Graphs and Complexity

4) Connectivity and related problems

Irena.Rusu@univ-nantes.fr

LS2N, bât. 34, bureau 303

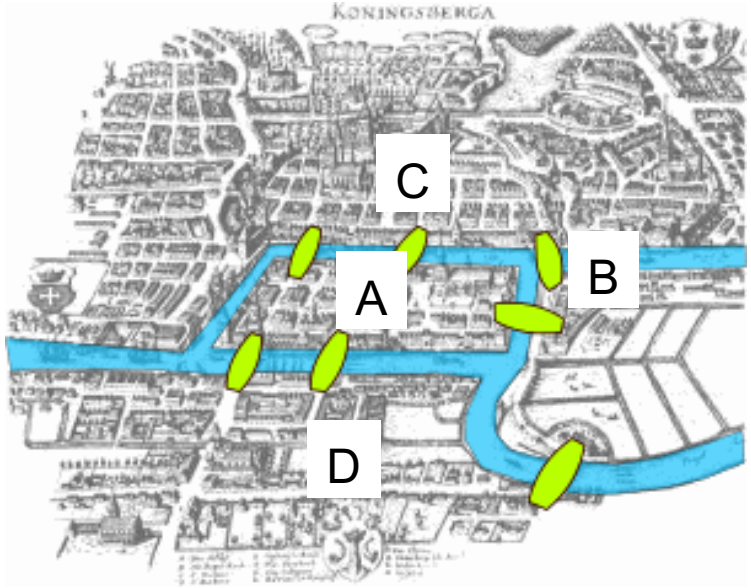
tél. 02.51.12.58.16

Plan

- Connectivity (undirected graphs)
- Strong connectivity (digraphs)
- Finding eulerian cycles/paths

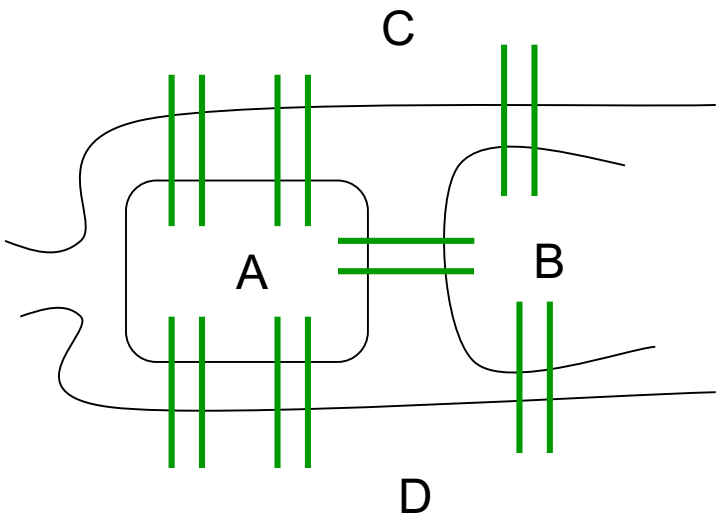
Back to the bridges of Königsberg

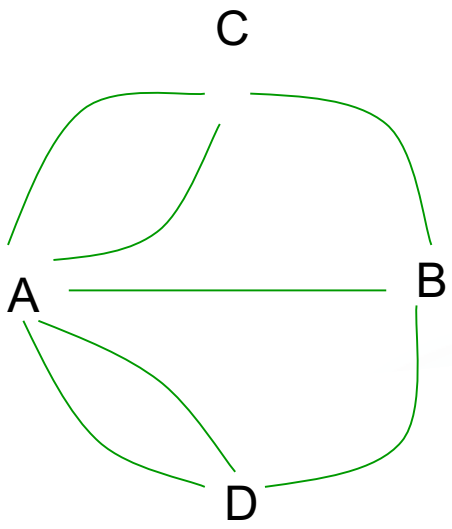
There are seven bridges in the town called Königsberg, in Prussia (now Kaliningrad, Russia). Is it possible have a walk in this town that uses each bridge exactly once?



© Bogdan Giuşcă, Wikipedia

What kind of a path are we looking for? In which graph? Which are the necessary and/or sufficient conditions for the existence of such a path?





- Eulerian path
- Existence ? (tip: suppose such a path exists and deduce properties of the graph)

→ It is necessary to:

connectivity

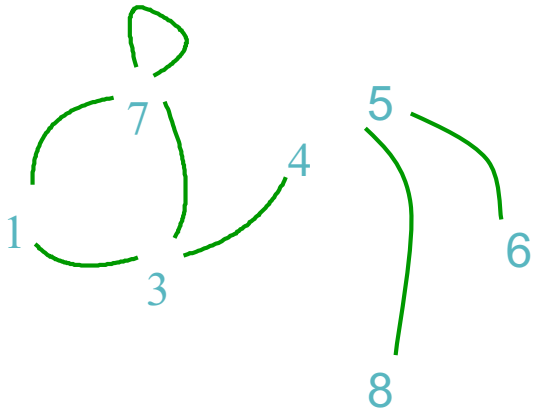
- Be able to join every two vertices by a path
- Be able to reach (1 edge) and leave (1 edge) each vertex, one or several times (except the endpoints, if they are distinct)

degree parity

Connected graph/component

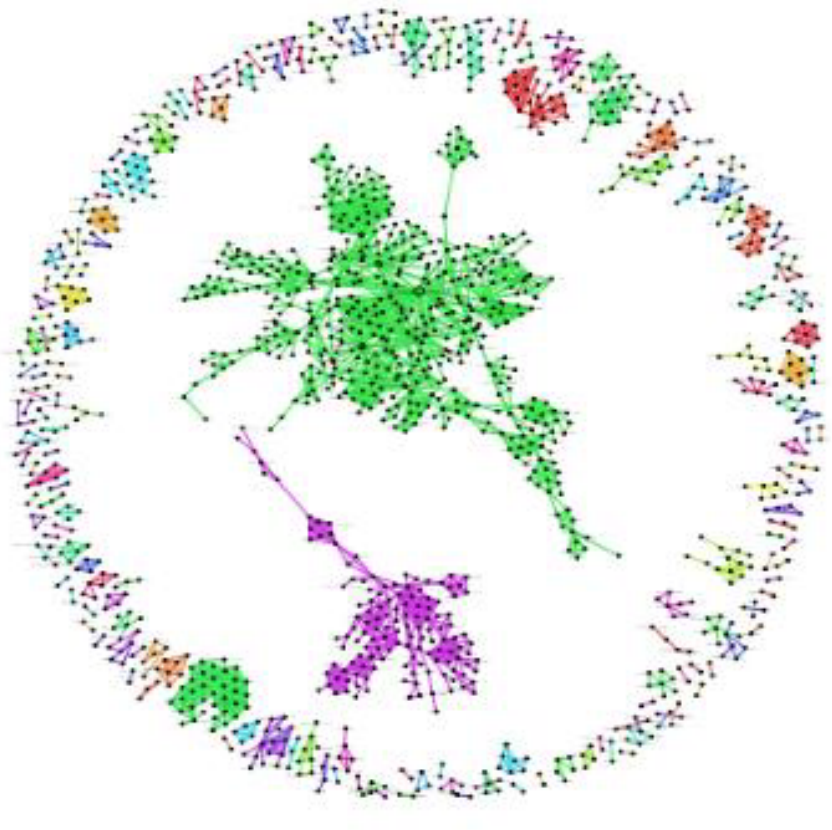
$G=(S,A)$ undirected graph
$|S|=n,\ |A|=m$

Connected graph: for each pair (s,s') of distinct vertices, there is a path joining s and s'.



Connected component (CC) of G :
maximal induced subgraph of G that is connected.

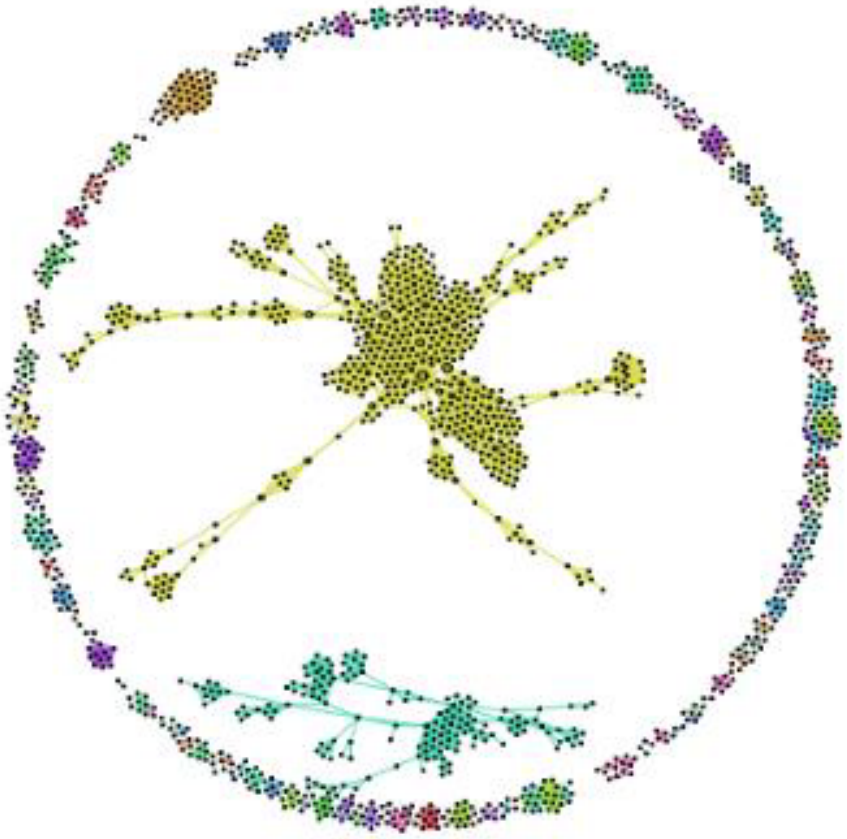
Application



© https://research.cchmc.org/od/02/

- Vertex: orphan disease
- Edge: joins two orphan diseases involving a common gene
- Connected components are indicated by colors
- Meaning: the diseases in the same connected component are somehow related.

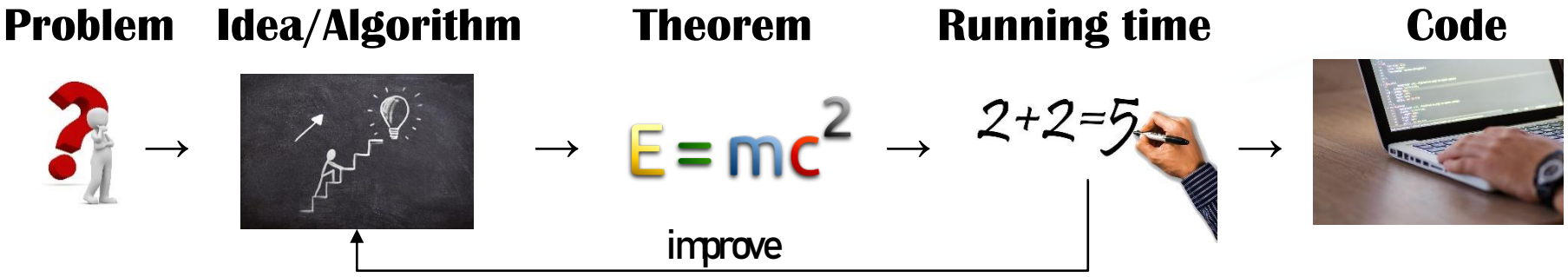
Different viewpoint



© https://research.cchmc.org/od/03/

- Vertex: gene
- Edge: joins two genes involved in a common disease
- Connected components are indicated by colors
- Meaning: genes in the same connected component are somehow related.

Finding the CC of a graph



Problem

- $G=(S,A)$ undirected graph
- Find all the connected components (CC) of G

Algorithm

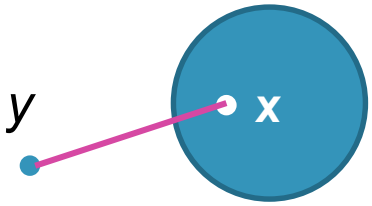
- A first idea (s is an arbitrary vertex):

  1) Start with C=$\{s\}$ (which induces a connected subgraph)

  2) Grow C by successively adding to it an external vertex y adjacent to some vertex x from C, as long as possible. Then C is a CC.

  3) If there is a vertex belonging to no CC, start another CC using it.

  4) And so on.



Quick check of correction and running time:

Correction:

- Let C be one of the sets built above
- Connectivity of the induced subgraph: yes
- Maximality: yes

→ **Theorem** ok (not written, but could be).

Algorithm

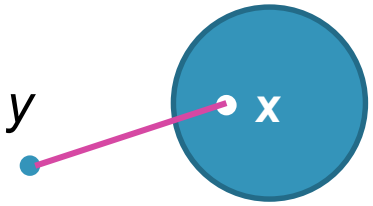
- A first idea (s is an arbitrary vertex):

  1) Start with C=$\{s\}$ (which induces a connected subgraph)

  2) Grow C by successively adding to it an external vertex y adjacent to some vertex x from C, as long as possible. Then C is a CC.

  3) If there is a vertex belonging to no CC, start another CC using it.

  4) And so on.



Running time (not really the best evaluation):

Step 2): we look for the **worst** case.

Adding one vertex to C:

- Tests for one external vertex: at most $|C|$ (the current cardinality). The tests may fail.
- Tests for $(n-|C|)$ external vertices: $|C|(n-|C|)$

Adding $n-1$ vertices to C (when G is a path, for instance):

$$\sum_{i=1}^{n-1} i(n-i) = \Theta(n^3)$$ tests (compute the sum !)

At worse, $\Theta(n^3)$ tests with 1 CC, probably similar with several CCs.

Conclusions

- Running time of $\Theta(n^3)$ probably not optimal
- *Don't bother to*
  - Write an algorithm
  - Prove a theorem
- Instead, look for a better algorithm

Improvement

Algorithm

- Another idea:

  1) Start with $C=\{s\}$ (which induces a connected subgraph)

  2) Grow C by performing a graph search starting with s

  3) If there is a vertex belonging to no CC, start another CC using it.

  4) And so on.

This is a *unique* graph search

- Correction: connectivity OK, maximality OK
- Algorithm: write it before you write the program

$\Theta(m+n)$ or $\Theta(n^2)$ depending on the graph representation

DFS-based algorithm for CC

Function DFS_CC (*s, compteur*)

```
begin
    CC [s] ← compteur ;
    for  each successor t of s do
        if CC[ t ] = 0 then
            DFS_CC (t, compteur)
        endif
    endfor
end
```

For the whole graph G

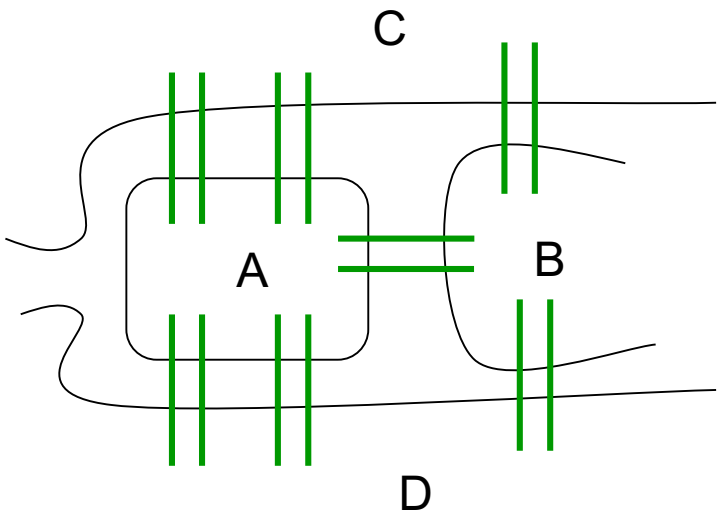
```
    for each vertex s of G  do
        CC[s] ← 0
    endfor
    compteur ← 1
    for each vertex s of G  do
        if CC [s]=0 then
            DFS_CC (s,compteur)
            compteur  ← compteur +1
        endif
    endfor
```

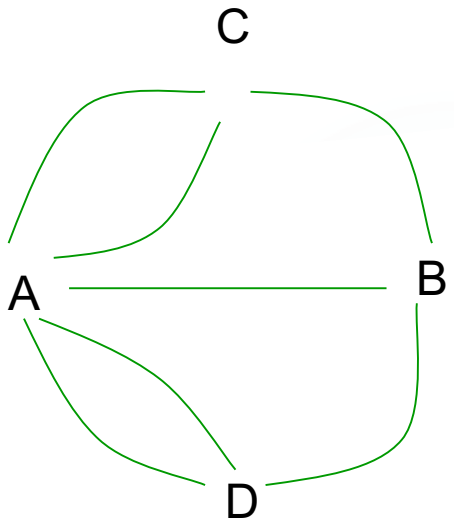
Running time: $\Theta(m+n)$ or $\Theta(n^2)$ depending on the graph representation.

Note. *C* is NOT represented as a set. Why ?

Test “belongs to no CC” too long !

Euler's theorem (solving Bridges of K.)





$G=(S,A)$ undirected graph

Theorem. Graph G with no isolated vertex (i.e. of degree 0) admits an eulerian path if and only if

- G is connected, and
- G has 0 or 2 vertices of odd degree.

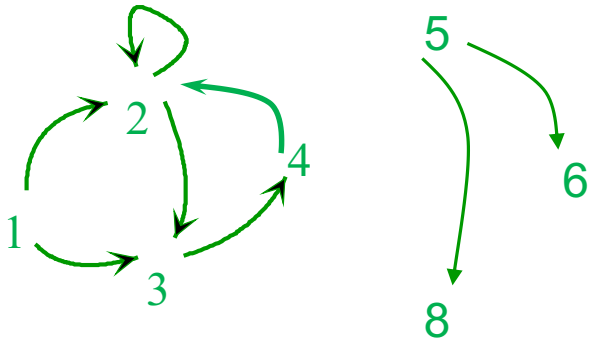
Plan

- Connectivity (undirected graphs)
- Strong connectivity (digraphs)
- Finding eulerian cycles/paths

Strong connectivity

$G=(S,A)$ directed graph
$|S|=n$, $|A|=m$

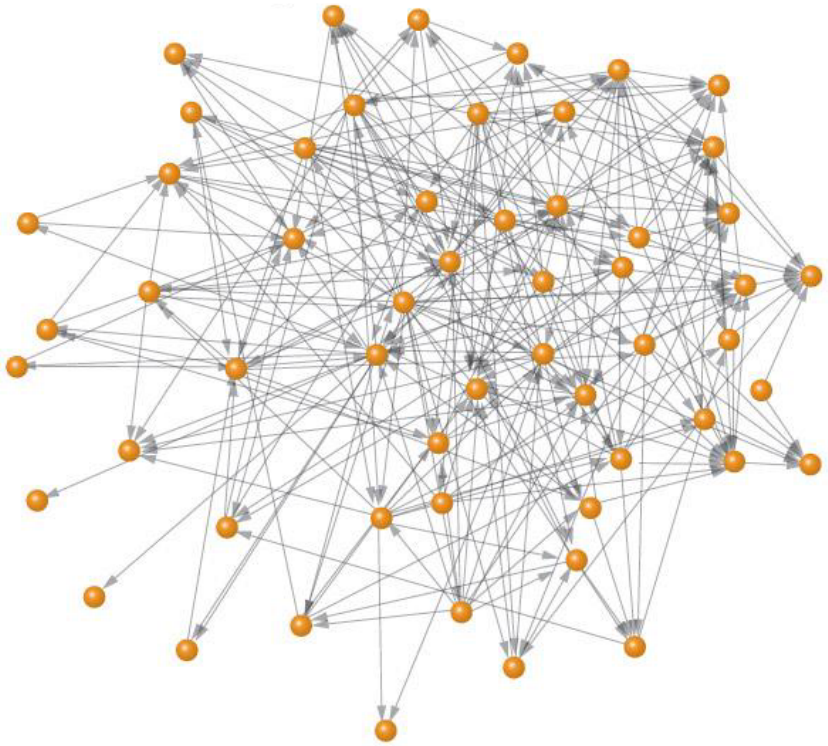
Strongly connected graph: for each pair (s,s') of distinct vertices, there is a path joining s and s'.



Strongly connected component (SCC) of G : maximal induced subgraph of G that is strongly connected.

Same definition as for connected graphs/components, but important changes!

Application



- Twitter-like network of followers

  A →B : A follows B

- Assume that if A →B then A is influenced by B

- A strong connected component groups people that are influencing each other (not necessarily directly).

© https://blogs.cornell.edu/info2040/2015/09/10/a-diverse-twitter-network-can-generate-better-ideas illustrating publication from MIT Sloan Management Review 56(4):21-25, 2015.

Twitter « Elite » Network

- *N*-Elite, where *N* is an integer: top *N* most followed (that is, most influential) Twitter users

Concerns years 2016, 2018

Table 1 Basic connectivity features of different views of the elite network and their strongly connected components for $S16$ and $S18$ napshots

	ALL_2016				LSCC_2016		All_2018				LSCC_2018	
View	$\|E\|$	Rcp	Diam	#SCC	$\%\|V\|$	$\%\|E\|$	$\|E\|$	Rcp	Diam	#SCC	$\%\|V\|$	$\%\|E\|$
1K-Elites	40K	0.35	4	64	93.5	94.55	37K	0.38	5	70	91.89	94.17
2K-Elites	104K	0.34	3	109	94.25	95.61	100K	0.36	4	139	92.74	94.81
3K-Elites	194K	0.32	4	169	94.13	95.76	181K	0.34	4	215	92.6	94.97
4K-Elites	289K	0.31	4	229	94.02	95.95	273K	0.34	4	264	93.15	95.36
5K-Elites	413K	0.32	4	277	94.2	96.13	376K	0.33	4	301	93.76	95.65
6K-Elites	543K	0.33	4	336	94.12	96.21	485K	0.33	4	355	93.86	95.77
7K-Elites	676K	0.34	4	369	94.5	96.44	607K	0.34	4	409	93.94	95.86
8K-Elites	841K	0.37	4	400	94.77	96.66	735K	0.34	4	454	94.11	95.96
9K-Elites	1.01M	0.4	4	438	94.92	96.91	903K	0.34	4	484	94.43	96.14
10K-Elites	1.15M	0.42	4	453	95.08	97.05	1.1M	0.35	4	526	94.55	96.23

Rcp: percent of reciprocal arcs

LSCC: largest SCC

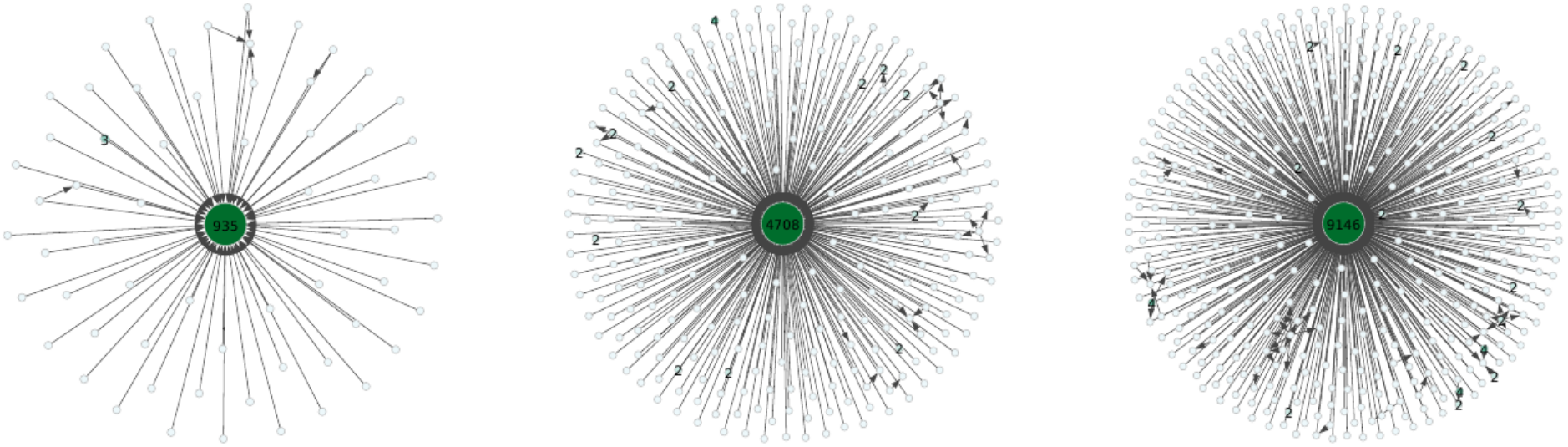
© This example, as well as the terms we use, come from Examining the Evolution of the Twitter Elite Network, *Social Network Analysis and Mining* volume 10, Article number: 1 (2020).

Reading the table

Regardless to the value of *N* (when *N* is large enough, e.g. *N*=1K, 2K etc, where K=kilo!):

- The « elite » network is connected and has a small diameter
- There is always a huge SCC (the one called LSCC)
- A little more than 1/3 of the connections are reciprocal (i.e. bidirectional)

The reduced graphs (1SCC=1vertex)



(a) 1K-ELITE (b) 5K-ELITE (c) 10K-ELITE

The connectivity of strongly connected components of the elite networks.

- The general shape of the network is star-like.

Note: here, the arcs are reversed: A→B indicates that A is followed by B.

Who are the « Elites »?

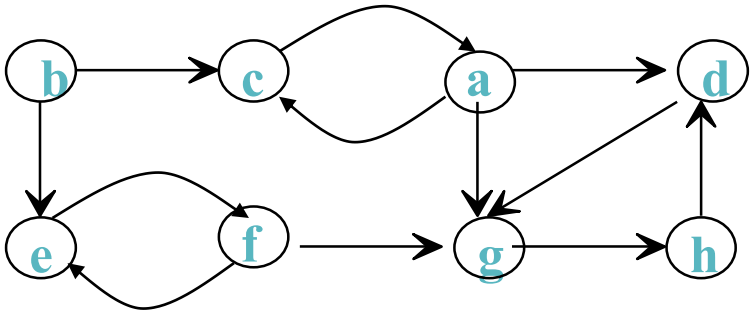
- N=10:

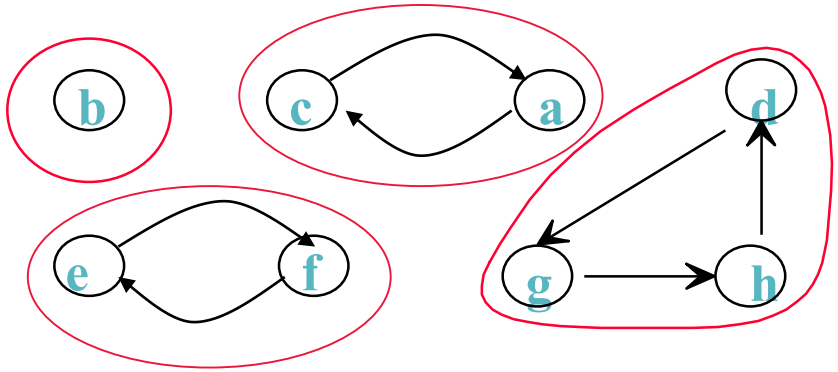
Table 2 The node level connectivity of the top 10 most-followed elites in 2018 across both snapshots

name	screen_name	followers_2018	friends_2018	followers_2016	friends_2016	Rank_change	Cur._Rank
KATY PERRY	katyperry	107.8M	<1K	81.8M	<1K	0	1
Justin Bieber	justinbieber	105.2M	303K	74.3M	254K	0	2
Barack Obama	BarackObama	103.5M	618K	69.1M	638K	-1	3
Rihanna	rihanna	88.7M	1K	55.4M	1K	-2	4
Taylor Swift	taylorswift13	84.0M	<1K	70.2M	<1K	2	5
Lady Gaga	ladygaga	77.8M	126K	55.0M	131K	-1	6
Ellen DeGeneres	TheEllenShow	77.1M	36K	53.3M	37K	-1	7
Cristiano Ronaldo	Cristiano	75.7M	<1K	40.2M	<1K	-6	8
YouTube	YouTube	71.6M	1K	59.0M	1K	4	9
Justin Timberlake	jtimberlake	65.1M	<1K	51.6M	<1K	0	10

- Friends: input degree
- Followers: output degree

Finding the SCC of a digraph





Problem

- Digraph $G=(S,A)$
- Find all the SCC of G.

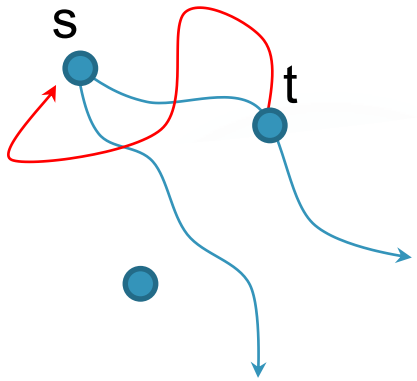
Algorithm

- A first idea (with hints for the **Theorem** proving its correction):

1) Start with $C=\{s\}$ (an arbitrary vertex).

2) A graph search starting in s reaches all the vertices t such that there is a path from s to t

→ set $P(s)$.



3) For each t from $P(s)$, a graph search starting in t allows to know if there is a path from t to s.

4) Add to C all the vertices t from $P(s)$ for which a path is found in the previous set. Then C is a SCC.

5) If there is a vertex belonging to no SCC, then use it to start another SCC, and so on.

Running time (quick estimation):

- In the worst case, at least one search for each vertex (for instance if G is a directed path)

→ $\Theta(n(m+n))$ or more

1st improvement

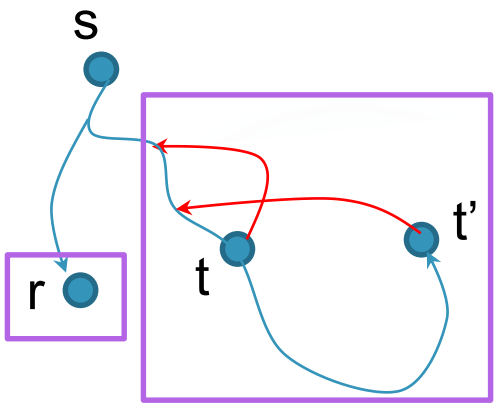
- Second idea (Tarjan's algorithm):

  1) Start with $C=\{s\}$ (an arbitrary vertex).

  2) A graph search starting in s reaches all the vertices t such that there is a path from s to t...

  3) … But also builds cycles using the back arcs.

  4) An SCC is a union of cycles (possibly trivial, i.e. of size 1) sharing one or several vertices, that cannot be grown by adding other cycles.



The graph search must maintain a stack of vertices not yet assigned to a SCC

→ Intuitive algorithm, but a bit long.

Running time:

- Performs one graph search (of the entire graph)

→ $\Theta(m+n)$ or $\Theta(n^2)$

2nd improvement

- Not intuitive, but short!

- 3rd idea (G^T = transposed graph of G):

  1) Note that a path from t to s in G is a path from s to t in G^T

  2) Identify the vertices s for which the search in G^T allows to build a SCC.

  3) Write it smartly and prove it…

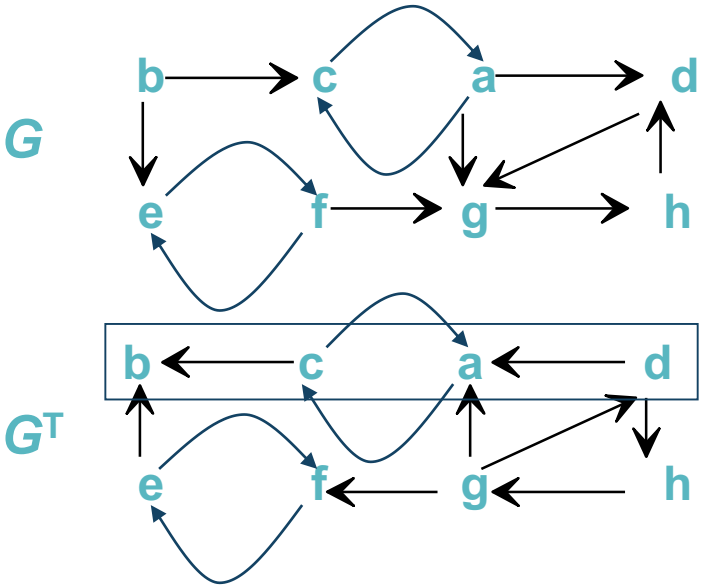
Note. A **theorem** exists showing the correction (but it is not given here)

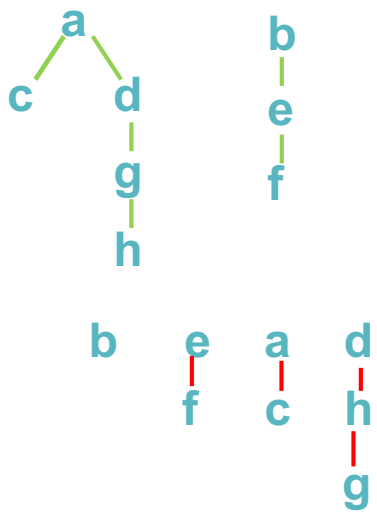
Algorithm [Kosaraju 78] [Sharir 81]

1) DFSNum search in G for computing the values $f(s)$

2) DFS search in G^T considering the vertices starting a new search in decreasing order of their values $f(s)$.

3) each search tree of the DFS search defines one SCC.





Running time:

- Performs two graph searches (of the entire graph)

→ $\Theta(m+n)$ or $\Theta(n^2)$

Plan

- Connectivity (undirected graphs)
- Strong connectivity (digraphs)
- Finding eulerian cycles/paths

Euler's-like theorem for digraphs

Theorem (König)

Digraph $G=(S,A)$ with no isolated vertex has an **eulerian circuit** if and only if:

- G is strongly connected, and
- each vertex of G has equal in-degree and out-degree.

Do it yourself

Write the similar theorem for the **eulerian path**, and justify it (assuming that König's theorem is true).

Eulerian Path/Cycle Problem

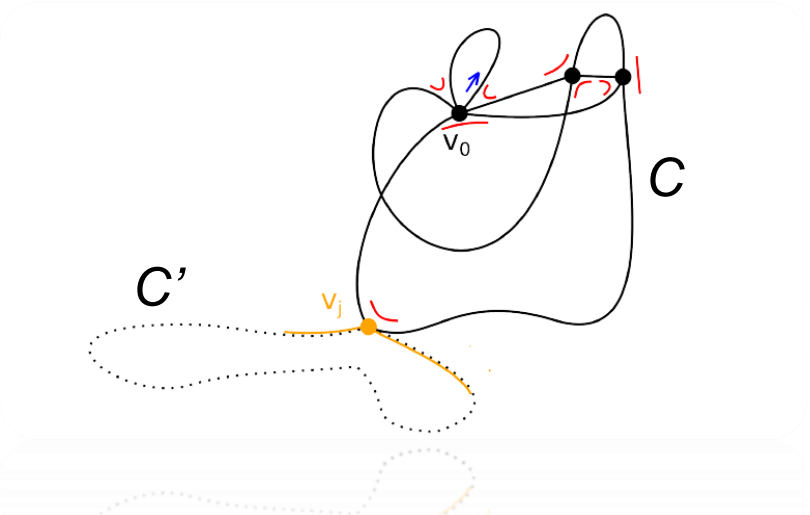
- Several questions (similar to knight's tour problem)
  - Characterizing: which precise graphs have an eulerian path/cyle?
    - Euler's theorem (undirected) and König's theorem (directed)
  - Solving the decision problem: is it easy/difficult to test whether a given graph satisfies the conditions in the characterization?
    - Connectivity (respectively strong connectivity) and degree conditions are tested in $\Theta(m+n)$
  - Finding a solution: how to build an eulerian path/cyle?
  - Counting: how many eulerian paths/cycles exist in a given graph?

Finding an eulerian cycle

$G=(S,A)$ undirected graph

Assume G has an eulerian cycle, by Euler's theorem.

Goal: Find an eulerian cycle in G.



Hierholzer's algorithm (rough description)

Start with an arbitrary vertex v_0

Build a cycle C starting with v_0 and using unvisited edges as long as possible.

While C does not contain all the edges in A do

Let w be a vertex of C that is the endpoint of an unvisited edge

Build a cycle C' starting with w and using unvisited edges as long as possible

Let C be the new cycle obtained by visiting C until w is reached, then visiting the whole C' (w is reached again), and then visiting the remaining part of C.

Return C.

Hierholzer's algorithm

```
function EulerianCycle(G) : stack          // modifies G !
Let v_0 be any vertex of G.
W ← Push(Empty-stack, v_0); Result ← Empty-stack;
while not IsEmpty(W) do
      u ← Top(W)
      while degree(u) > 0 do    //compute all the degrees once, and then update them
            Let v be a vertex adjacent to u
            W ← Push(W, v)
            Delete edge uv from G  // modifies the degrees of u and v
            u ← v  // u is the top of W
      end while
      while not IsEmpty(W) and degree(u)= 0 do
            W ← Pop(W,u)
            Result ← Push(Result,u)
            u ← Top(W);
      end while
end while
return Result
```

Running time:

$\Theta(m+n)$

(en supposant test existence fait)

Choose the data structure and explain !

Do it yourself

- Algorithm for an eulerian path.
- Algorithms for digraphs.

- Don’t forget that some algorithms have demos, as for instance Hierholzer’s algorithm

https://algorithms.discrete.ma.tum.de/graph-algorithms/hierholzer/index_en.html

Counting eulerian paths/cycles

For cycles (for paths, use the counting for cycles …):

- Digraphs: so-called BEST theorem gives

$$nb_{cycles(G)} = tw(G) * \prod_{s \in S} (degree(s) - 1)!$$

where *tw(G)* is the number of rooted trees in *G*.

tw(G) is computed in polynomial time.

- Undirected graphs: problem belongs to NP-hard \ NP-complete

Very hard !