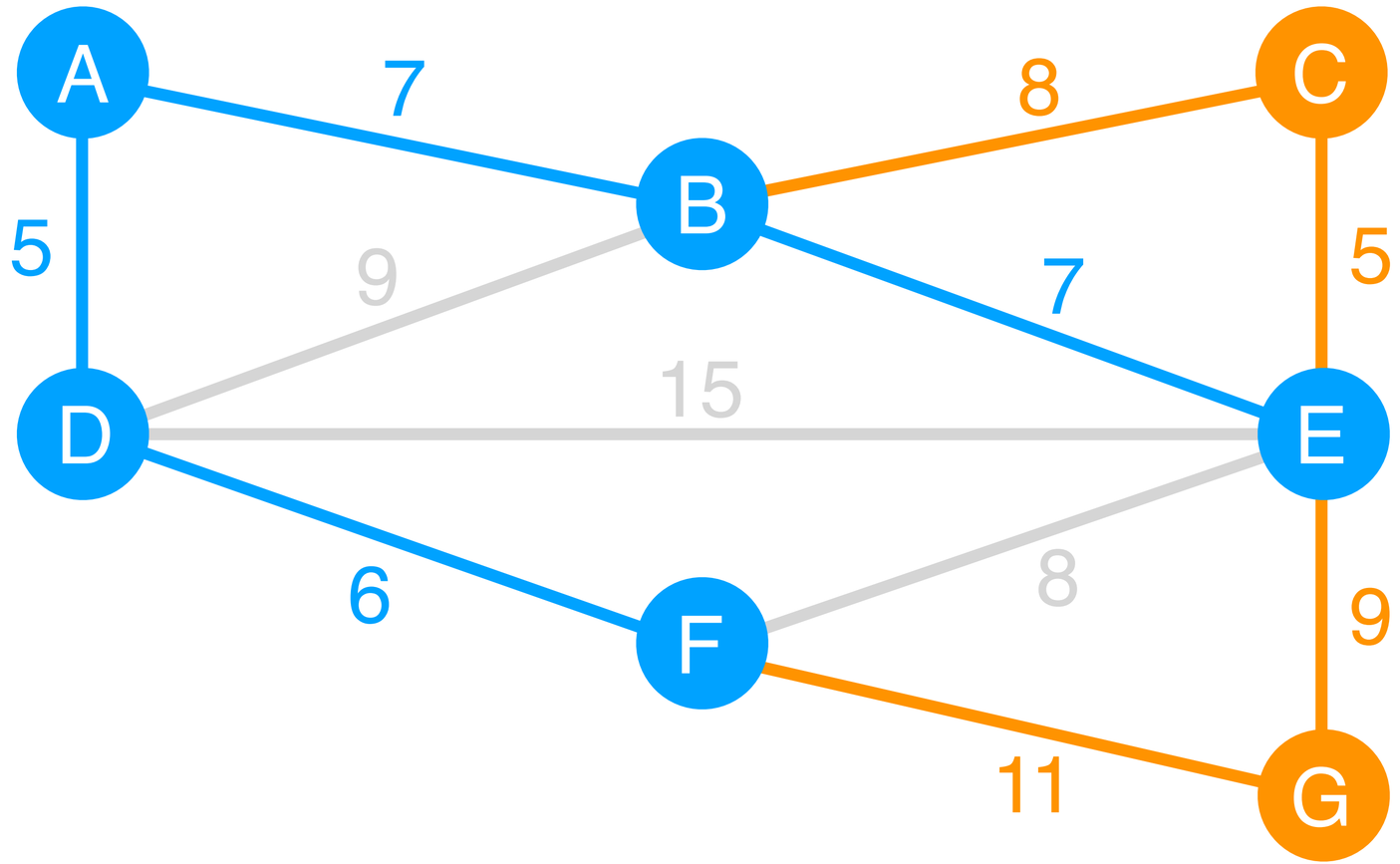
Prim's algorithm example: Update frontier



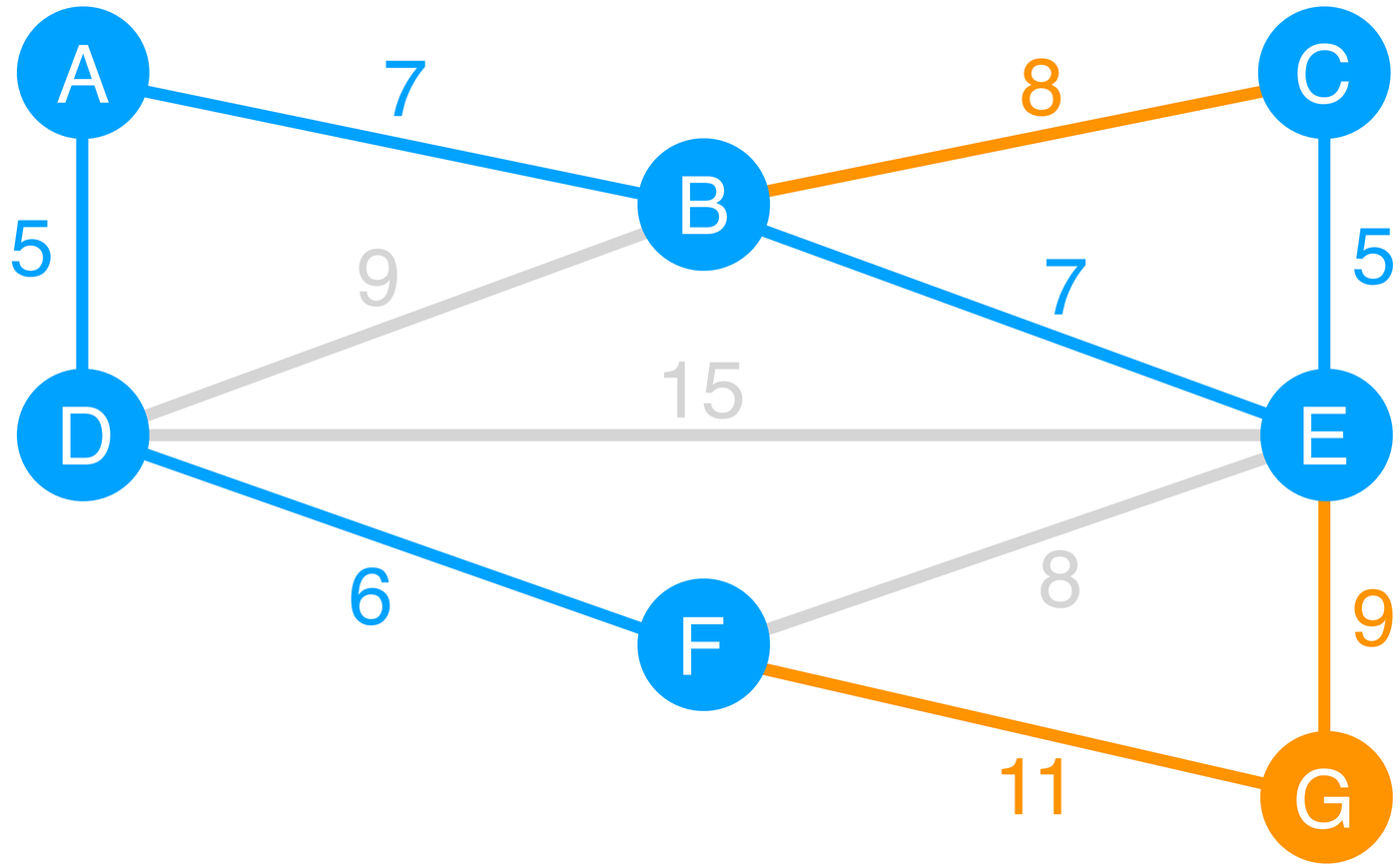
Prim's algorithm example: Choose lowest weight



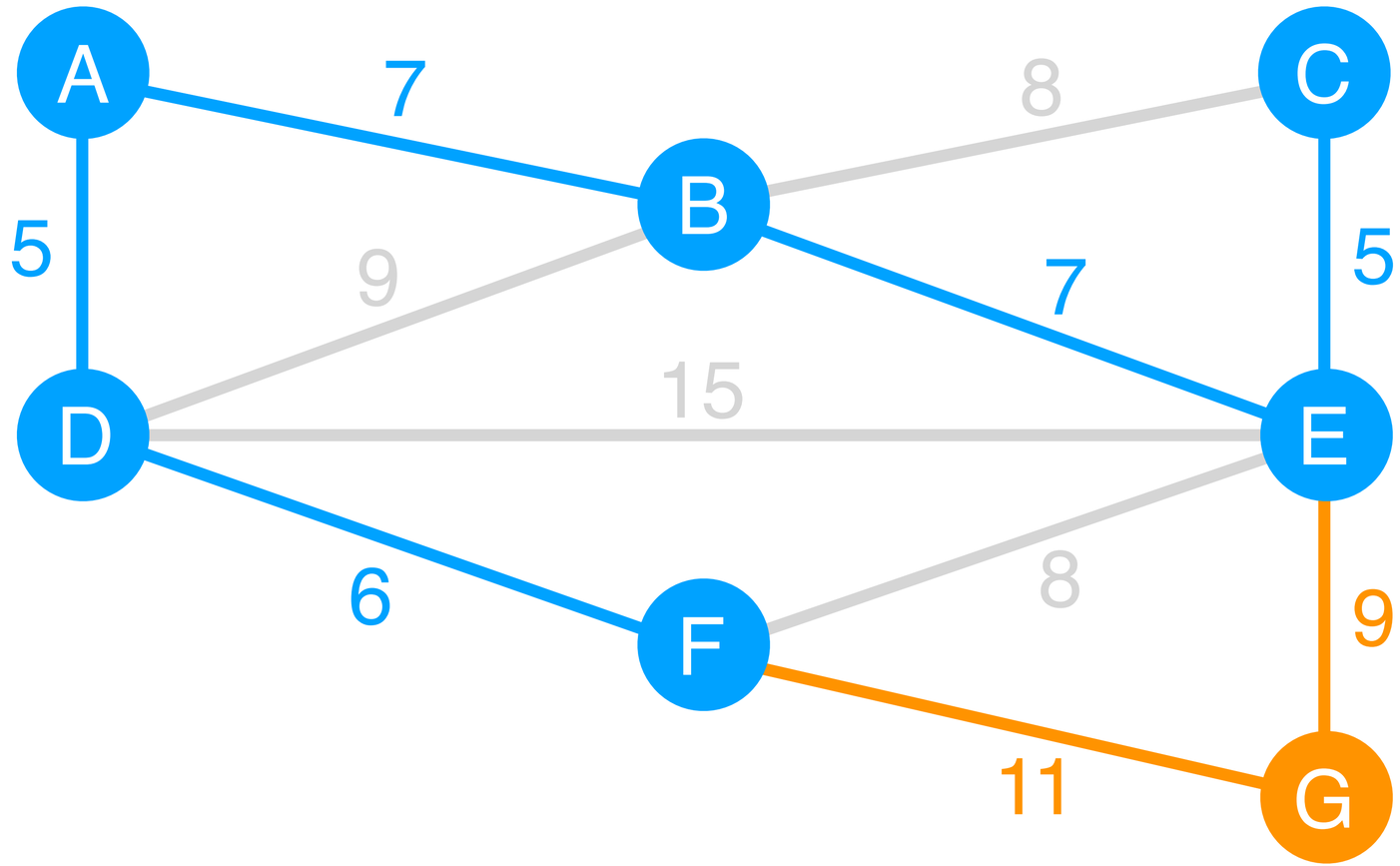
Prim's algorithm example: Update frontier



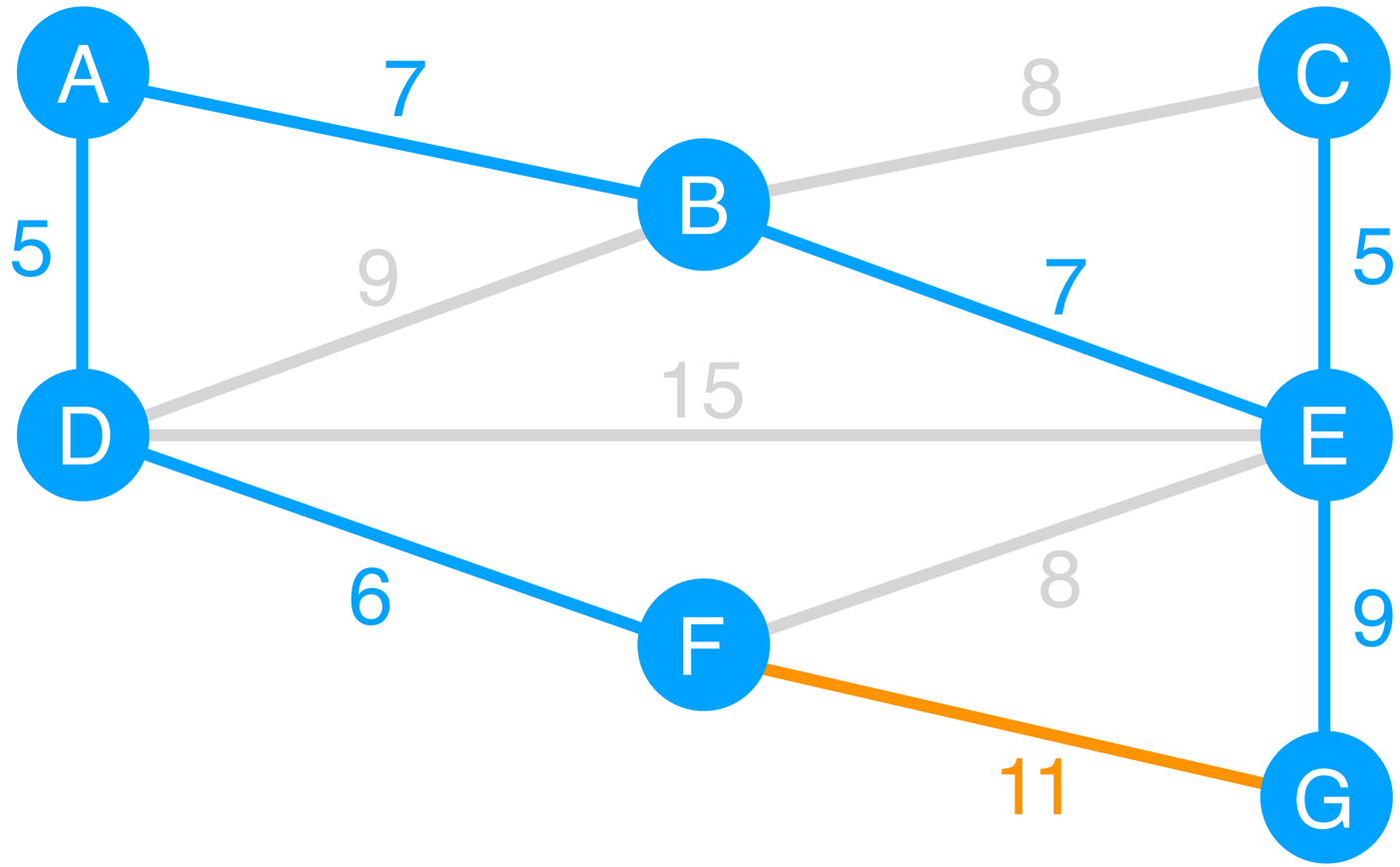
Prim's algorithm example: Choose lowest weight



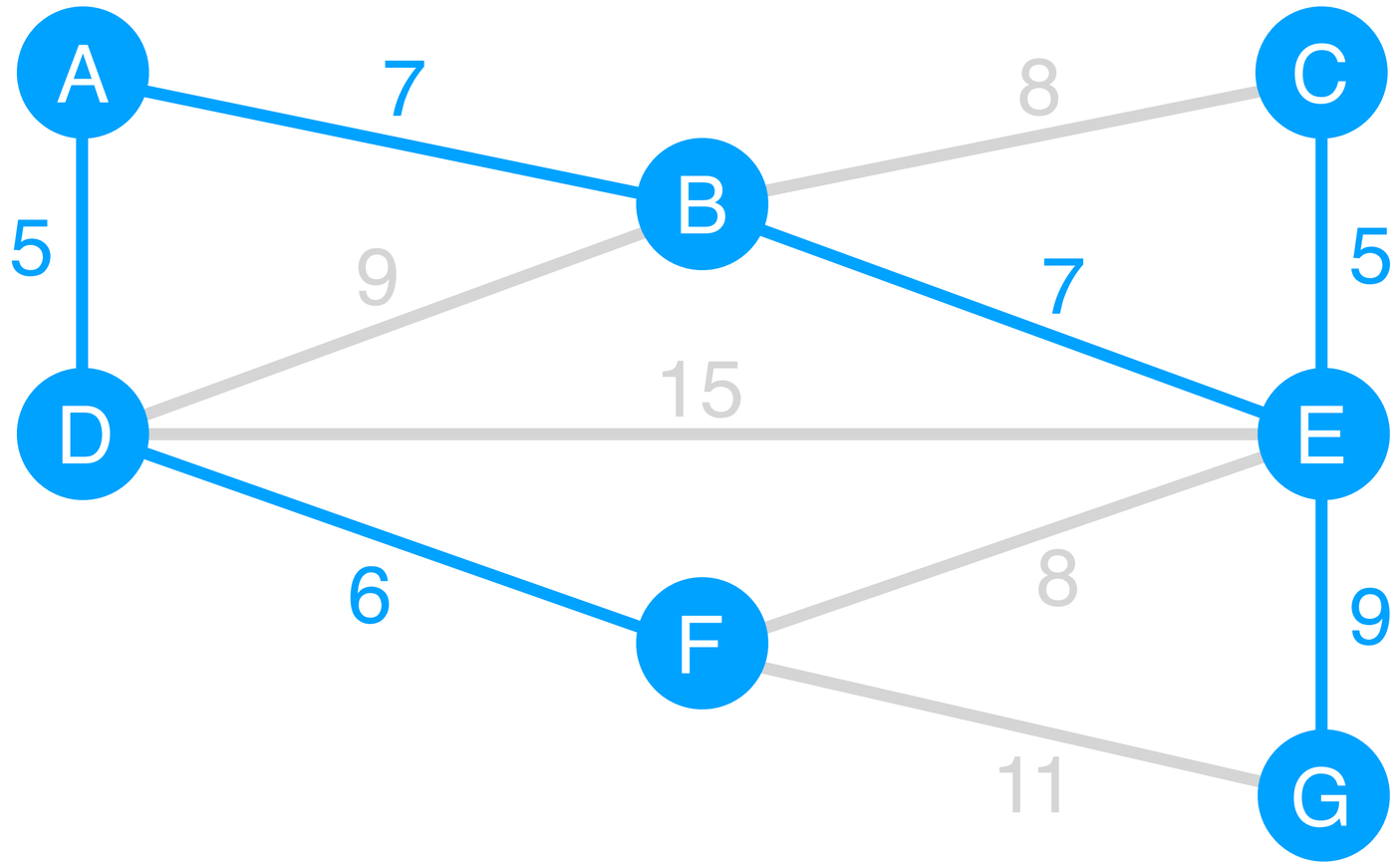
Prim's algorithm example: Update frontier



Prim's algorithm example: Choose lowest weight



Prim's algorithm example: All nodes covered - Finished!



Kruskal's algorithm:

1. Sort all edges by increasing weight
2. Add the lowest weight edge, if it does not create a cycle
3. Repeat step 2 (until all vertices are in the tree).

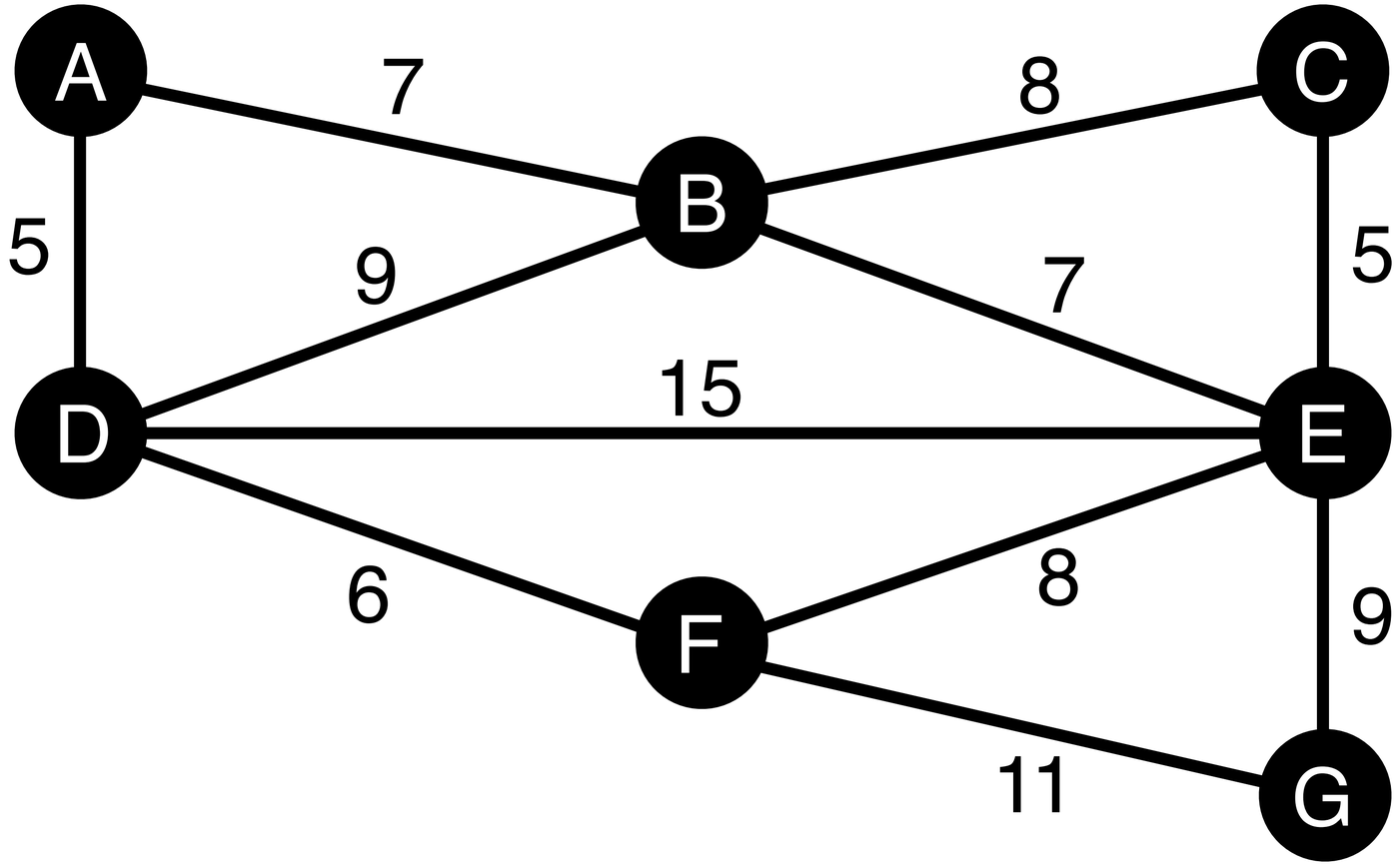
Kruskal's algorithm:

1. Sort all edges by increasing weight
2. Add the lowest weight edge, if it does not create a cycle
3. Repeat step 2 (until all vertices are in the tree).

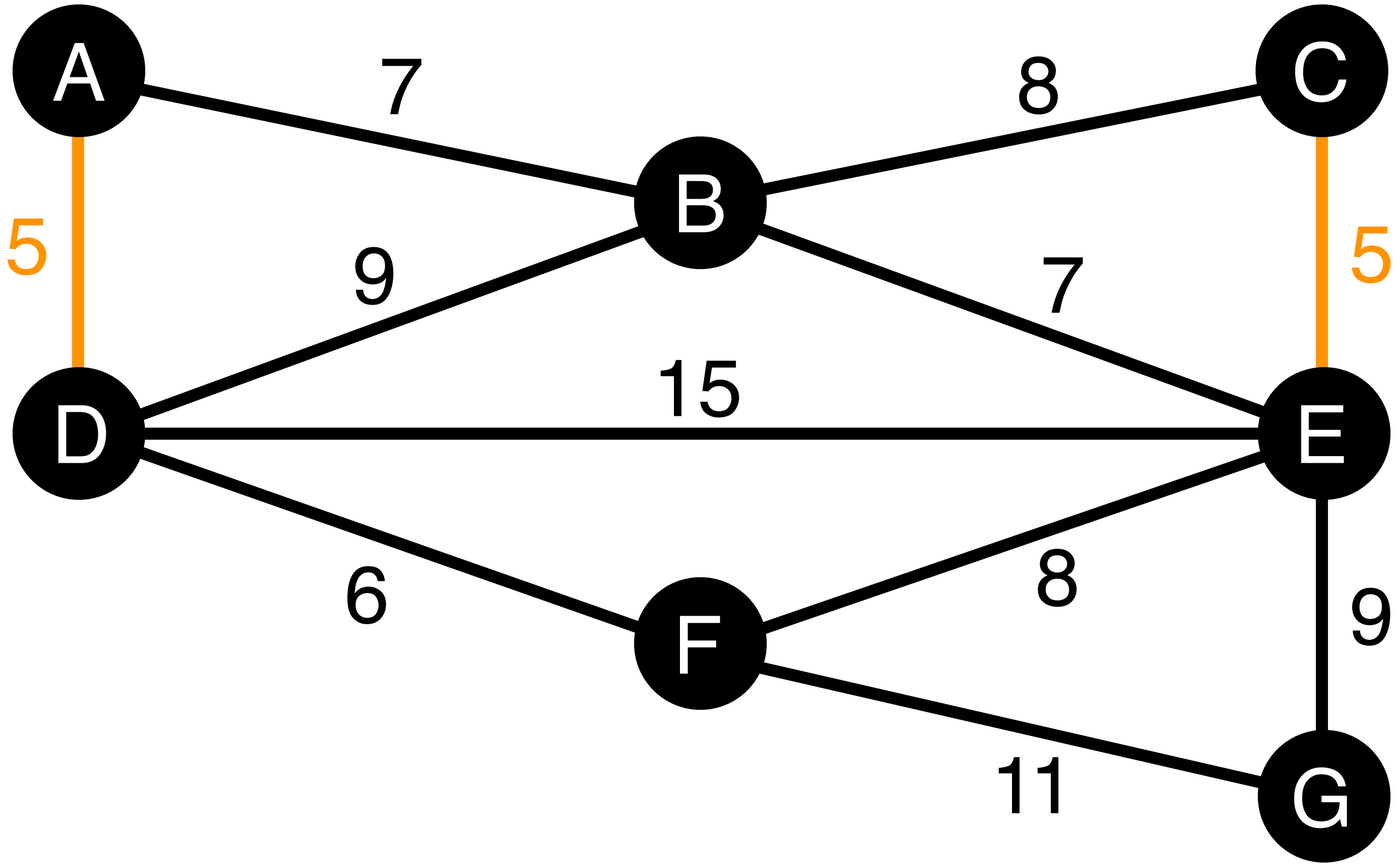
How do we check this?

https://www.mentimeter.com/s/b41540d000f6a33c29e74720863543ba/9e0e0845bee6

Kruskal's algorithm example



Kruskal's algorithm example: Add the lowest-weight link



Union-find

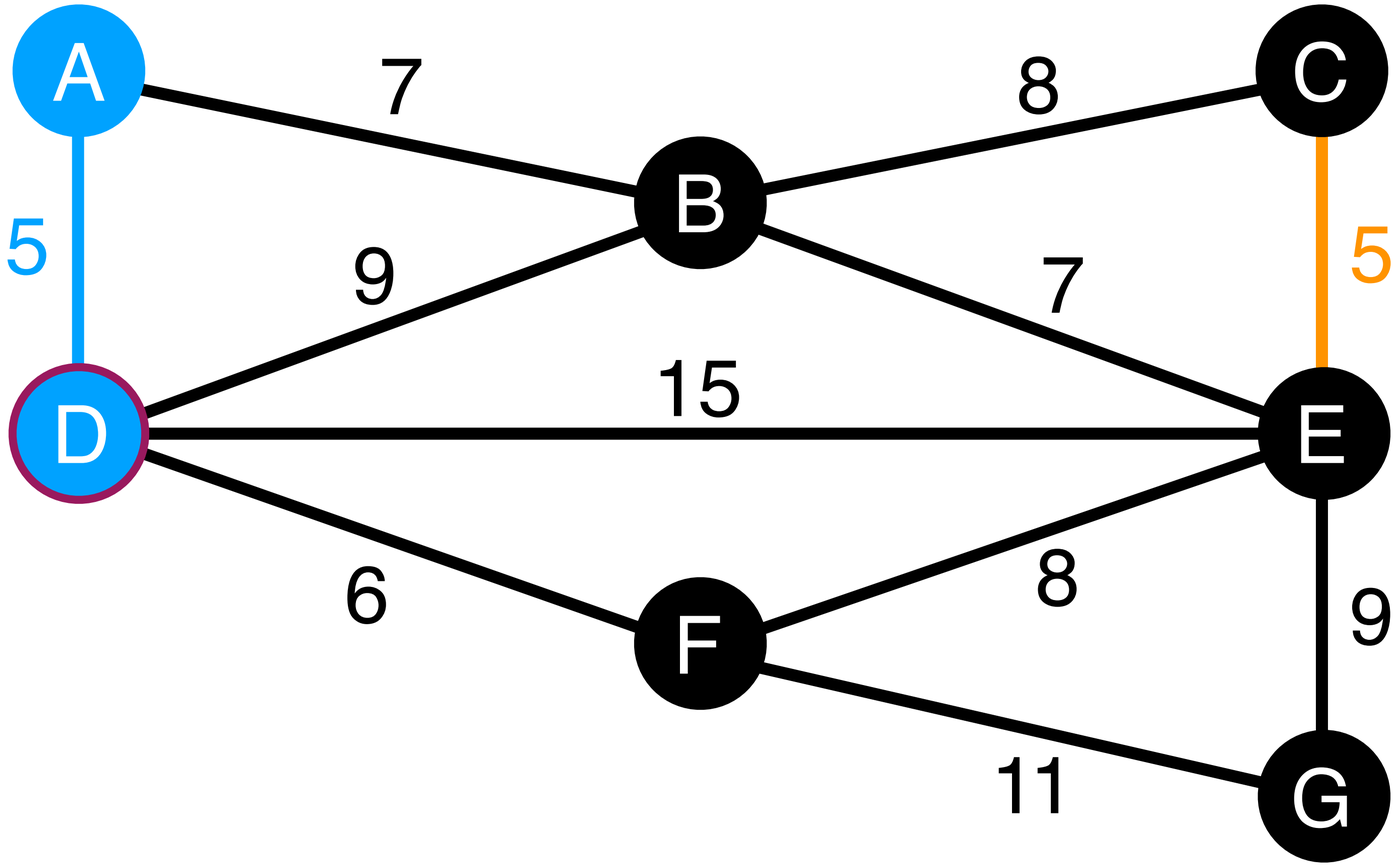
Rank

2

1

0

Kruskal's algorithm example: Add the lowest-weight link



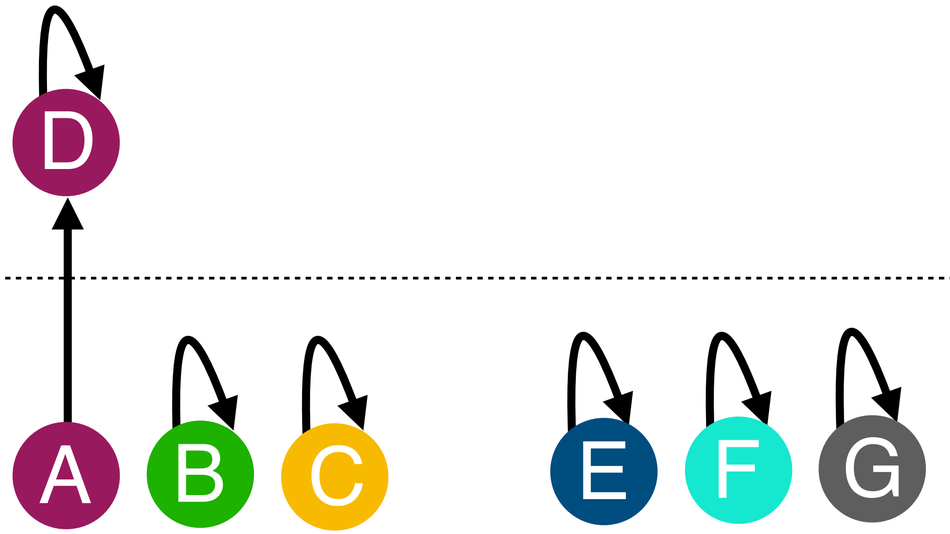
Union-find

Rank

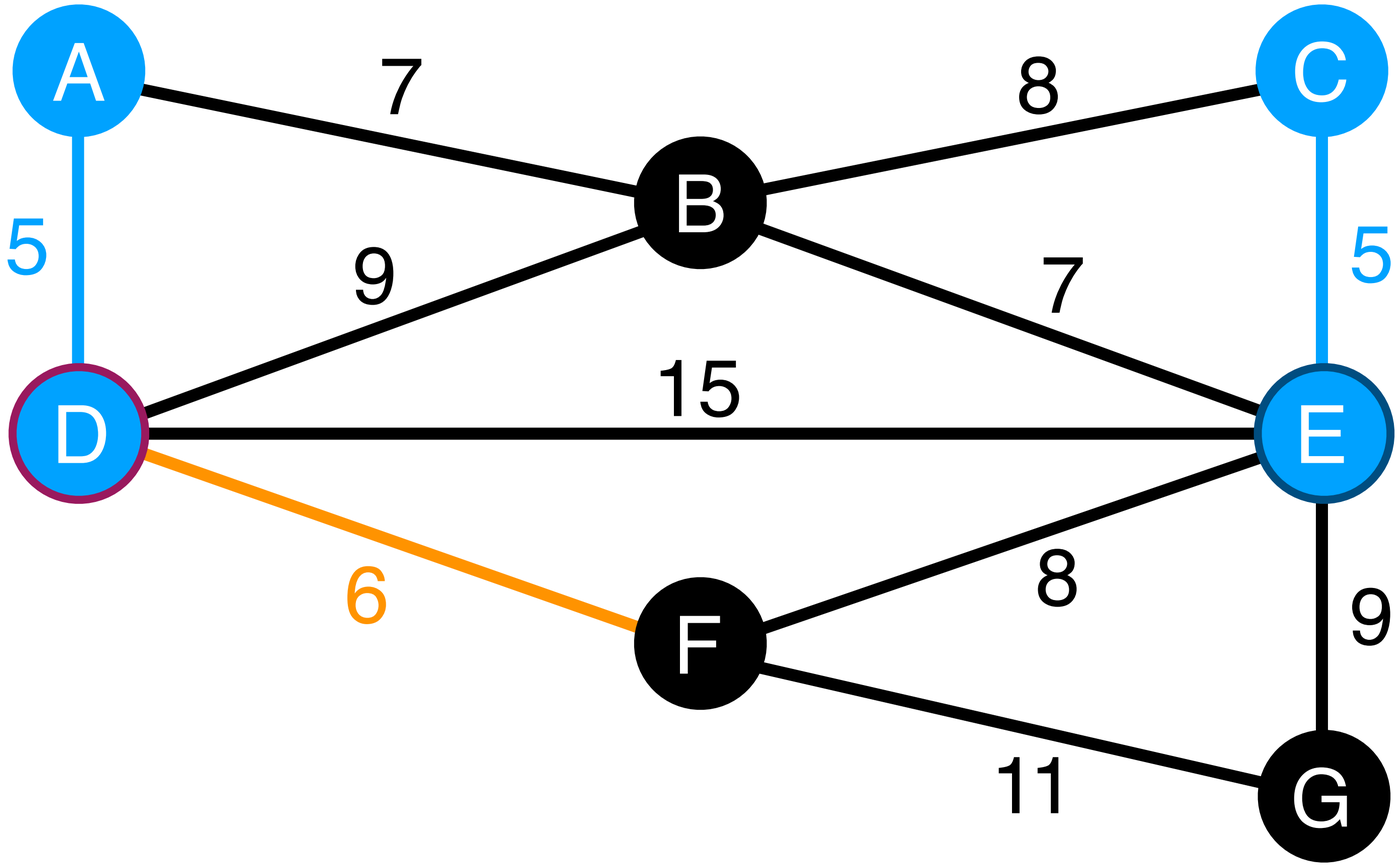
2

1

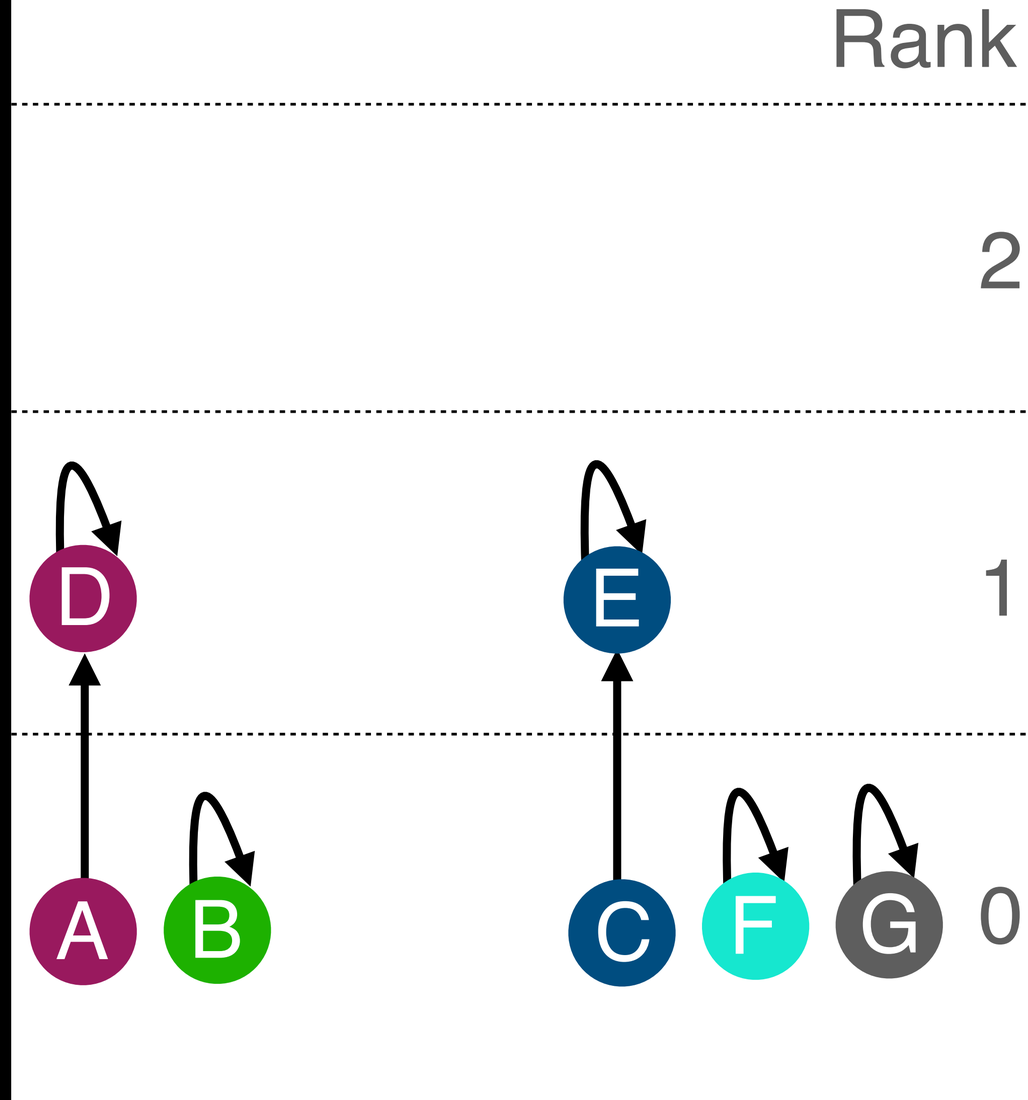
0



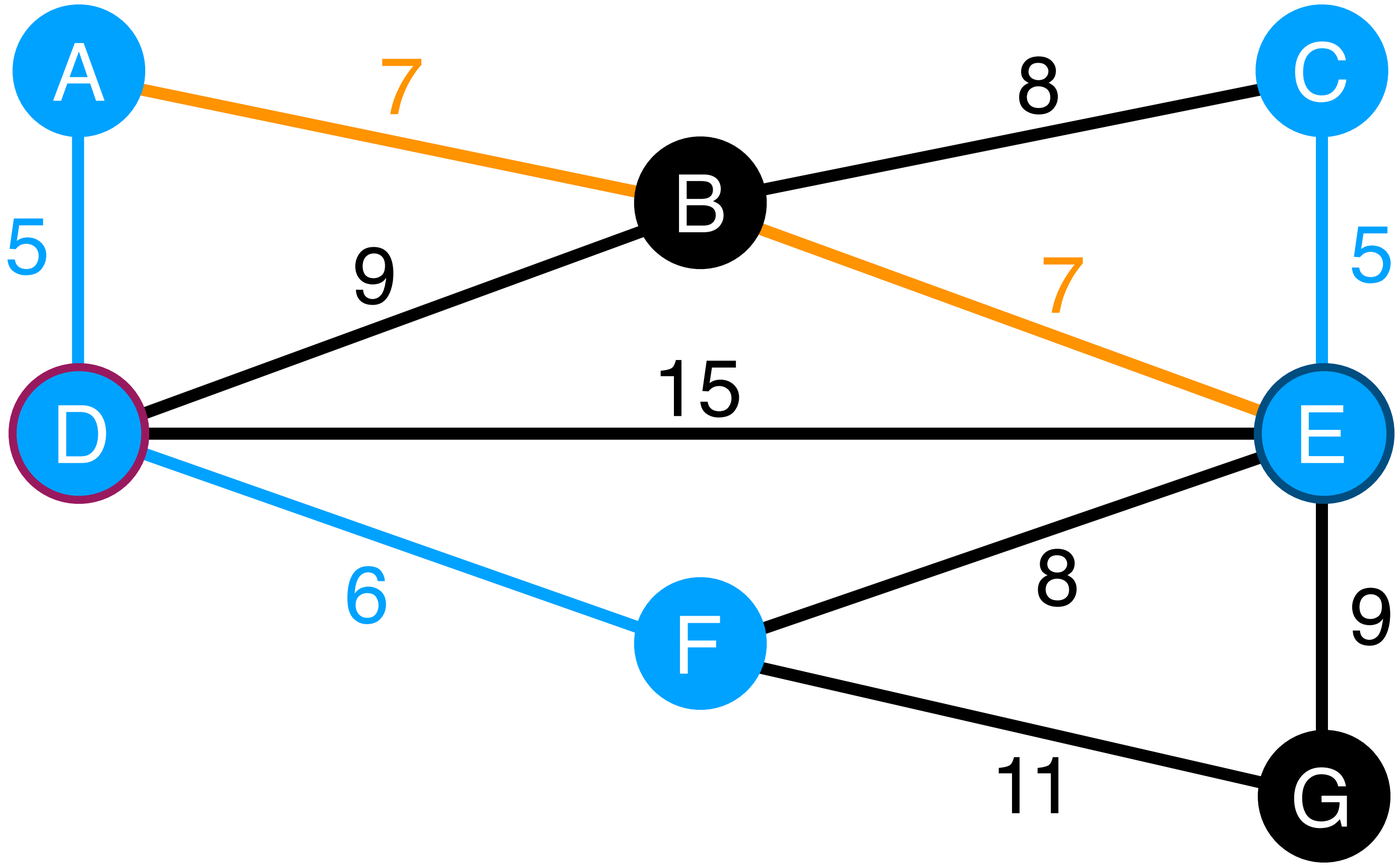
Kruskal's algorithm example: Add the lowest-weight link



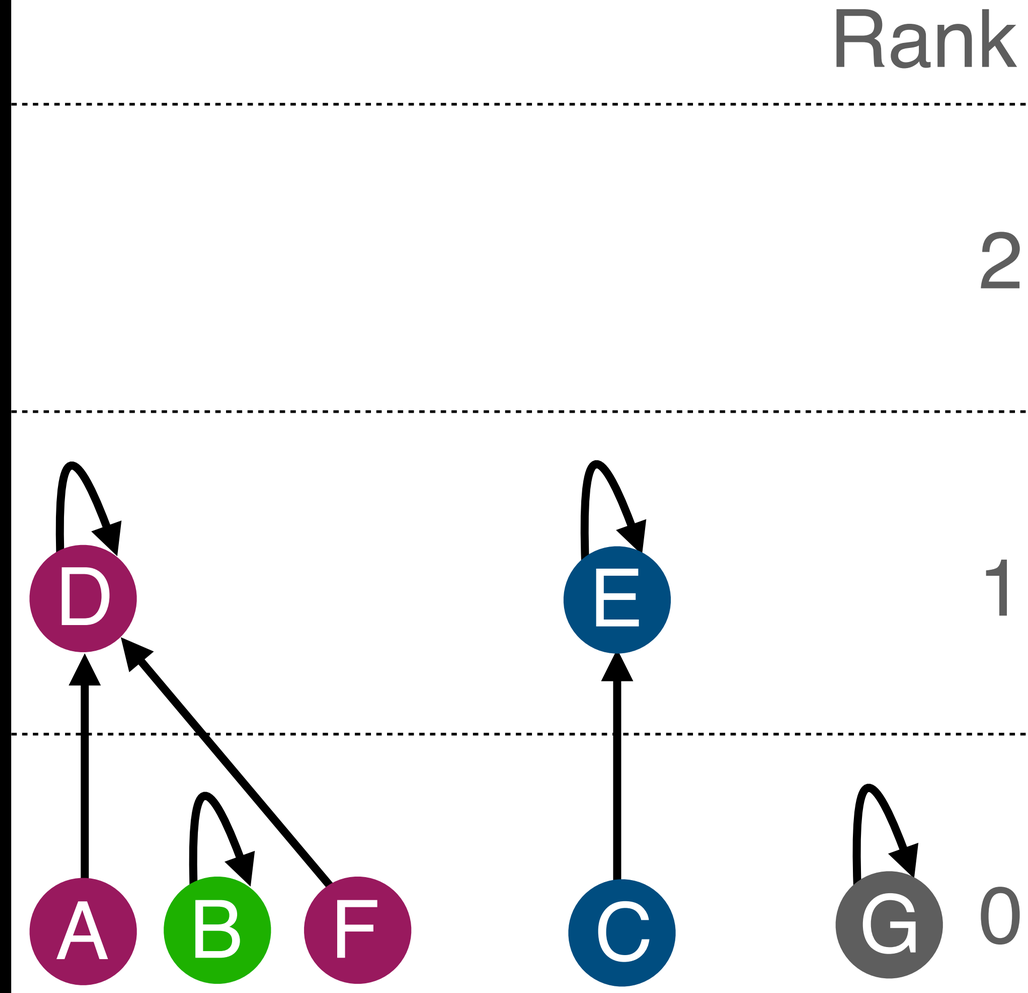
Union-find



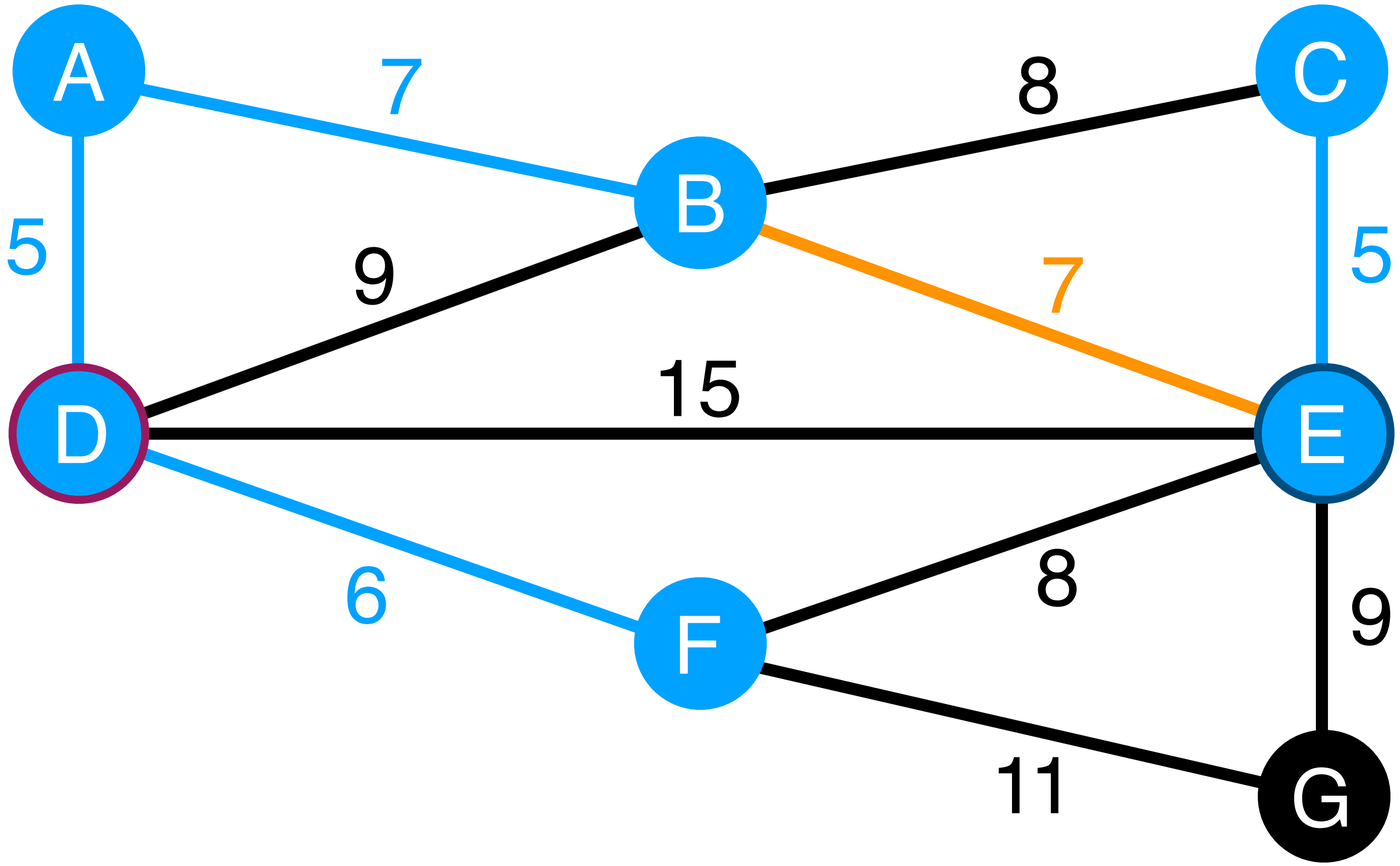
Kruskal's algorithm example: Add the lowest-weight link



Union-find



Kruskal's algorithm example: Add the lowest-weight link



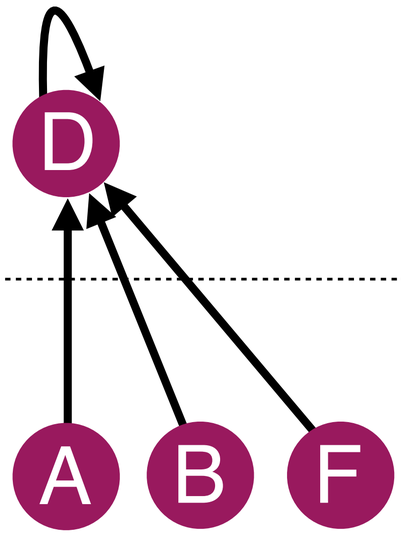
Union-find

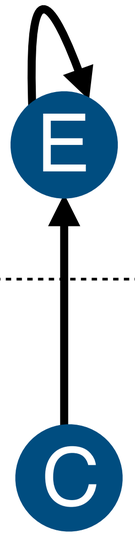
Rank

2

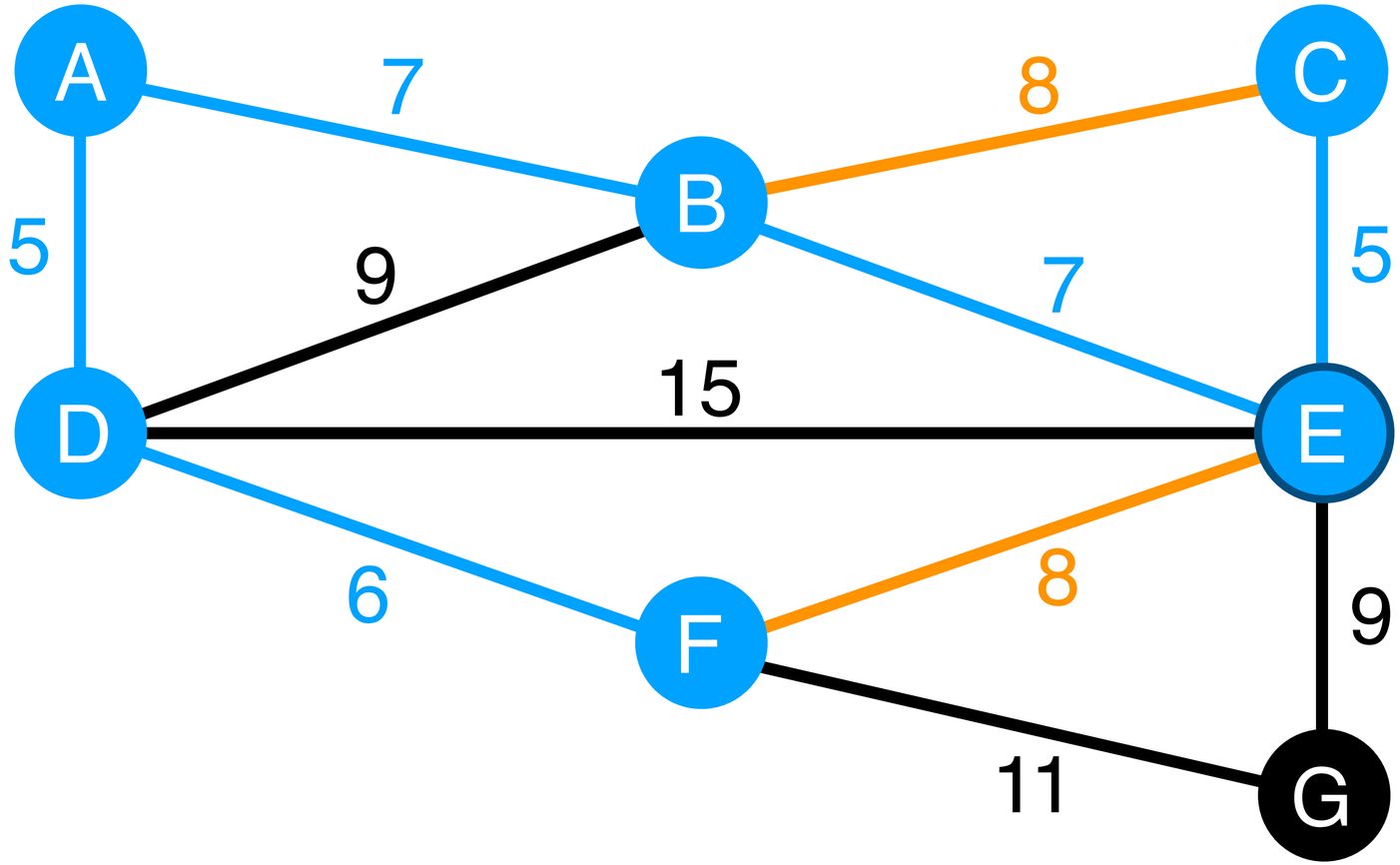
1

0

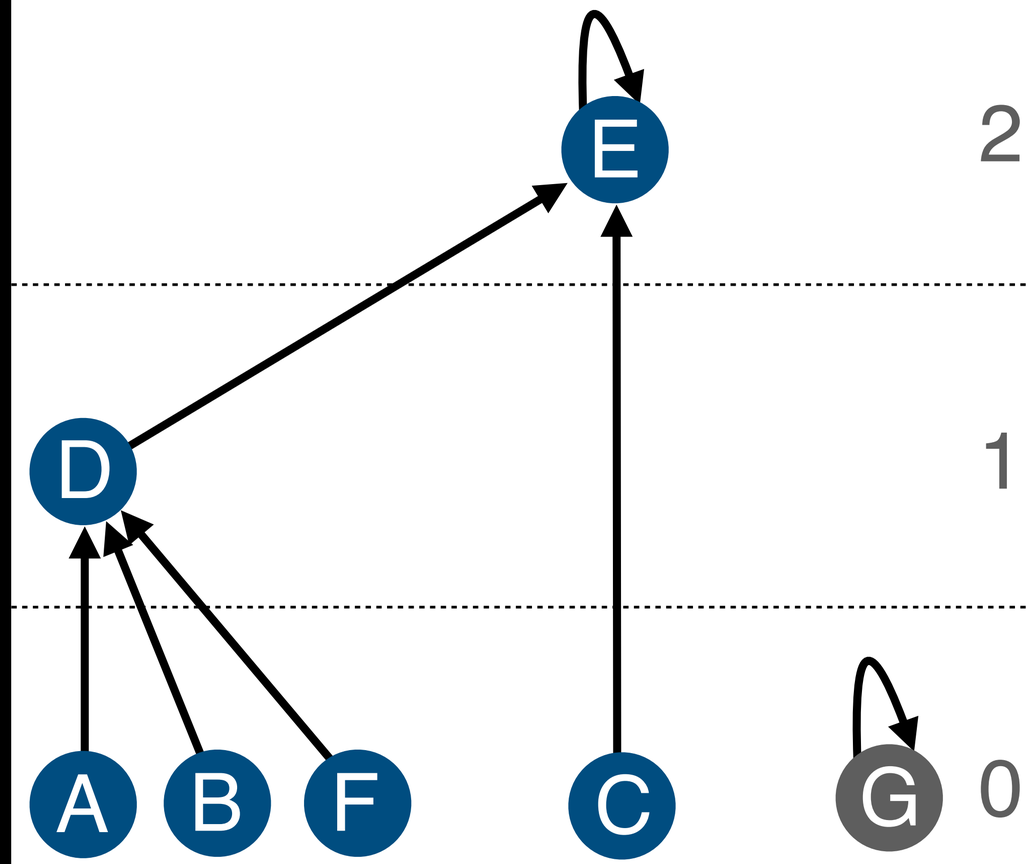




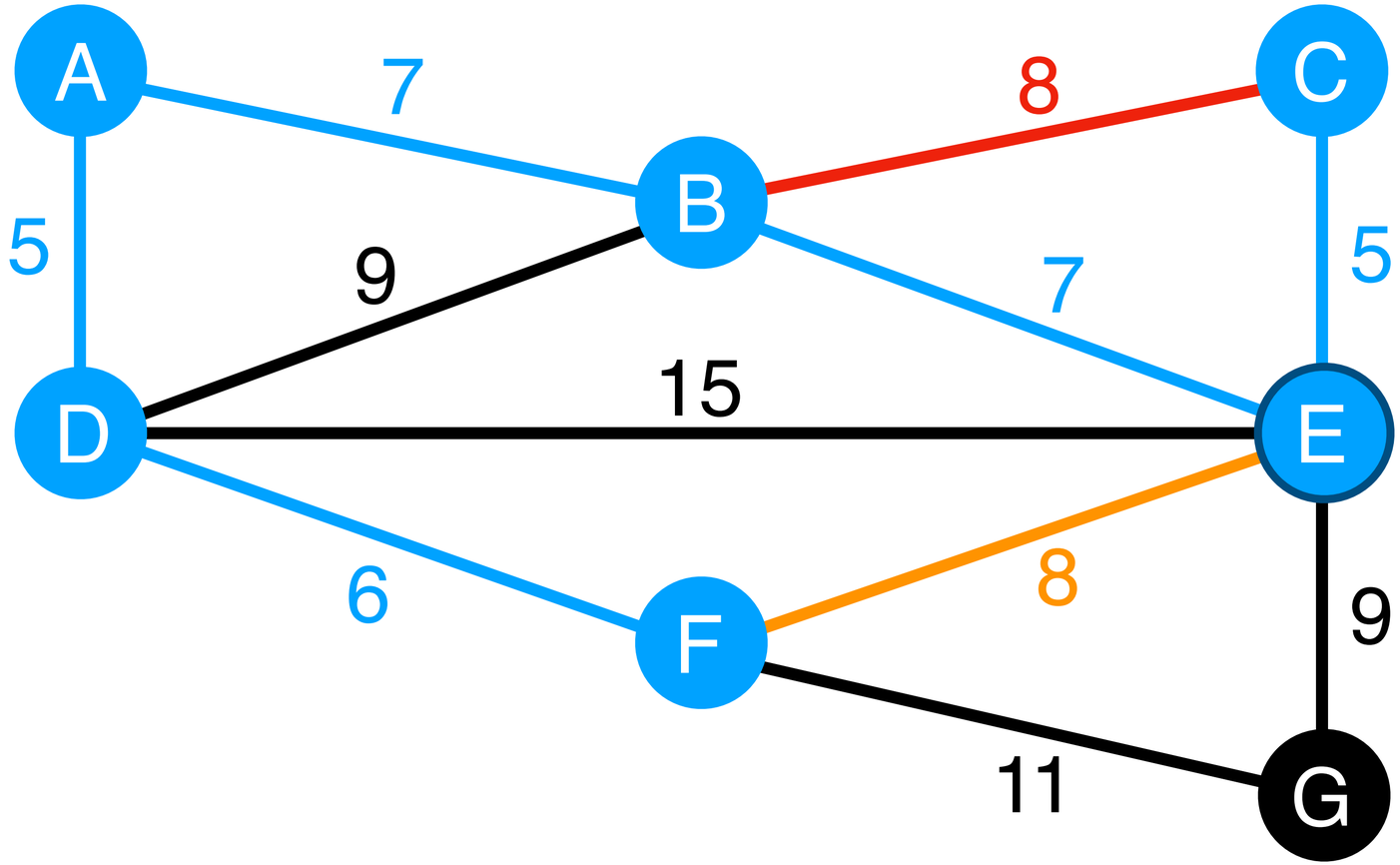
Kruskal's algorithm example: Add the lowest-weight link



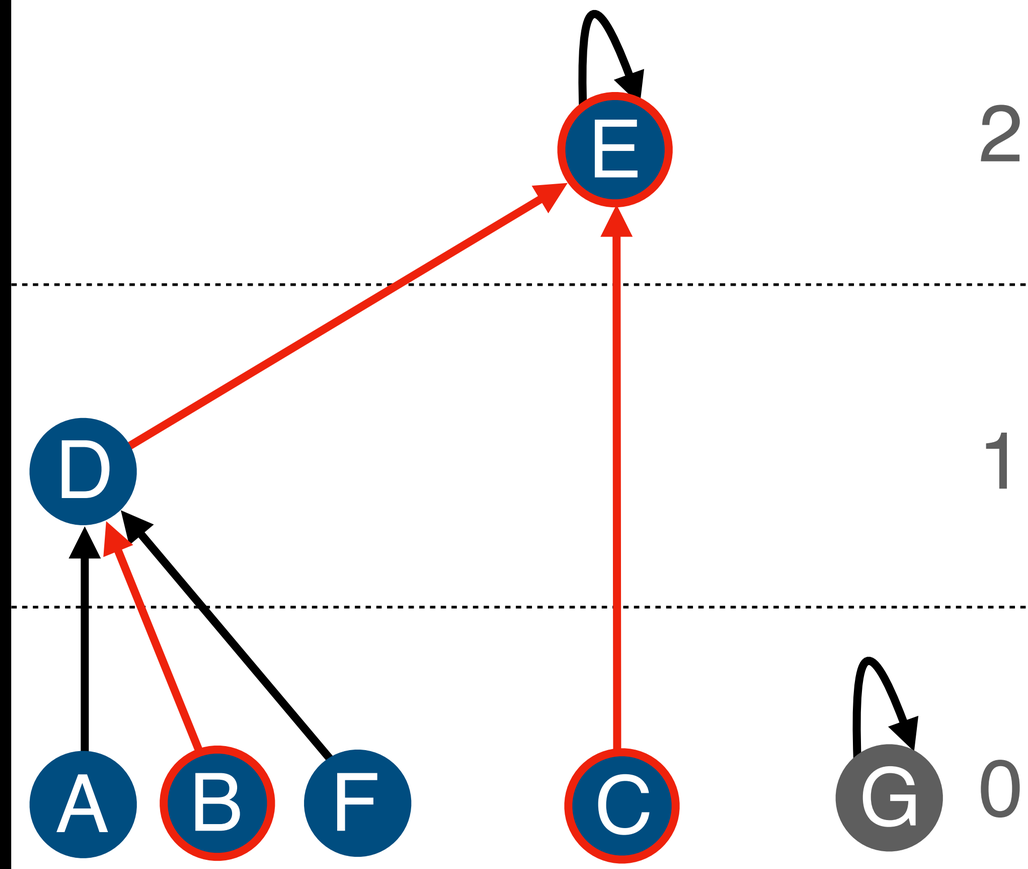
Union-find



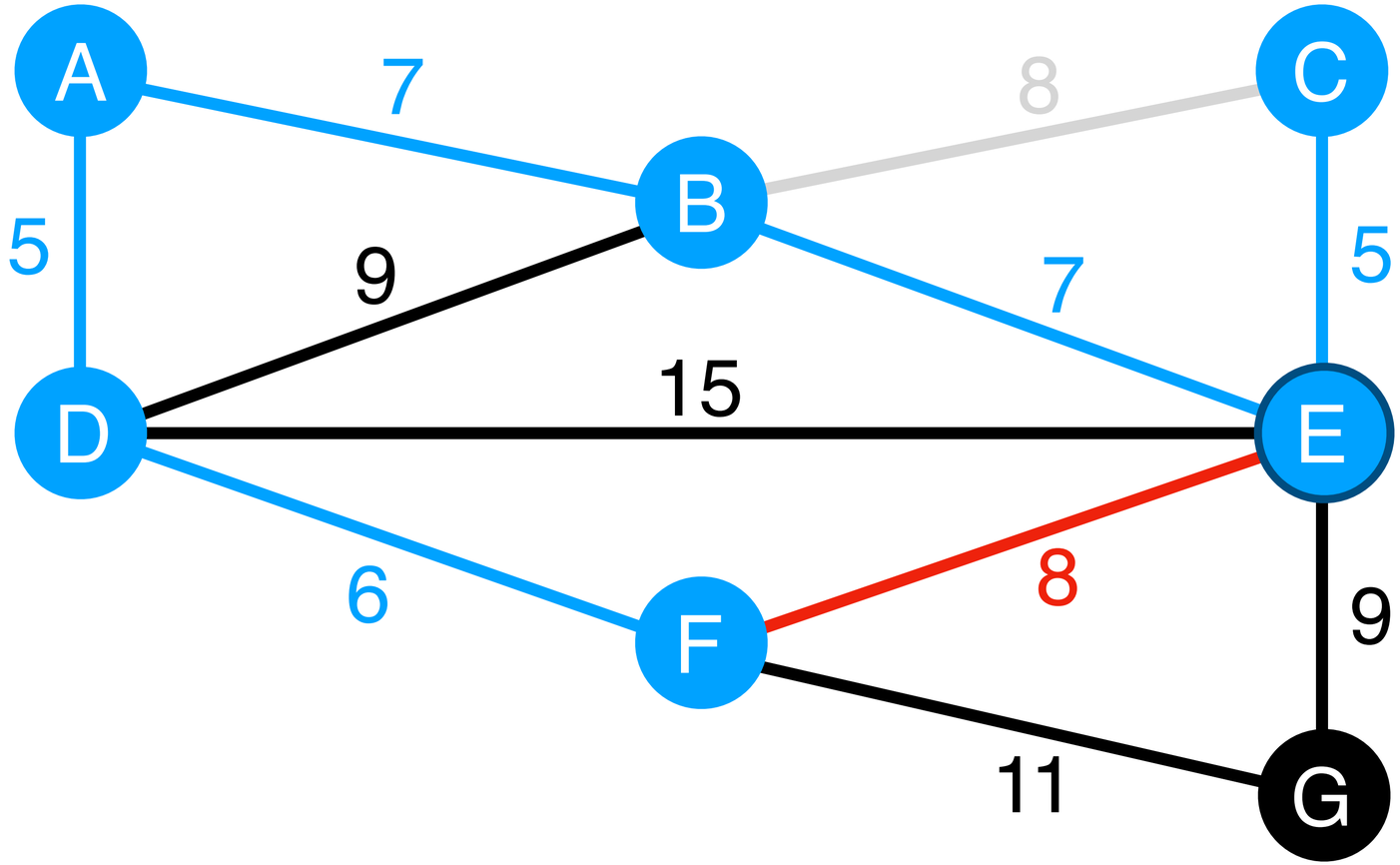
Kruskal's algorithm example: Cannot add link



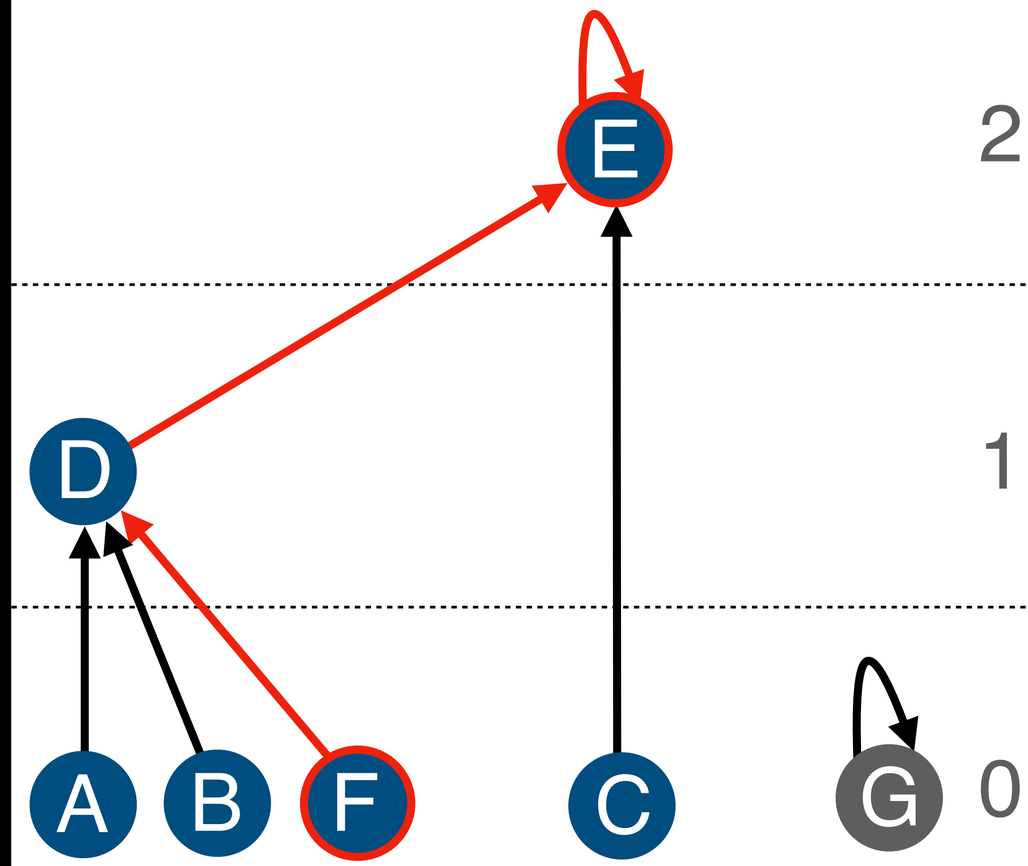
Union-find



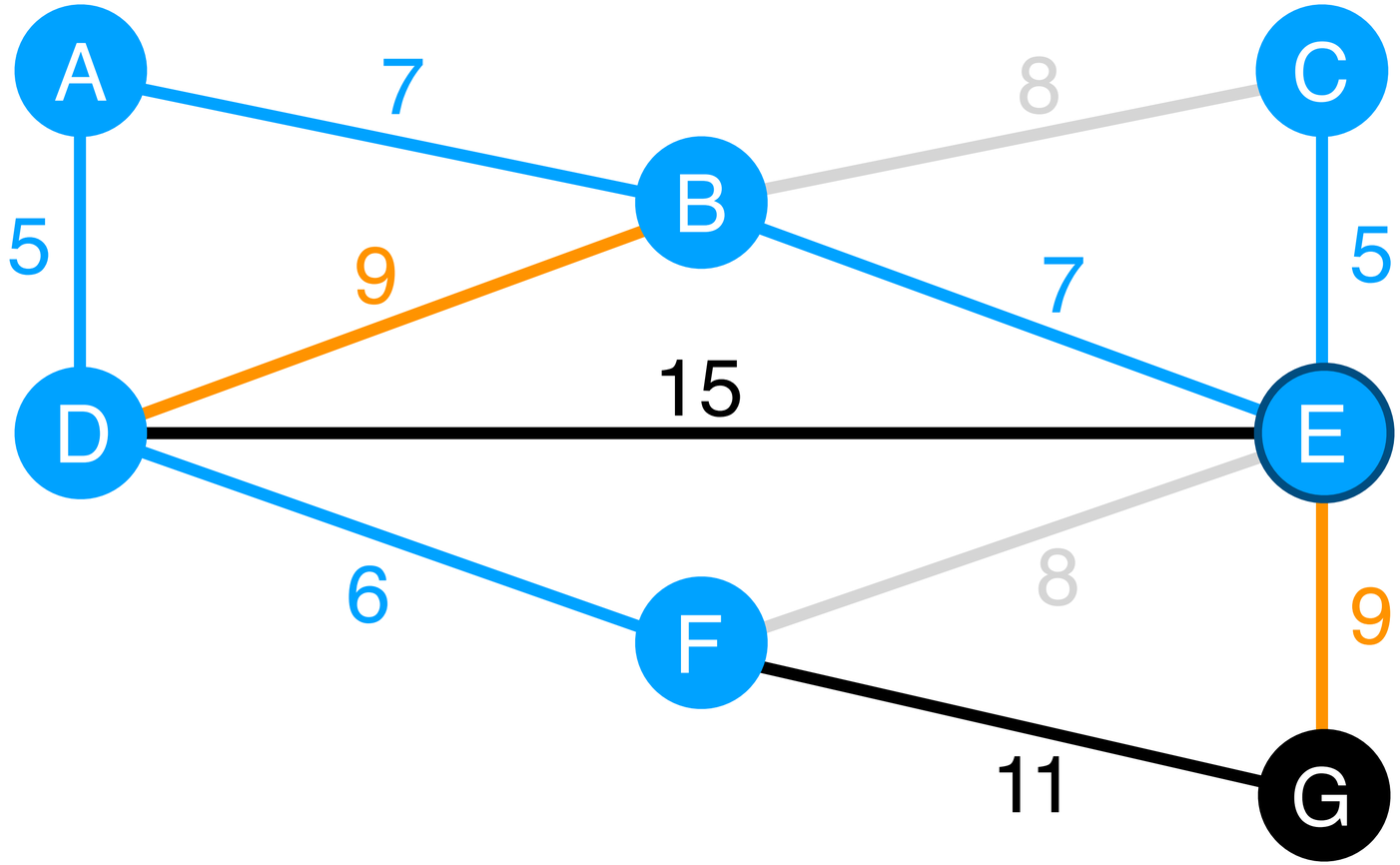
Kruskal's algorithm example: Cannot add link



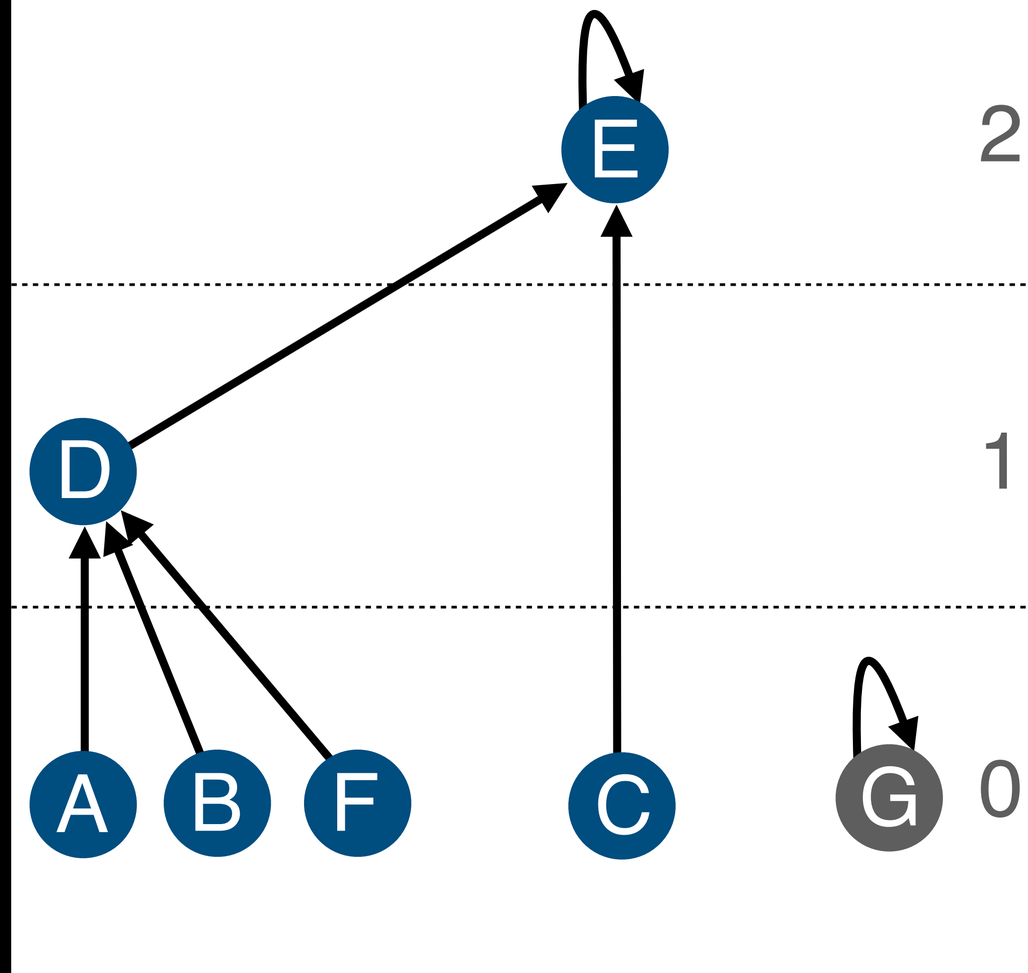
Union-find



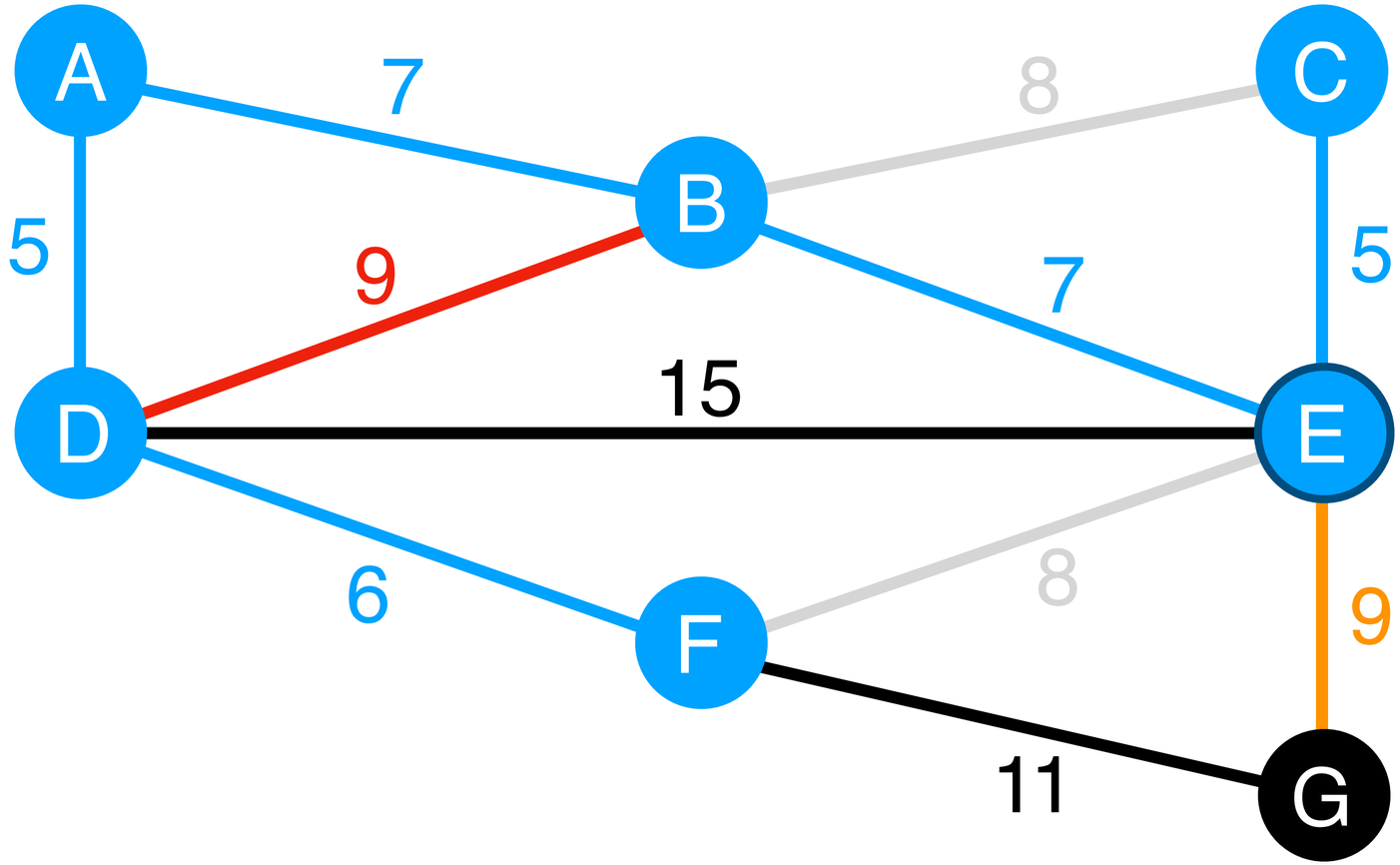
Kruskal's algorithm example: Add the lowest-weight link



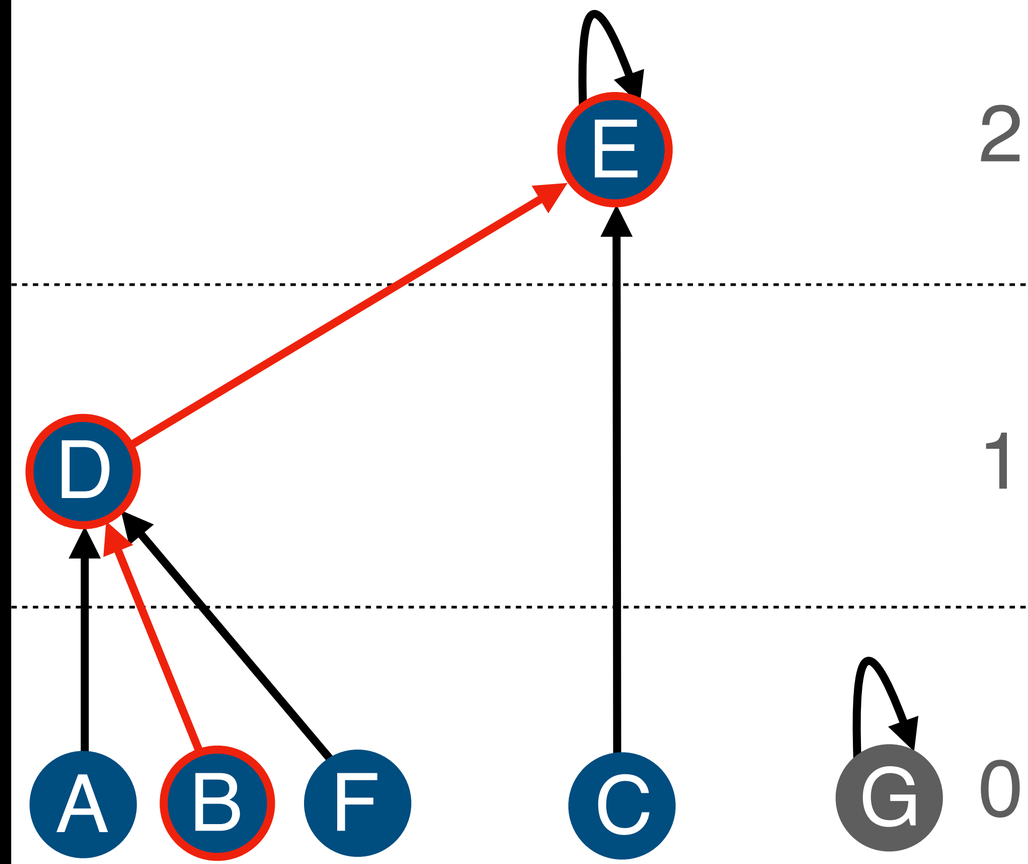
Union-find



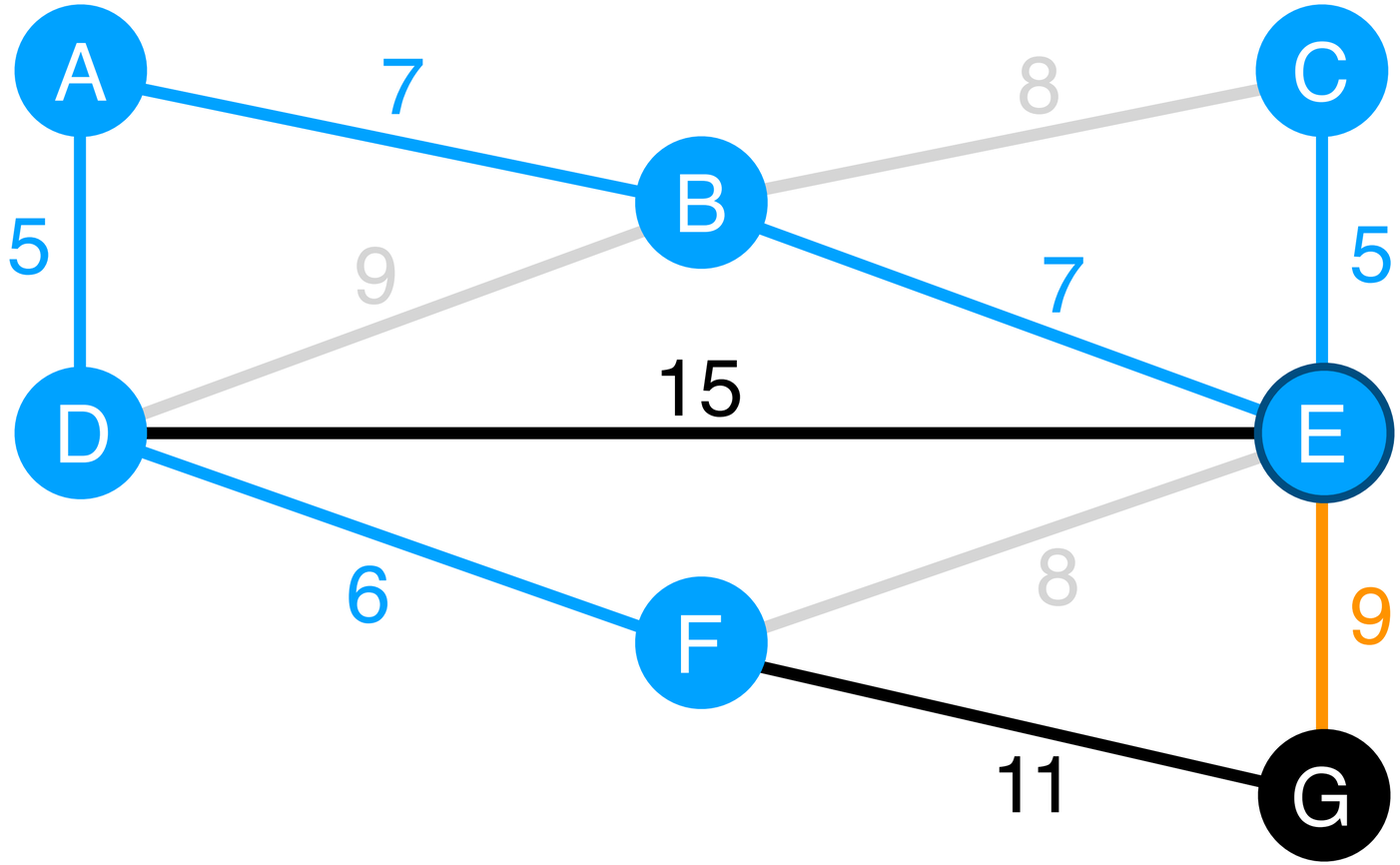
Kruskal's algorithm example: Cannot add link



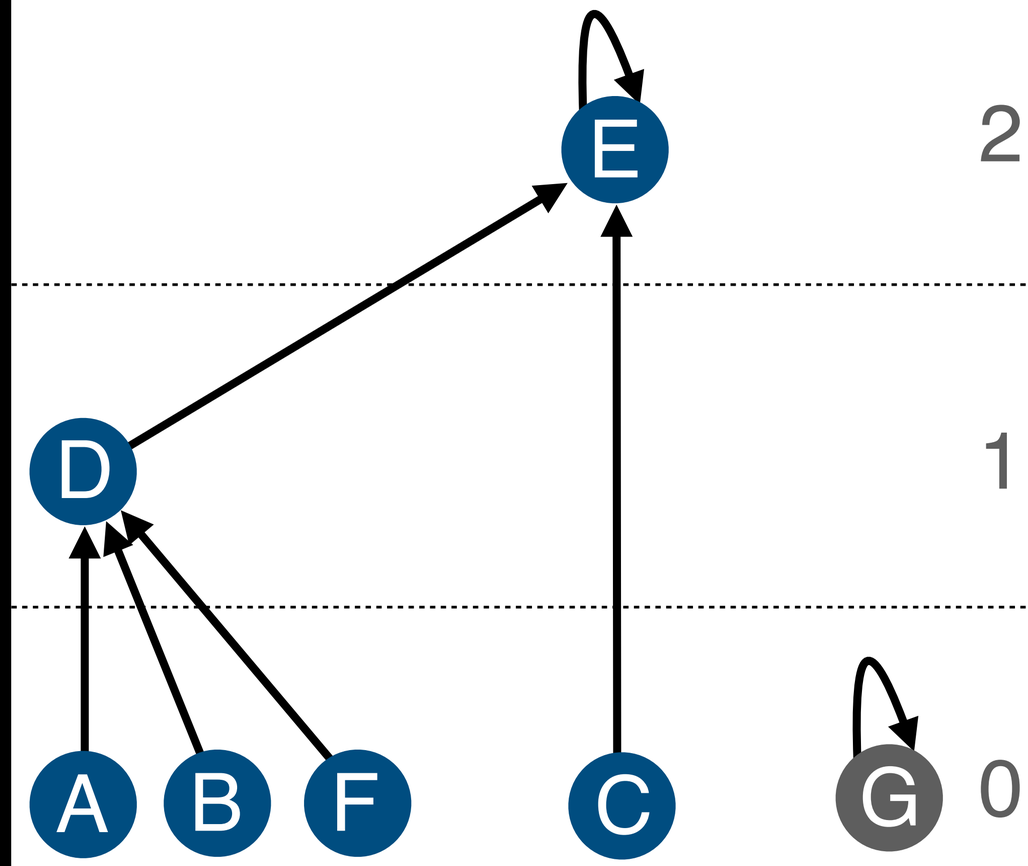
Union-find



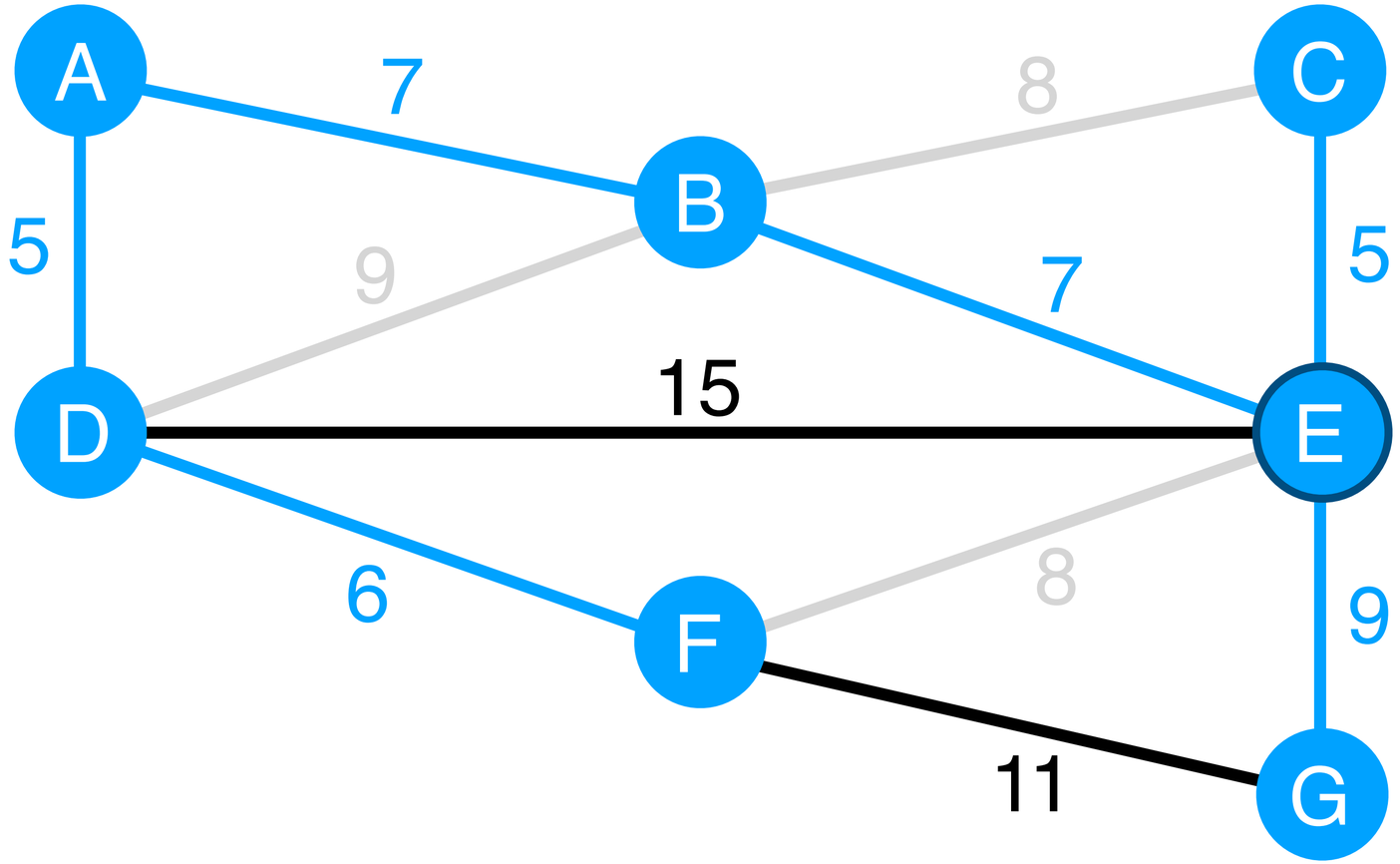
Kruskal's algorithm example: Add the lowest-weight link



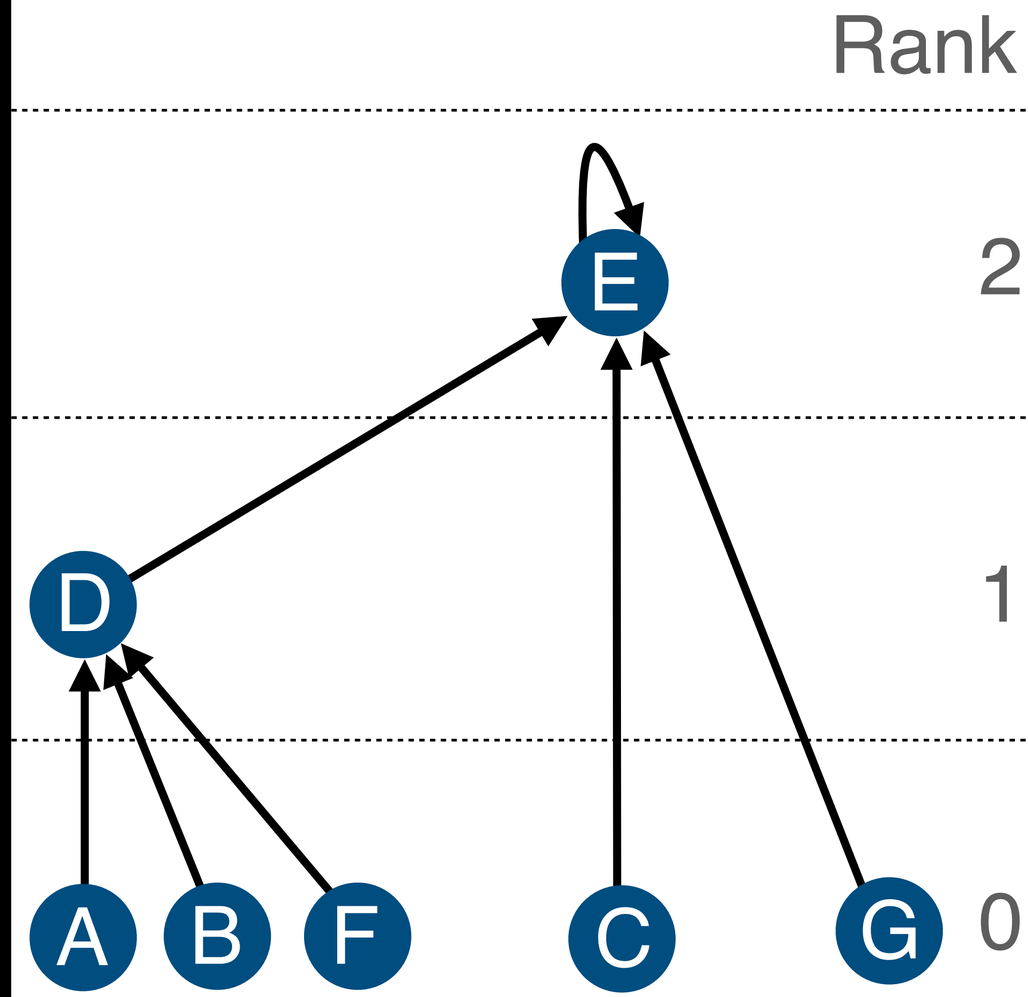
Union-find



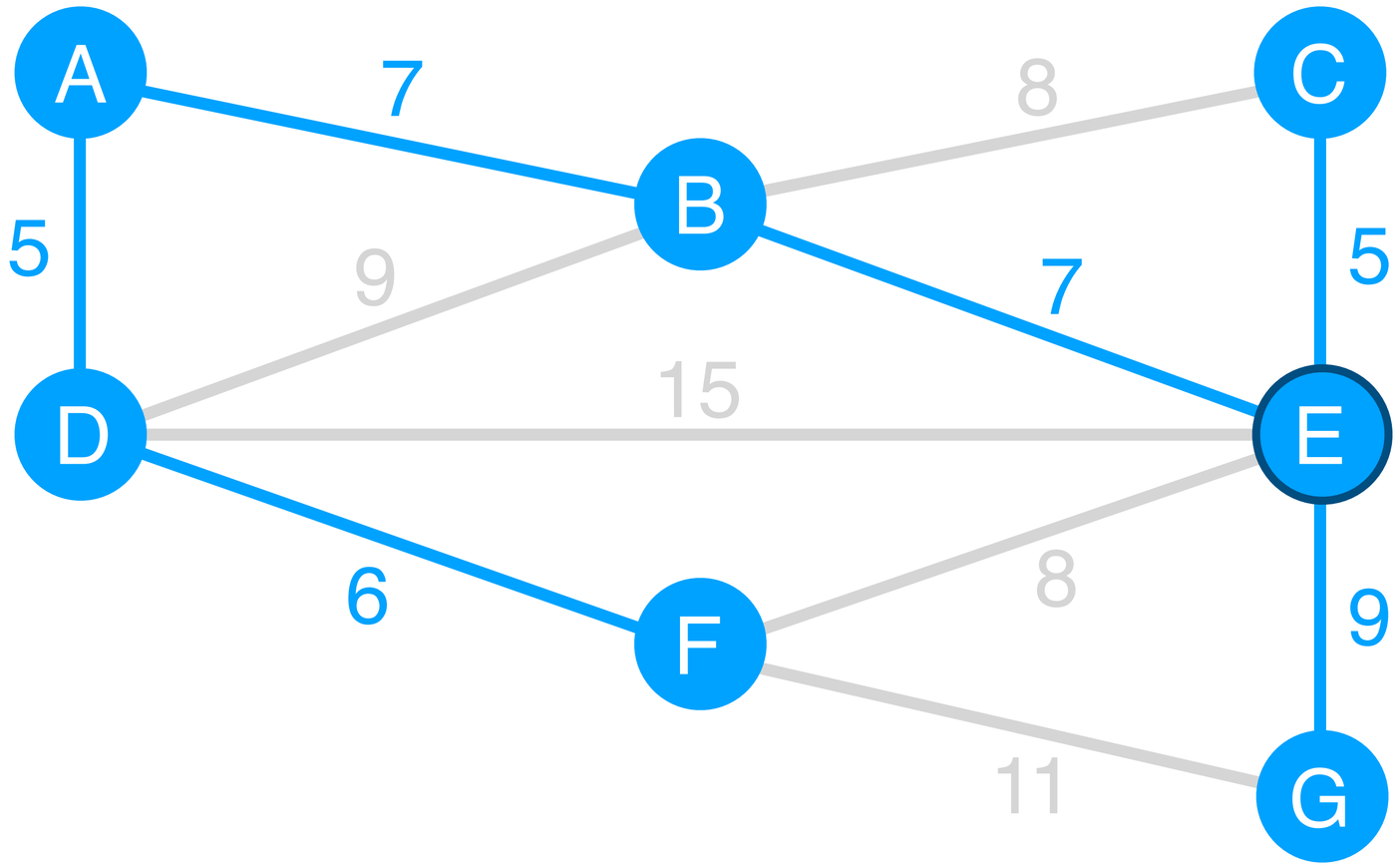
Kruskal's algorithm example: Add the lowest-weight link



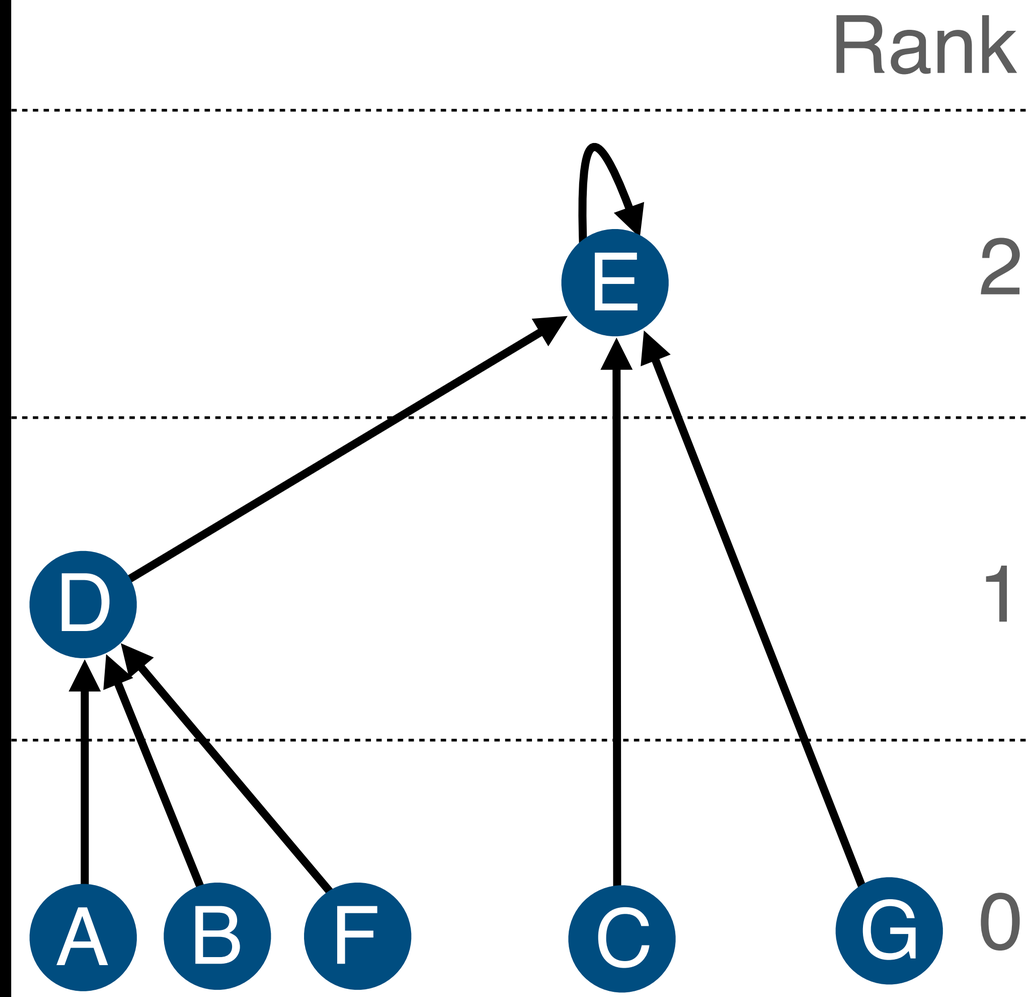
Union-find



Kruskal's algorithm example: All nodes covered - Finished!



Union-find



Why are there two algorithms for the MST?

There are actually many more.

They have Pros and Cons:

Fast for dense graphs

Works for disconnected graphs

Only works for connected graphs

Needs a special data structure (union-find) to work fast

Prim

Kruskal

The MST is a spanning tree with lowest sum of weights

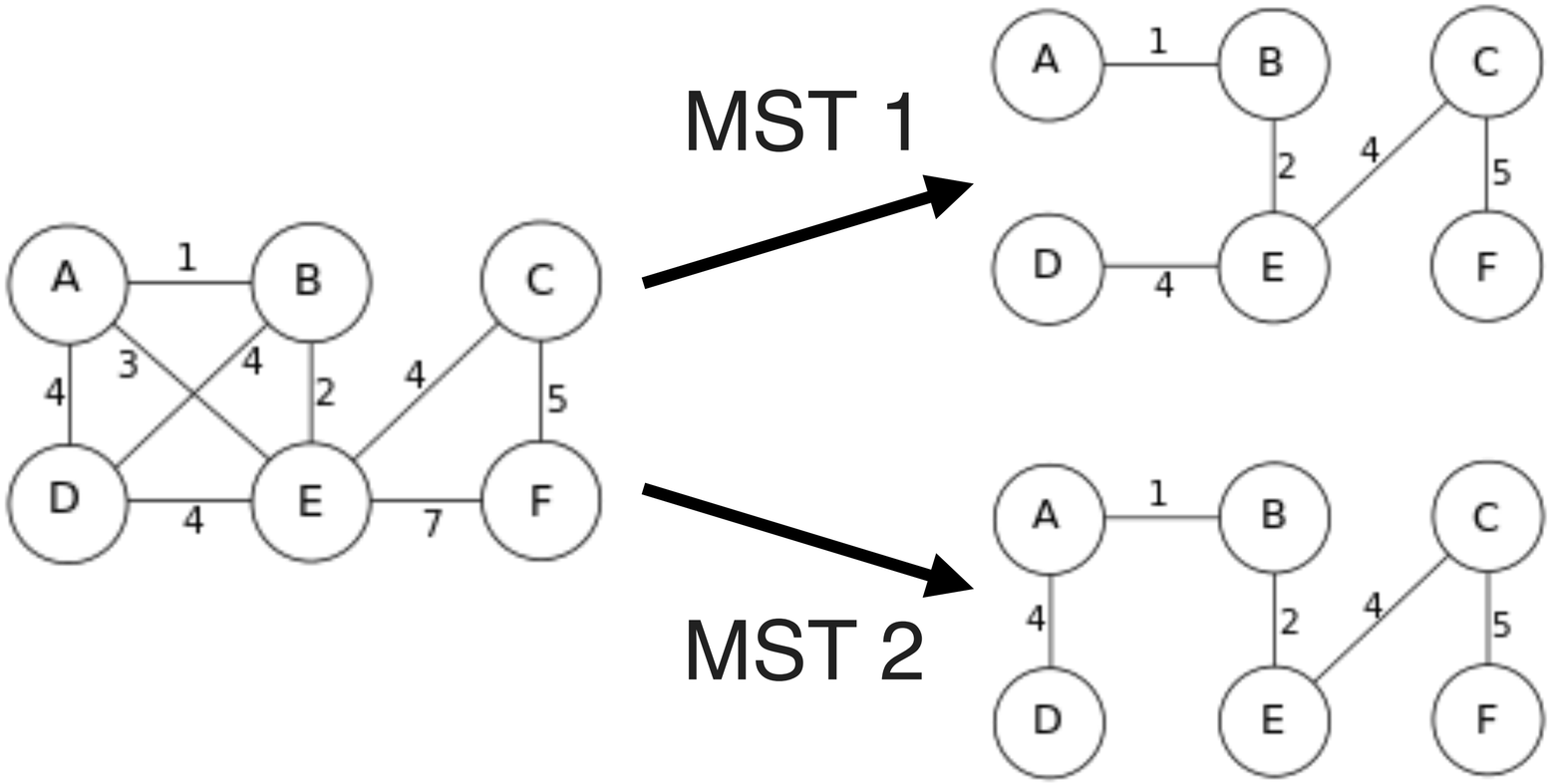
In general, is the MST unique for a given graph?

https://www.mentimeter.com/s/b41540d000f6a33c29e74720863543ba/9e0e0845bee6

The MST is a spanning tree with lowest sum of weights

In general, is the MST unique for a given graph?

No.



The MST is a spanning tree with lowest sum of weights

When can we be sure that the MST is unique?

The MST is a spanning tree with lowest sum of weights

When can we be sure that the MST is unique?

If all weights are different, we can be sure that the MST is unique.

A greedy algorithm makes a locally optimal choice

Prim and Kruskal are greedy.

A greedy algorithm makes a locally optimal choice

Prim and Kruskal are greedy.

Often greedy algorithms are globally not optimal.

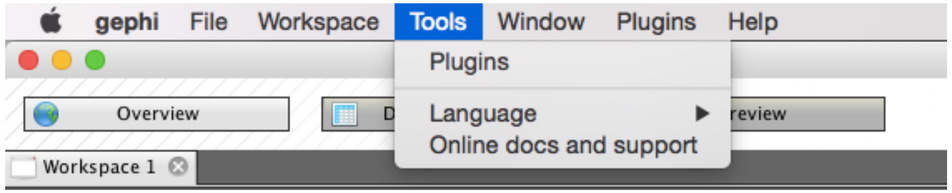
But Prim and Kruskal are globally optimal:
They actually find the MST.

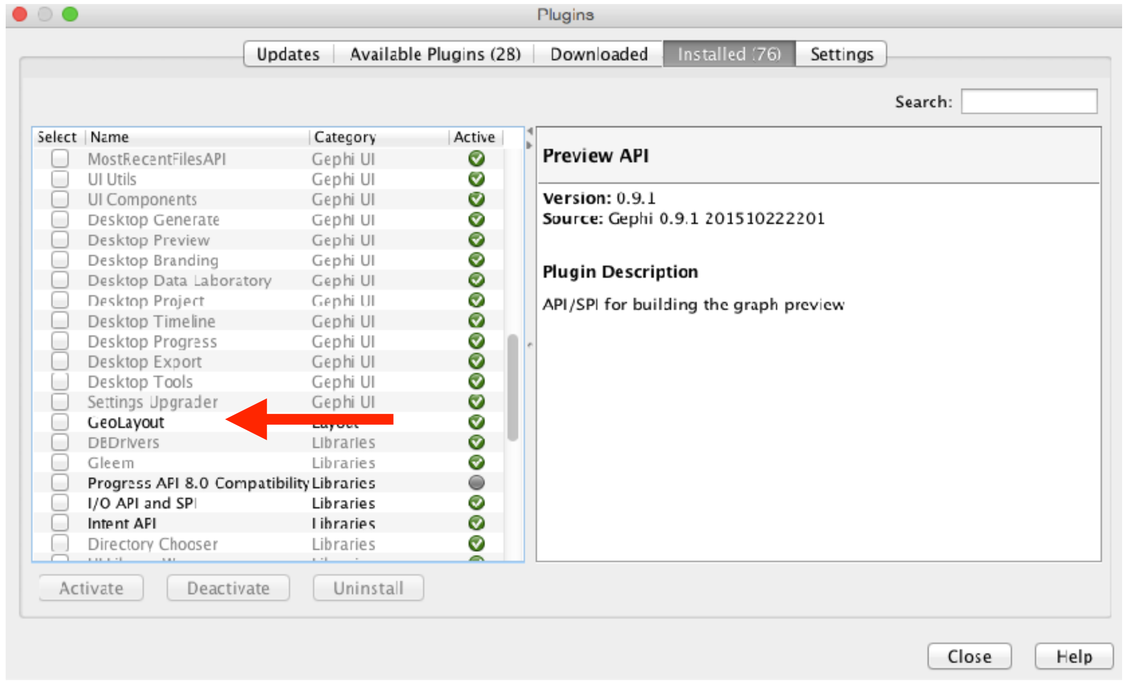
Jupyter

Gephi

Class inspired by the tutorial of Martin Grandjean
http://www.martingrandjean.ch/gephi-introduction/

Geographical Layout

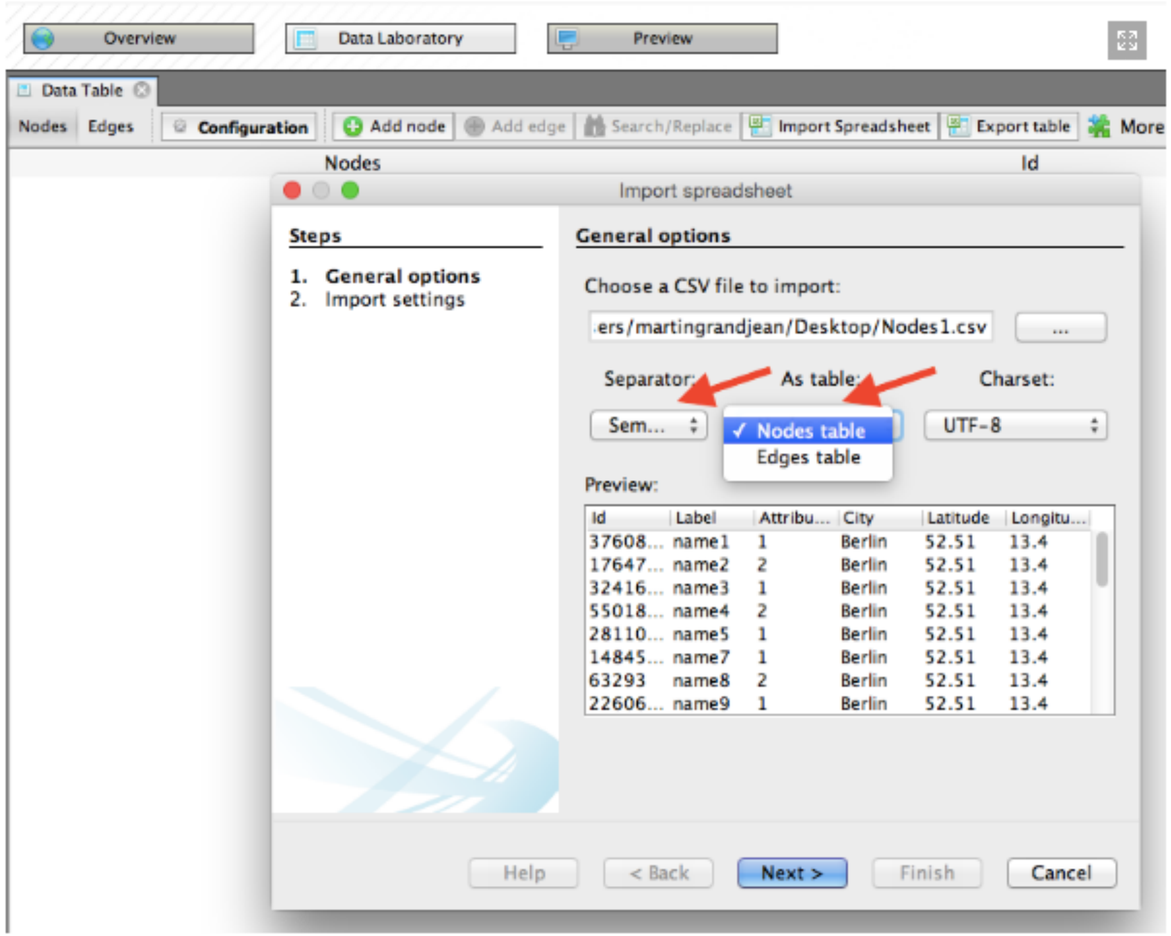




First, you will have to install the Geolayout plugin. Go to tools, then Plugins. Install the GeoLayout plugin.

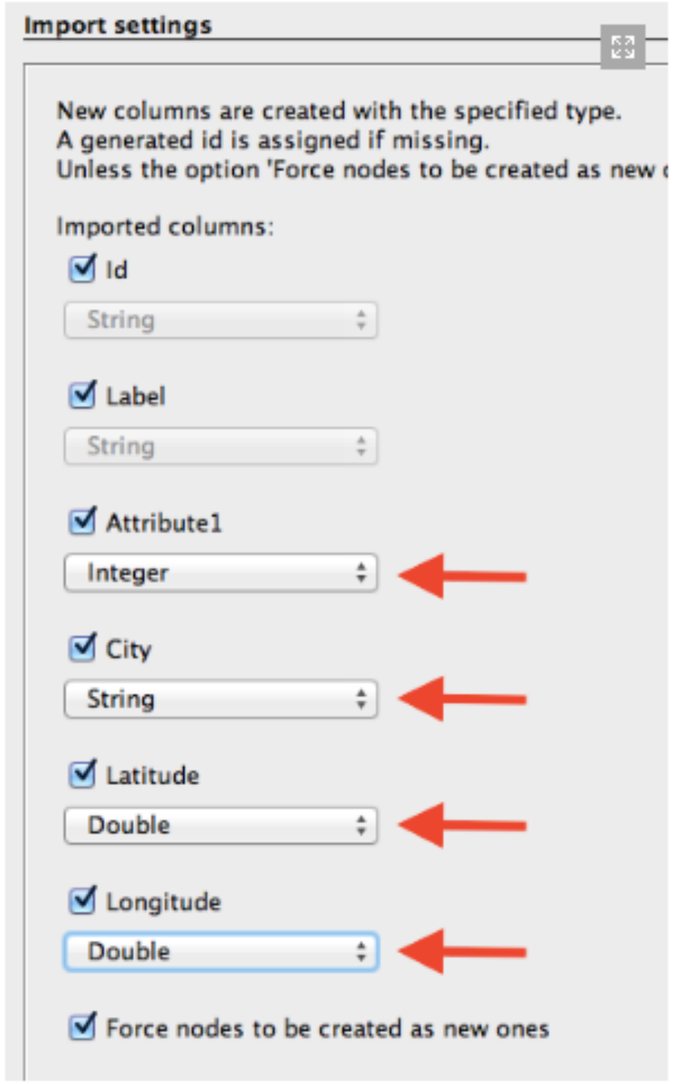
Import a csv node file

`File: dataset1/nodes.csv`



Specify that the separation between your columns is expressed by a semicolon and do not forget to inform Gephi that the file you import is containing nodes. Then press “next” and fill the import settings form as proposed (next slide).

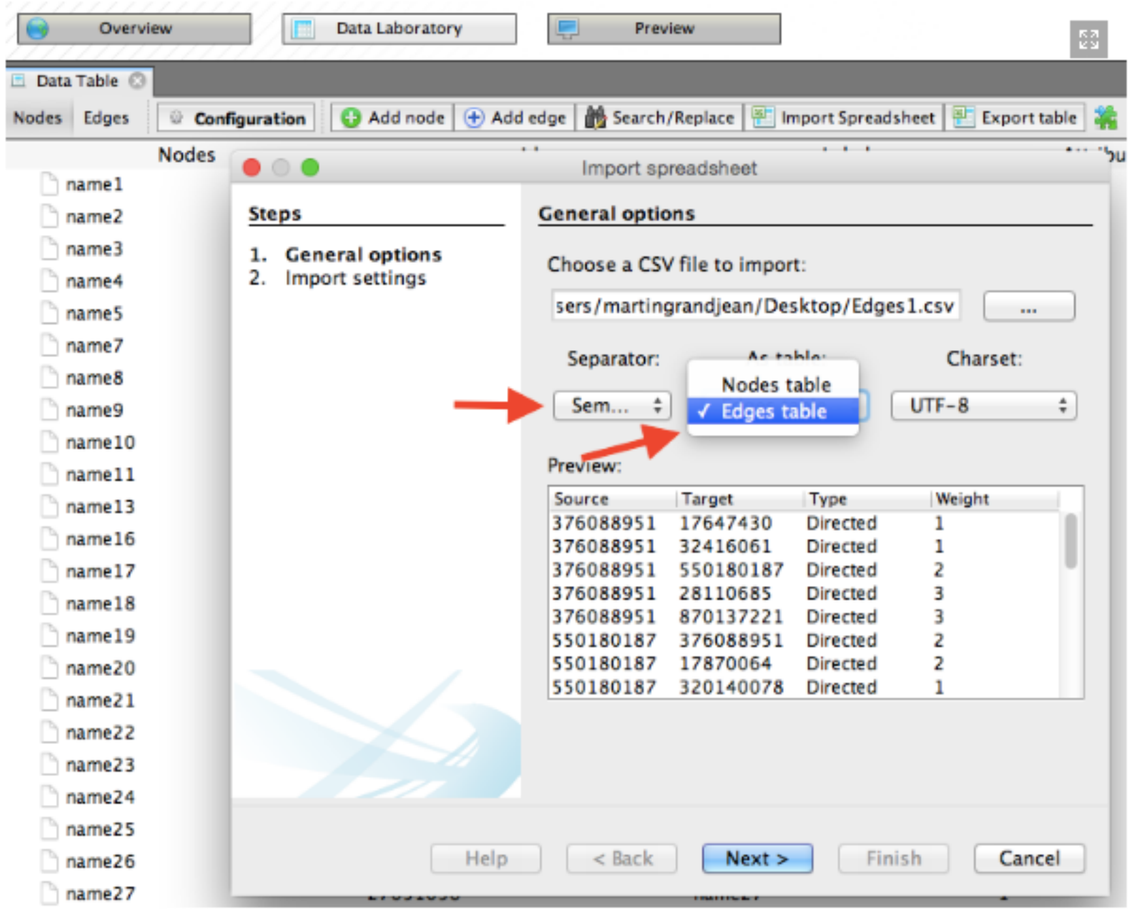
Import a csv node file



The “import settings” step is very important: Gephi will recognize some of the columns because of their header, but you’ll always have to check that the software will be able to understand the nature of your data. In our example, be sure to inform Gephi that our latitudes and longitudes are a “double” variable (not an “integer”)

Import a csv edge file

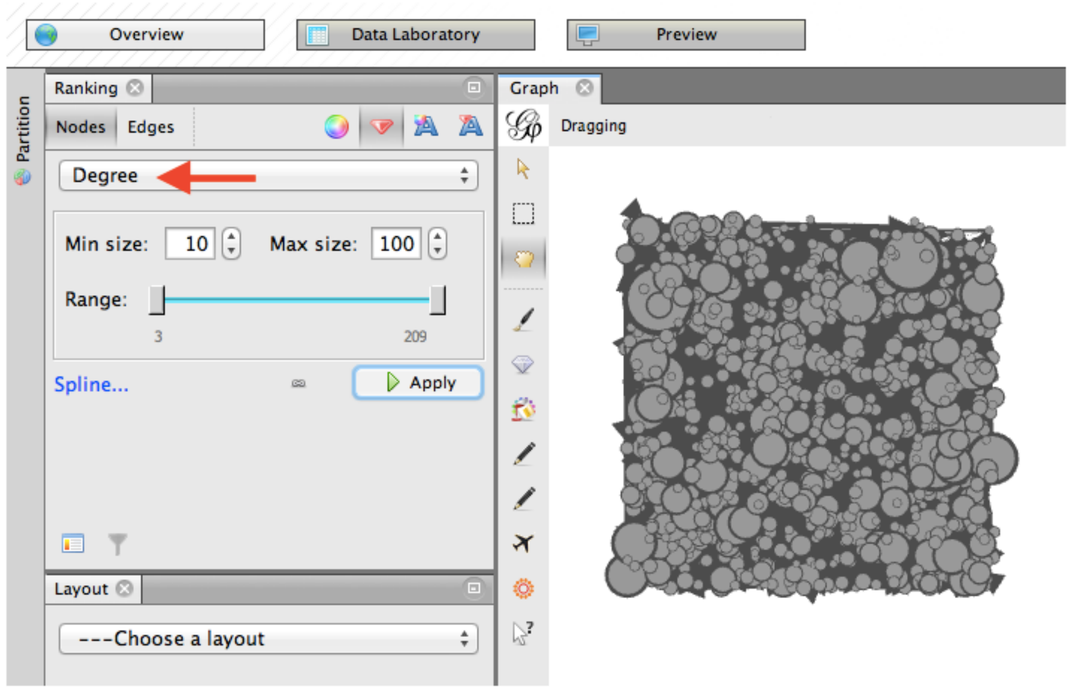
`File: dataset1/edges.csv`



Follow the same procedure, but with the “edges” file downloaded before and fill the forms in the following manner: specify the semicolon and inform Gephi that you’re importing the edges. Fill in the last fields and uncheck “create missing nodes”, because you’ve already imported them.

The action now takes place on the Overview panel. The software produces an overview of the graph, spatialized randomly and completely unreadable.

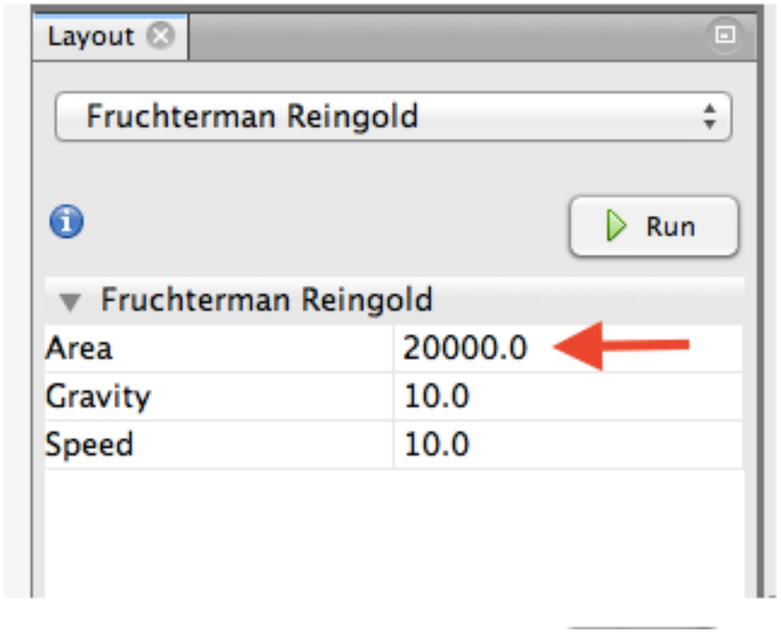
Node size



Note: your menu will look slightly different, as Gephi has modified it in the last release. But you can proceed anyway!

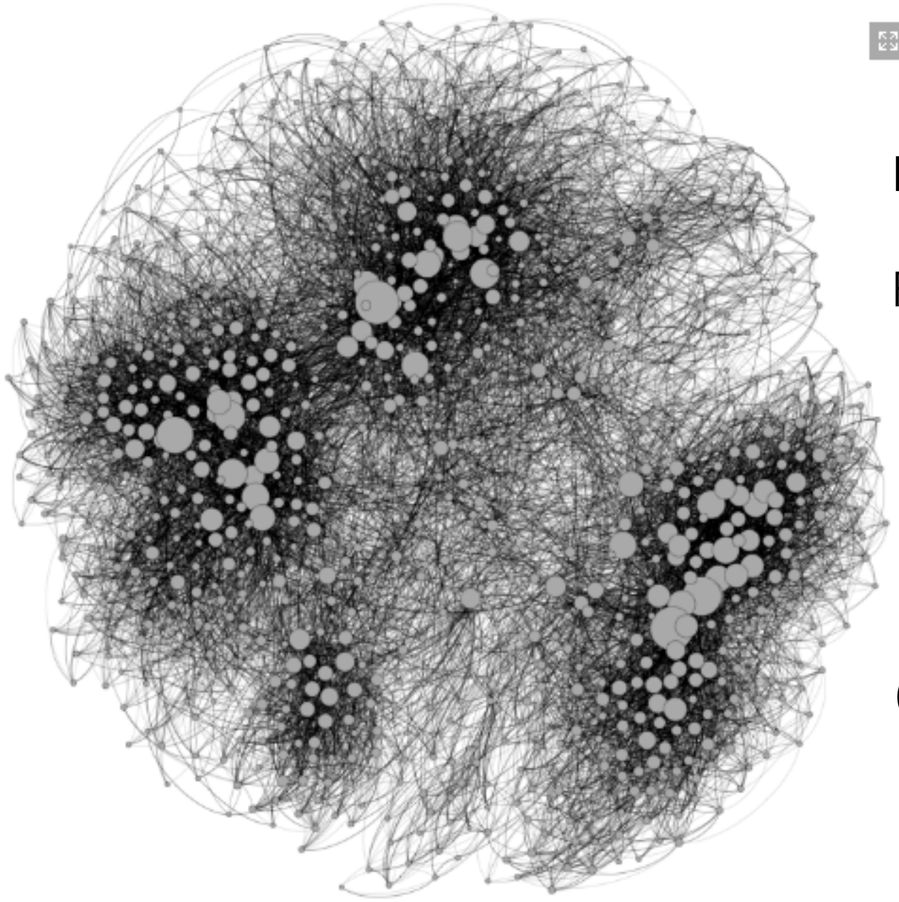
Let's give nodes a size proportional to their degree (number of connexions). In the Ranking panel of the left column (top), select "Nodes" and the size menu (red diamond in the example, different in your version), then select "Degree" in the rolling menu and enter the minimal and maximal value (we propose 10-100).

Layout



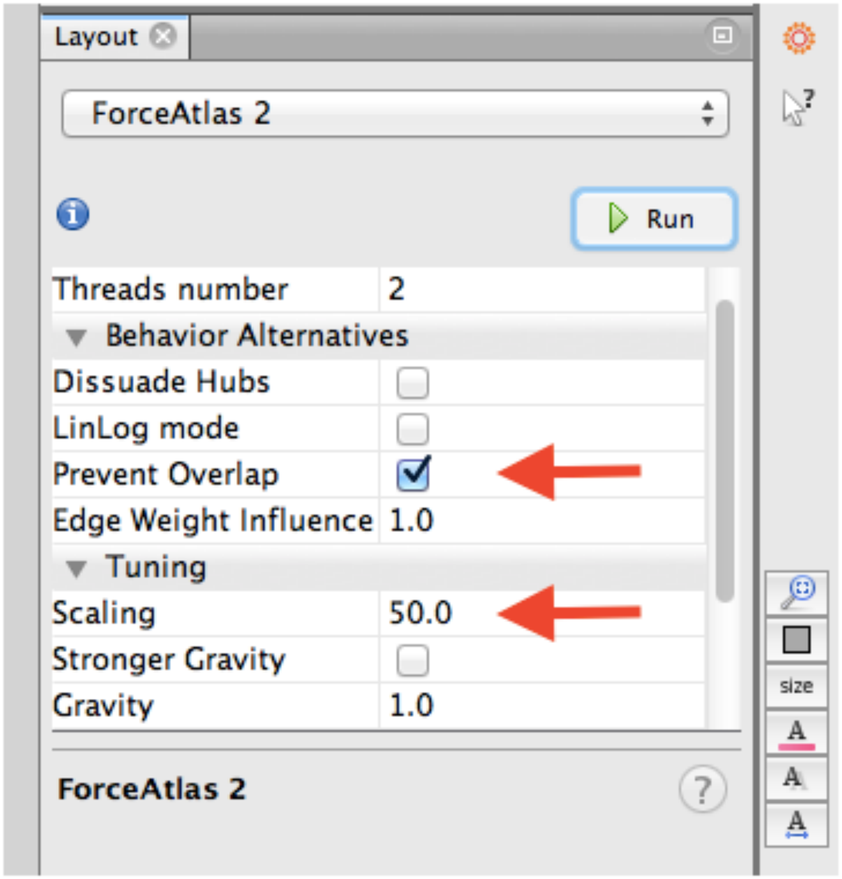
Let's begin with a spatialization that gives more space to the graph, but maintain it in a decided area: Fruchterman Reingold, with the same values as in this model (20.000 – 10 – 10). This visualization disposes nodes in a gravitational way (attraction-repulsion, in fact, as magnets). You're already able to distinguish communities (more densely connected parts of the network). Let the function run until the graph is stabilized. Use the little blue magnifying glass (bottom left of the graph panel) to re-center the zoom.

Layout



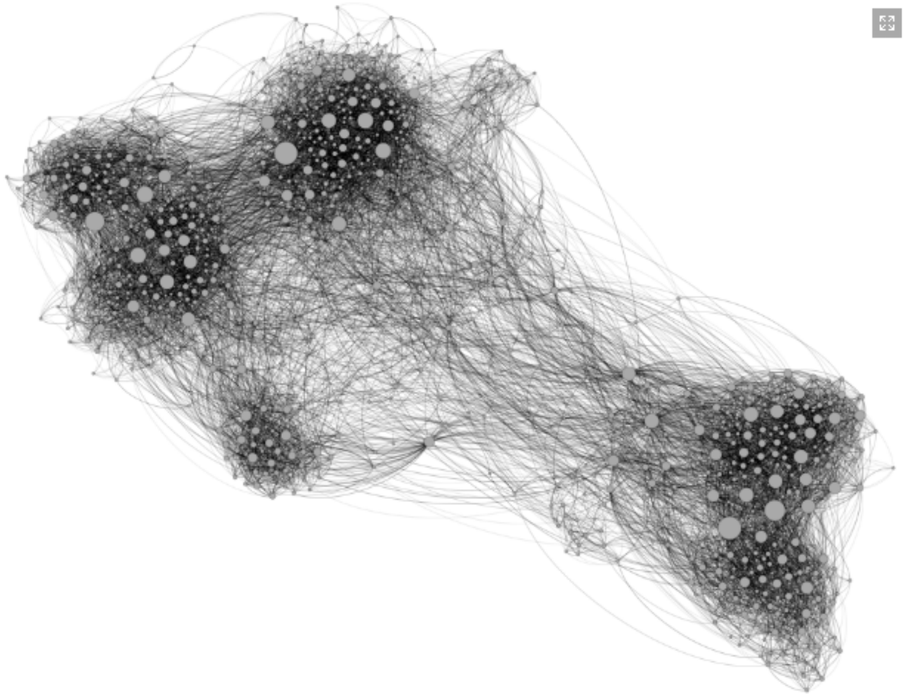
Let’s begin with a spatialization that gives more space to the graph, but maintain it in a decided area: Fruchterman Reingold, with the same values as in this model (20.000 – 10 – 10). This visualization disposes nodes in a gravitational way (attraction-repulsion, in fact, as magnets). You’re already able to distinguish communities (more densely connected parts of the network). Let the function run until the graph is stabilized. Use the little blue magnifying glass (bottom left of the graph panel) to re-center the zoom.

Layout



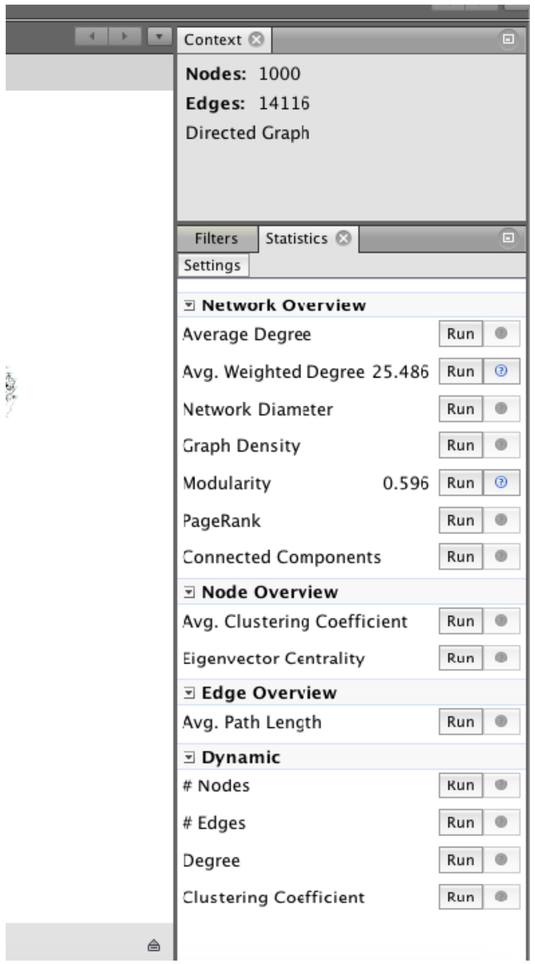
Then, we propose to use the Force Atlas 2 (a bit different from the layout we saw last time, Force Atlas). This layout disperses groups and gives space around larger nodes. Be careful, the parameters you enter significantly alter the final appearance (suggestion: Check “prevent overlap” and change “Scaling” to 50). Let the function run until the graph is mostly stabilized. We can apply Force Atlas 2 directly without applying Fruchterman Reingold before, but as the “random layout” from the beginning is a … random layout, it’s better to untangle the network before submitting it to a strong force-algorithm.

Layout

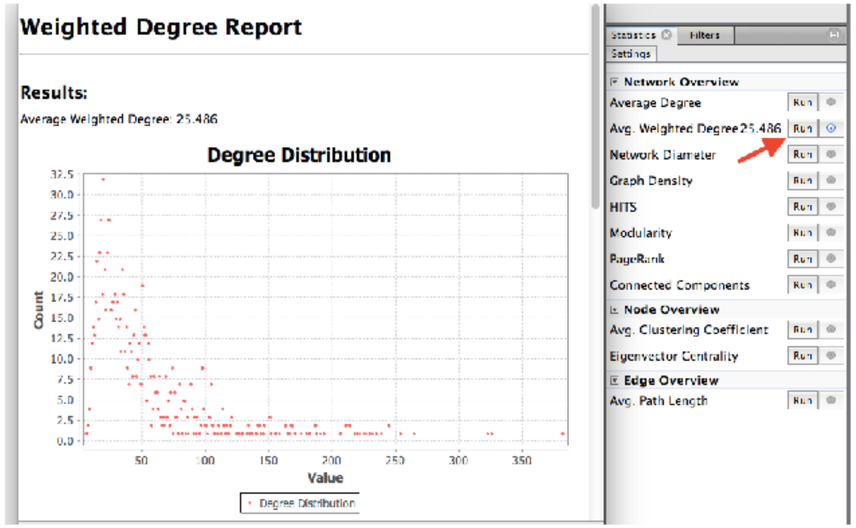


Then, we propose to use the Force Atlas 2 (a bit different from the layout we saw last time, Force Atlas). This layout disperses groups and gives space around larger nodes. Be careful, the parameters you enter significantly alter the final appearance (suggestion: Check “prevent overlap” and change “Scaling” to 50). Let the function run until the graph is mostly stabilized. We can apply Force Atlas 2 directly without applying Fruchterman Reingold before, but as the “random layout” from the beginning is a … random layout, it’s better to untangle the network before submitting it to a strong force-algorithm.

Statistics panel

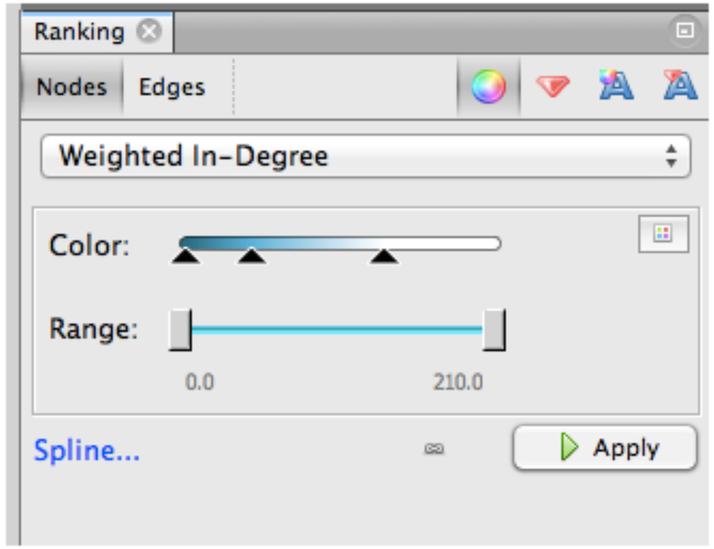


Statistics: Weighted degree



Let's add some more information to our graph by giving the nodes new attributes, influencing their color. In the Data laboratory, select the Edges Table, and sort them according to their weight. Some edges have a weight of 3, some 2 and some 1. That means that we have to take these differences into account by calculating the weighted degree of the nodes. You also observe that this graph is directed: the edges have a source and a target, a direction shown by a little arrow on the Overview display. So, the degree we'll have to calculate has to distinguish the in- and out- connexions. In the Statistics panel, click on "Average Weighted Degree" to calculate these values for every nodes. You get a report showing the distribution of theses measures.

Statistics: Weighted degree

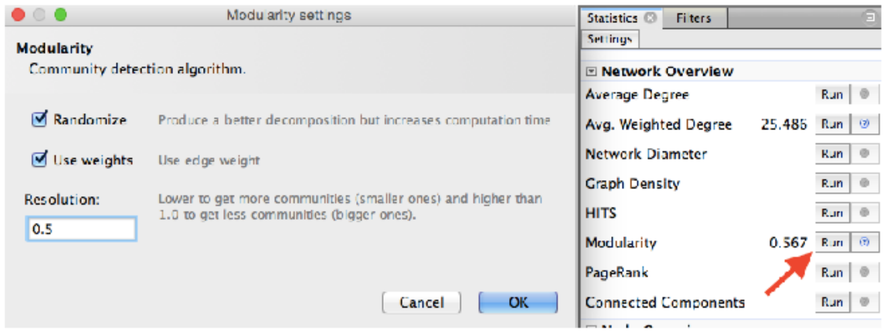


Now that theses values are calculated, new attributes are available in the ranking panel. Select the “color” icon, and chose “weighted in-degree” to color nodes according to the number of incoming edges. Little visual tip : use a dark color for small values and a light color for the highly connected nodes, in order to make the little nodes visible on the final graph (the well connected nodes are generally more visible).

Statistics: Weighted degree

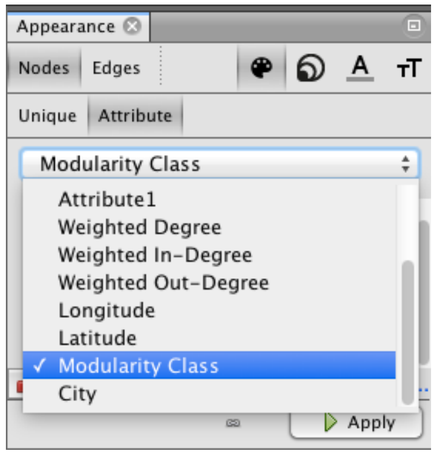
Result: the biggest nodes (=with a high degree) are not always those with the biggest weighted in-degree: if we consider an edge like a letter written between 2 people, those who are writing a lot are not necessary those who are receiving a lot. It's interesting to give different attributes to nodes size and color, to compare them. Of course, you can export this data to conduct a full statistical analysis, scatter plots, etc. (the measures you make are automatically added to your nodes table).

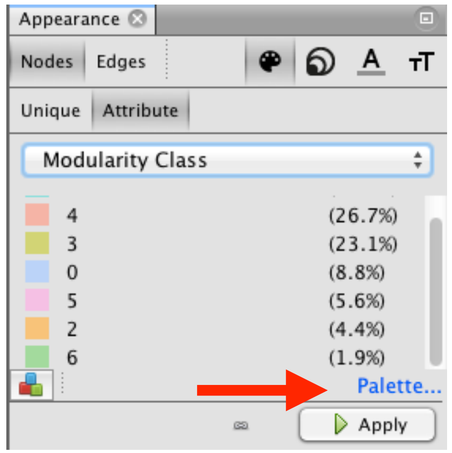
Statistics: Communities

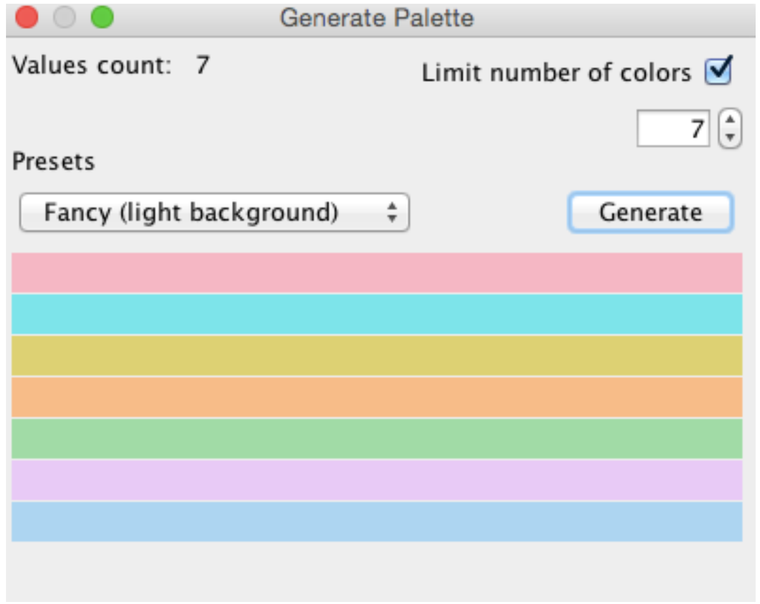


A network contains internal subdivisions called communities. There are methods that permit to highlight these communities, which depend on the comparison of the densities of edges within a group, and from the group towards the rest of the network (if interested, have a look at https://en.wikipedia.org/wiki/Community_structure) In the right column of the "overview" page, click on Statistics/Modularity/Run to display the modularity window. Choose a resolution (between 0.1 and 2), click OK and close it.

Statistics: Communities







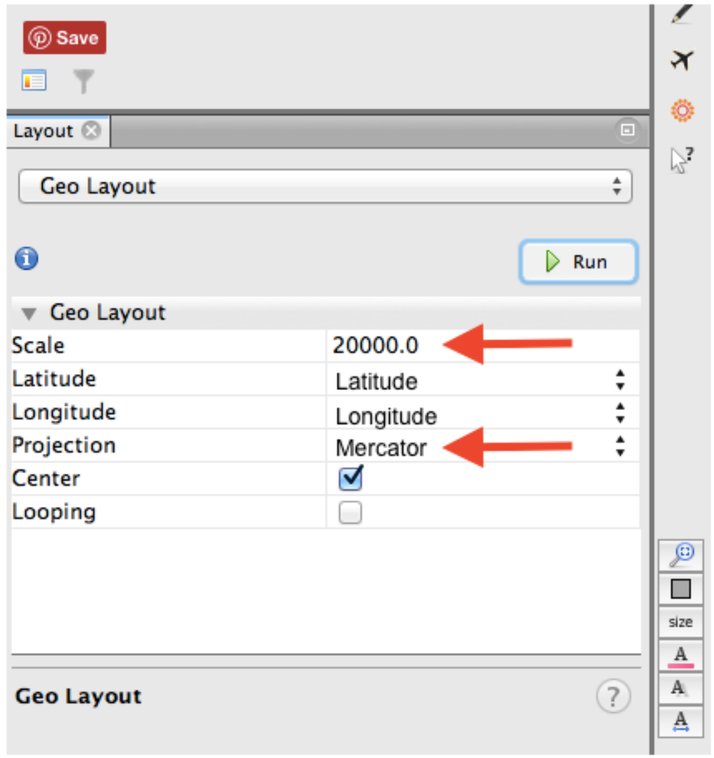
The next step takes place in the Partition menu situated in the left column. Select “Nodes” and “Modularity Class” (rolling menu). You will be then able to modify the colors attributed to the detected communities by clicking on them. You can also click on palette, and generate a new palette. I chose the fancy light background one. Do not hesitate to run the modularity statistic with many “Resolutions”! If you decide to do so, you must deselect and reselect “Modularity Class” in the left column, and refresh color calculation.

Geographical Layout

Id	Label	Interval	Attribute1	City	Latitude	Longitude
376088951	name1		1	Berlin	52.51	13.4
17647430	name2		2	Berlin	52.51	13.4
32416061	name3		1	Berlin	52.51	13.4
550180187	name4		2	Berlin	52.51	13.4

Nodes here have also two spatial attributes: a **latitude and a longitude**. You can check this in the data laboratory. **Let's use this for a layout**.

Geographical Layout



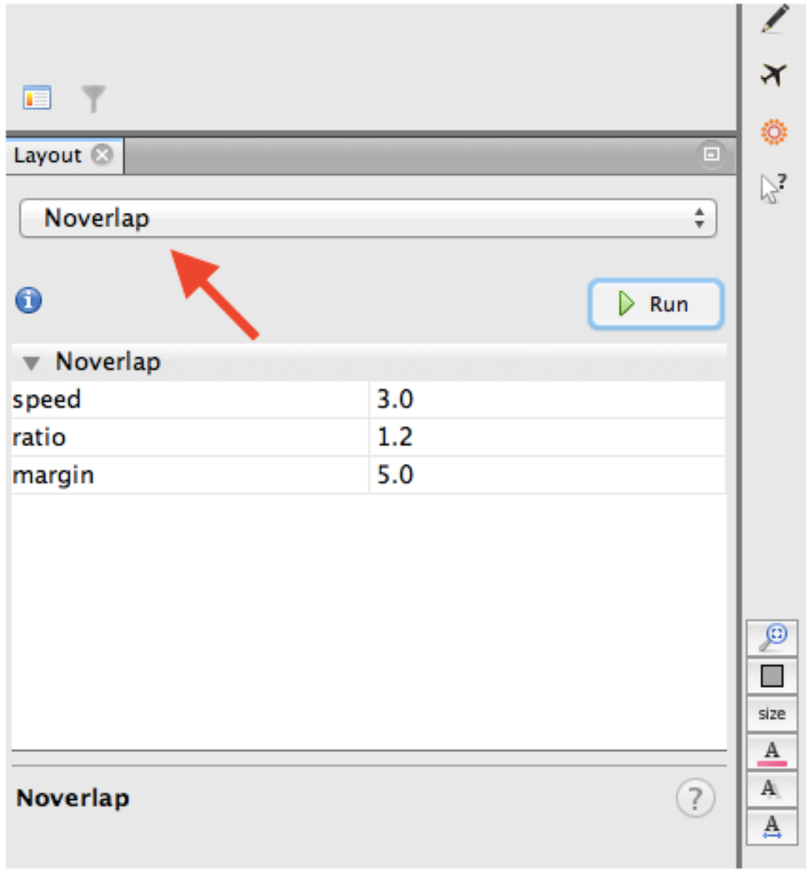
In the Layout panel, select Geo Layout and give it a scale of 20.000. Be sure that the plugin understand correctly that “Latitude” as a “Latitude” and “Longitude” as a “Longitude” and set the projection to “Mercator” (this projection should be adapted to the map you’ll use after).

Map projections from xkcd

https://xkcd.com/977/

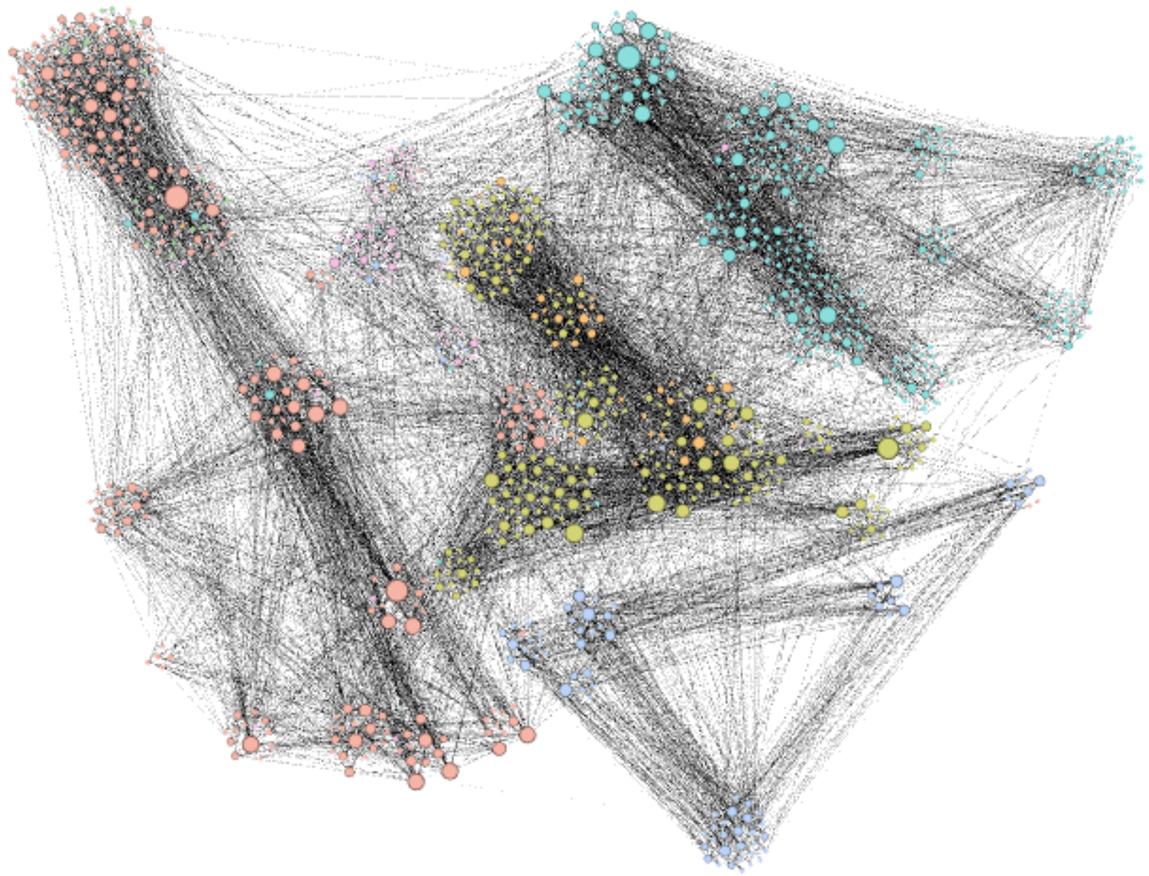
Mercator projection for maps

Geographical Layout

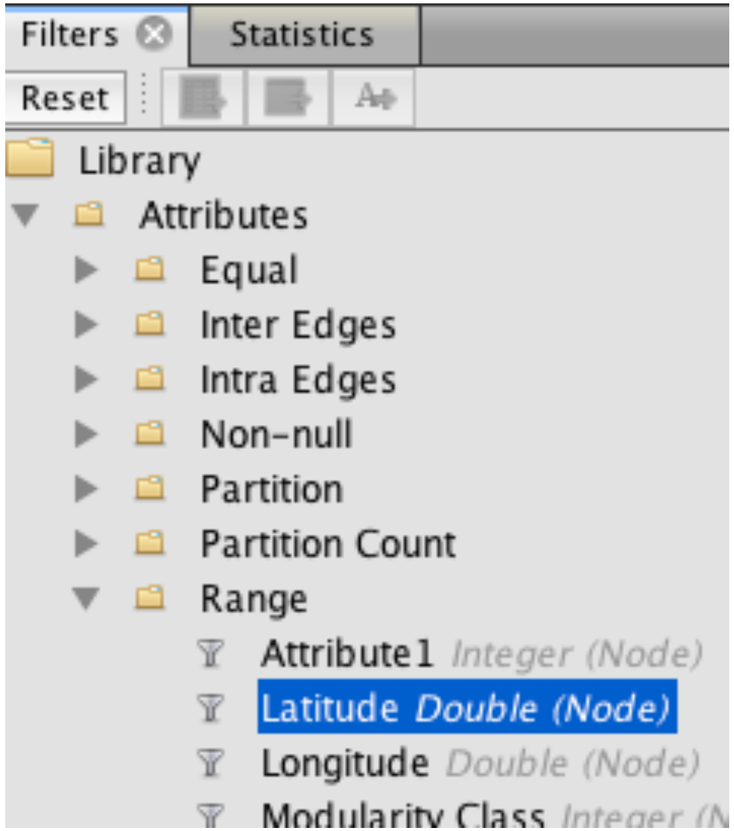


As nodes are now grouped on a geographical coordinate, you'll have to give them some space: use the Noverlap layout plugin to avoid them overlapping (a margin of 5.0 is enough with the chosen map scale).

Geographical Layout

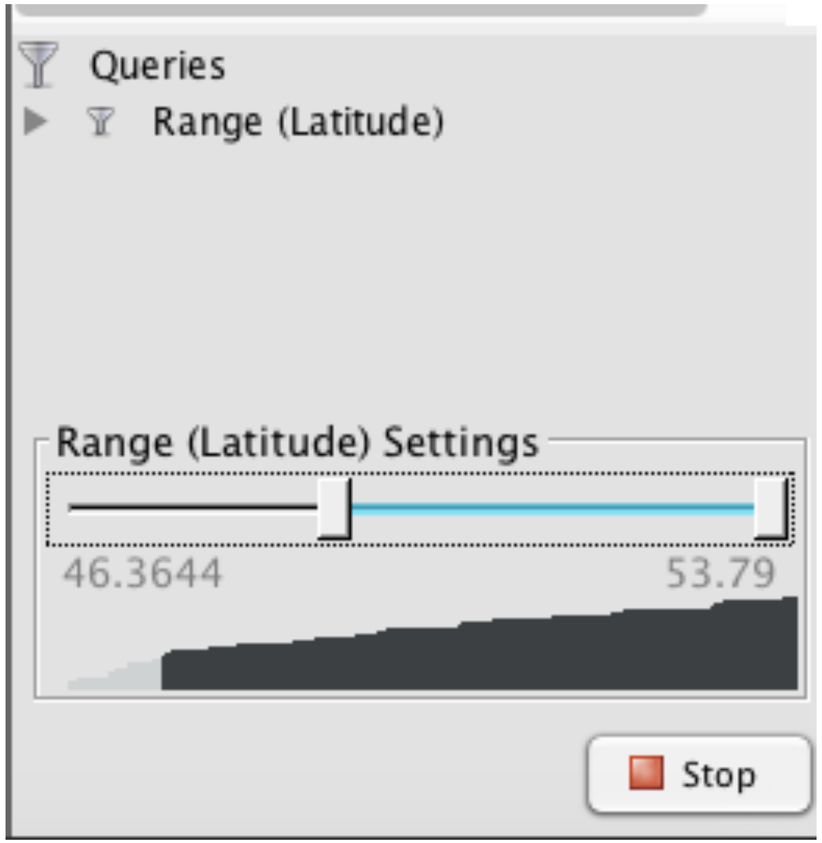


Filtering the graph



You can also filter part of the networks using the filter menu. For example, we can filter nodes based on their attributes, like Latitude. Drag the filter latitude, in Attributes -> range in the query below.

Filtering the graph

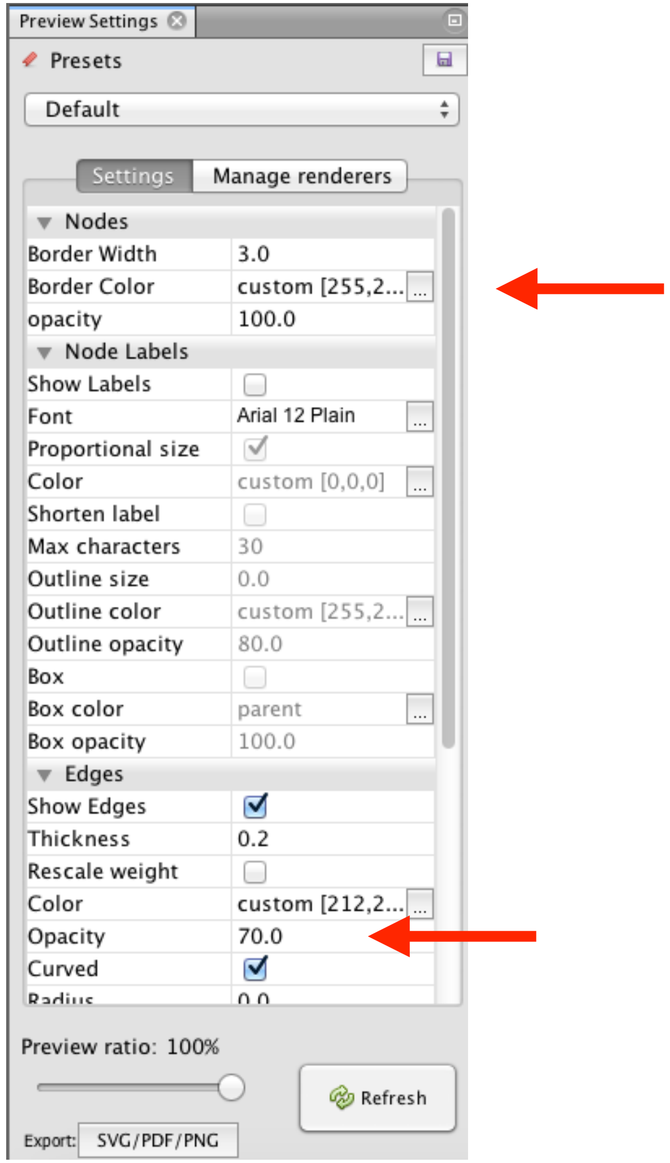


You'll see something like this. Say we want to filter all nodes south of 46.3 degrees of Latitude. Apply the filter

Filtering the graph

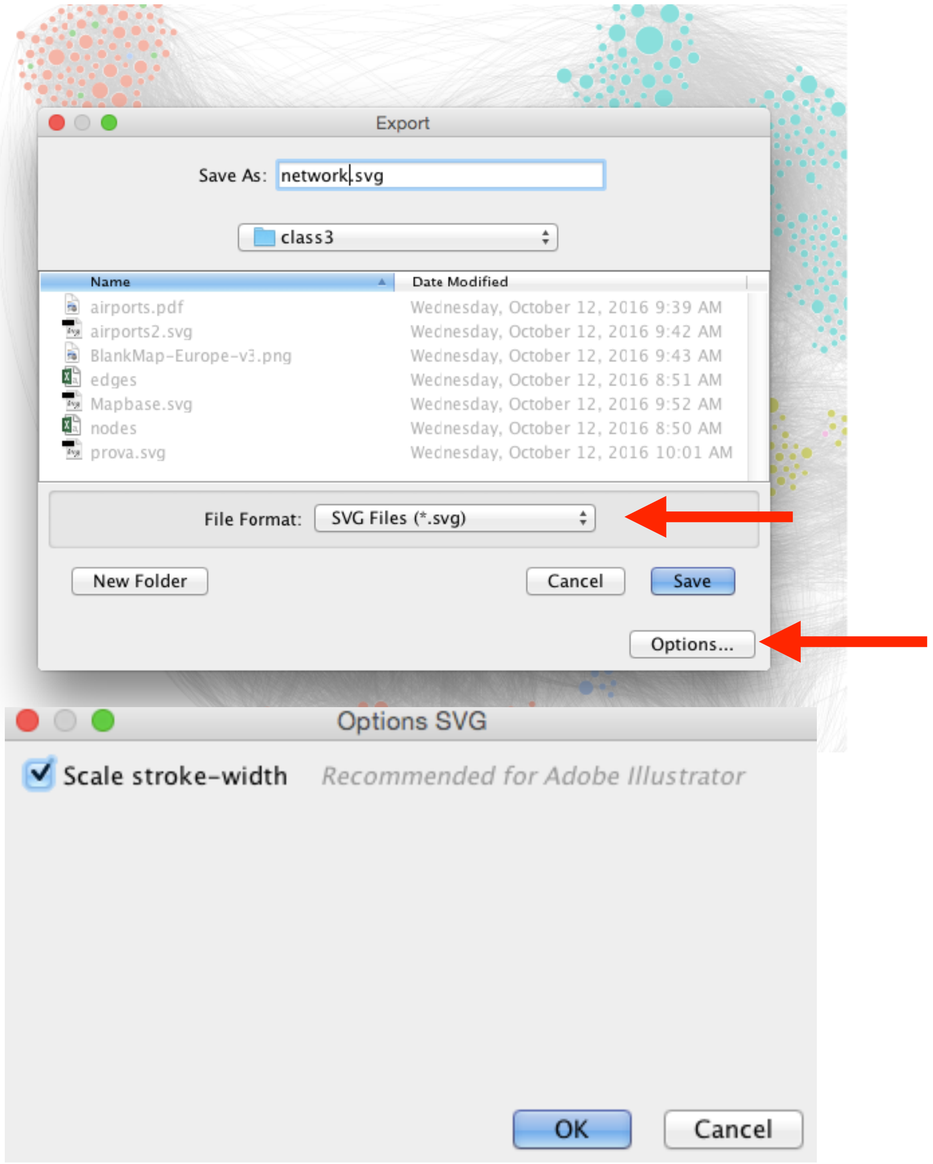
Some of the nodes in your network will disappear

Export the file

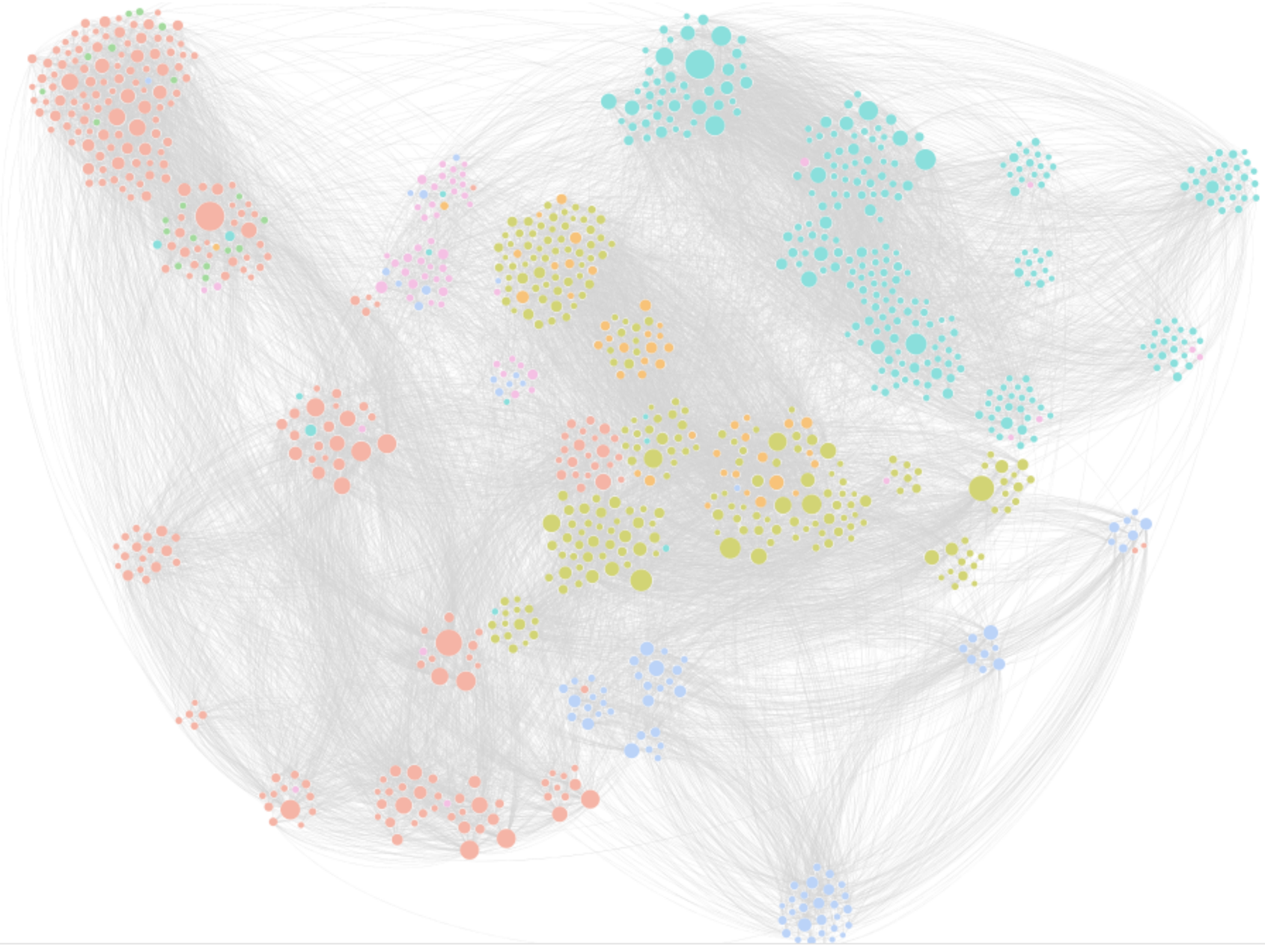


In the Preview panel, check the final appearance of your artwork and export it in .svg. Suggestions: use a white thick border for the nodes, and use opacity 70 for the edges. Remember to click Refresh for every change.

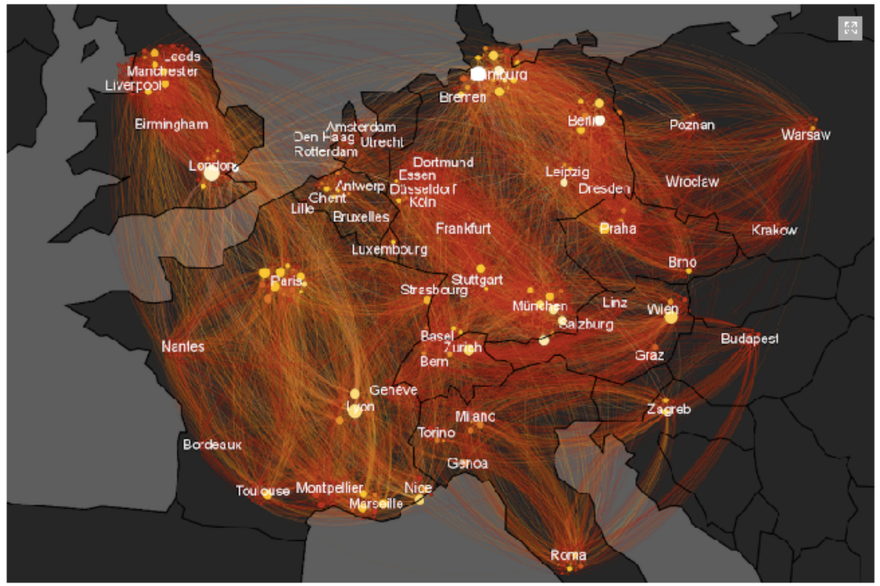
Export the file



Export in svg. Remember to click options and check the scale stroke option.



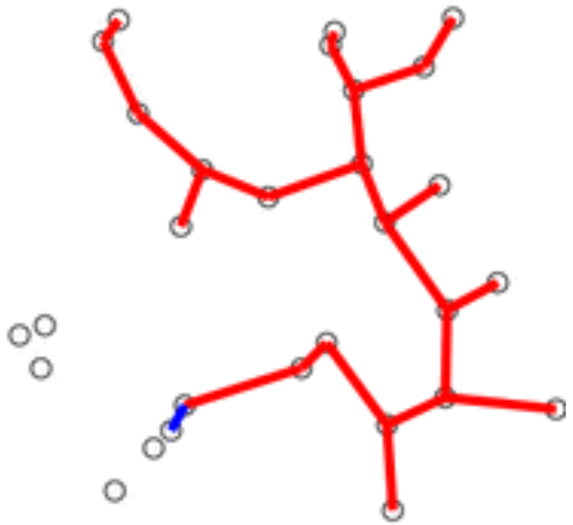
Exercise: Embedding in a map



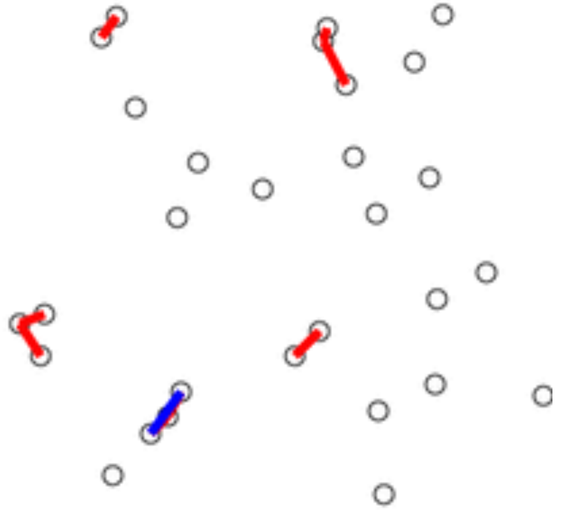
With a program like Adobe Illustrator or Inkscape, take the map Mapbase.svg (provided with the class materials and created to fit with the chosen scale and Mercator projection). Open it, and after having imported your network in it, select the city names layer and bring it to the front to make it readable.

Today you learned about minimum spanning tree algorithms

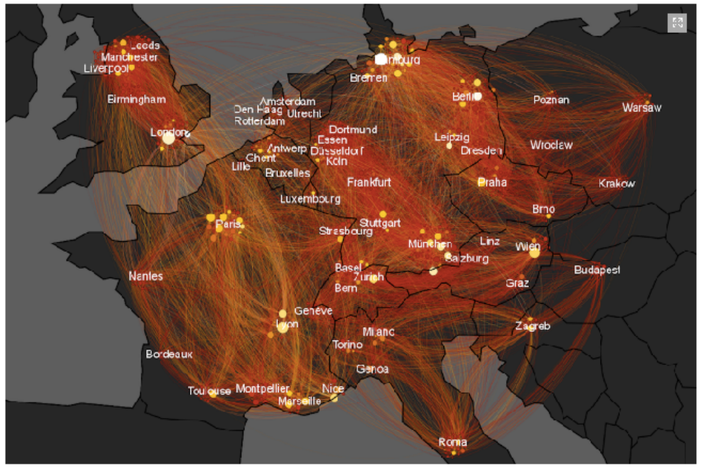
Prim



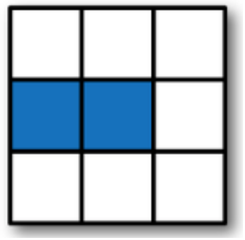
Kruskal



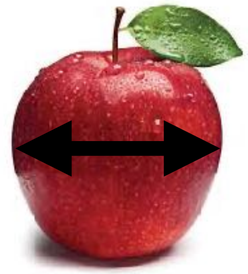
Mapping with Gephi



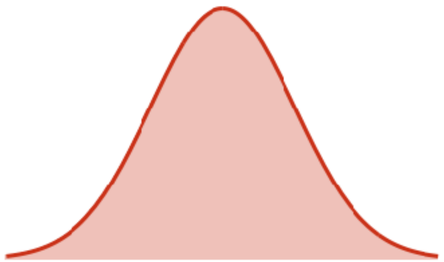
numpy



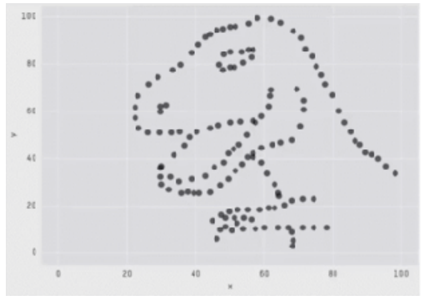
single variable analysis



normal distributions

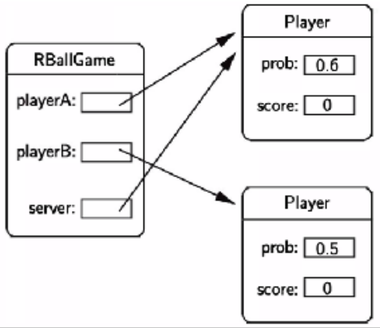


data relationships

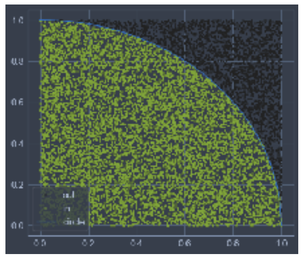


simulation and top-down design

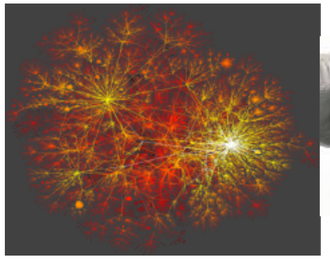
object-oriented programming



What are your questions?

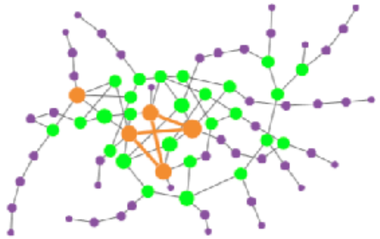


make code faster

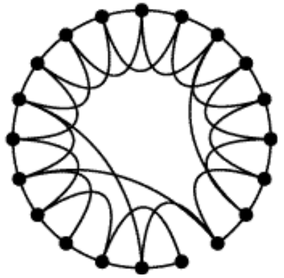


network science

skewed data

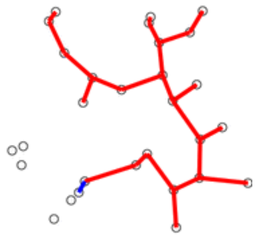


graph properties



network models

network analysis



graph algorithms

Go to www.menti.com and use the code **XX XX XX**