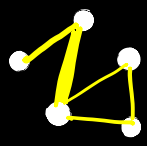
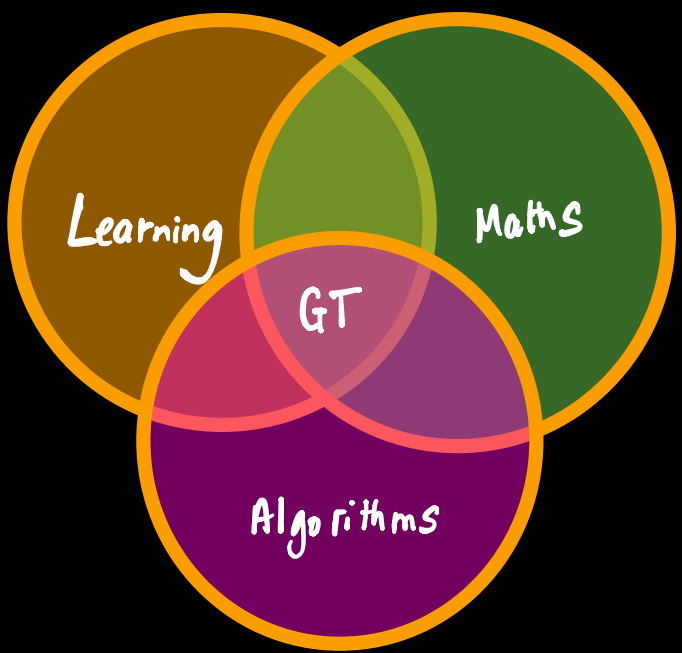
# GraphT 1

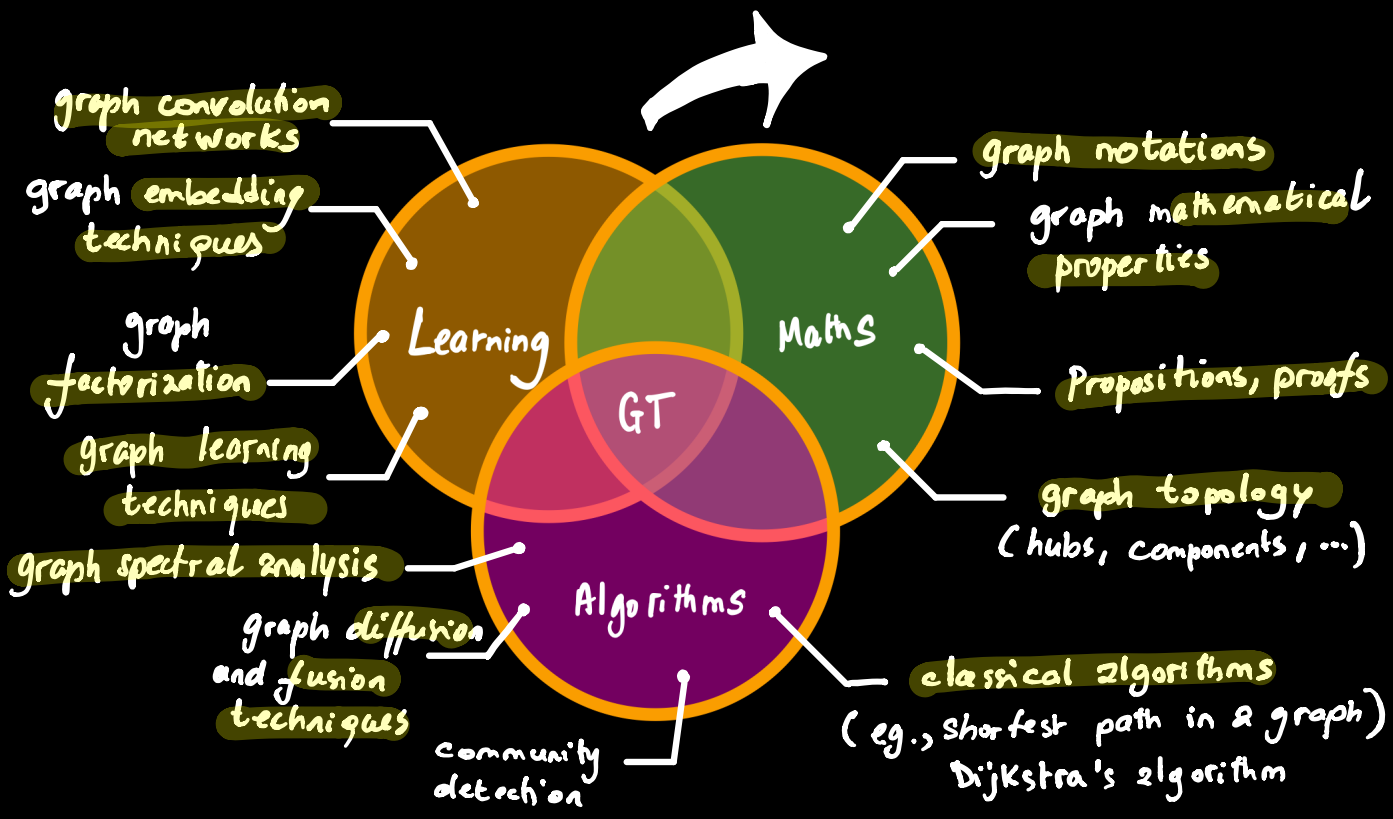
1. Why you are here?
"Graphs are everywhere!"

2. Basics of graph theory
   - graph representations
   - notations & examples
   - Types of graphs



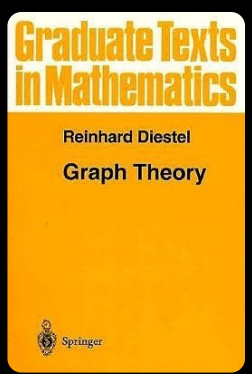
3. Introduction to basic topological measures of a graph

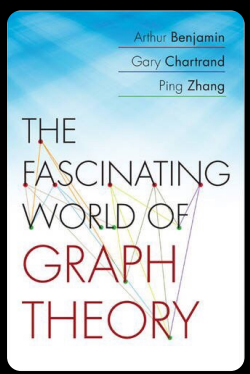


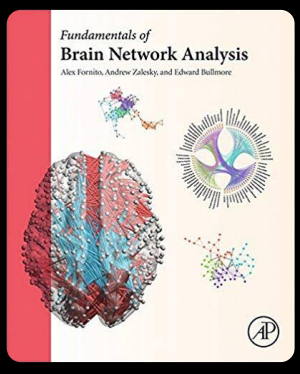



*inquisitive smart bee ... always self-quizzing ... searching for optimal solutions to complex problems ...*

1. What is a graph [taxonomy]?
2. What are the fundamental basics of graph theory [classical problems never age!]?
3. How to design graph-based methods to solve a real-world problem? [Learn well. Think better. Design better.]?
4. What about hypergraphs, multi-graphs, multiplexes, networks [More to learn.]?




• • •

# How to get the most out of the lecture?

1. *Think about the solution before it is revealed to you [Make it stick; Brown et al.].*
2. *Self-quizzing helps you better remember [Make it stick; Brown et al.]*
3. *Take notes by hand [don't take notes by a laptop; Mueller et al. 2014]*

**The Pen Is Mightier Than the Keyboard: Advantages of Longhand Over Laptop Note Taking**

Pam A. Mueller, Daniel M. Oppenheimer

First Published April 23, 2014 | Research Article | Check for updates
https://doi.org/10.1177/0956797614524581

Article information

Altmetric 3,648

A correction has been published: Corrigendum: The Pen Is Mightier Than the Keyboard: Advantages of Longhand Over Laptop Note Taking

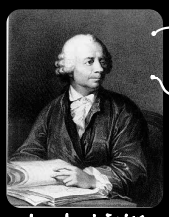
## Abstract

Taking notes on laptops rather than in longhand is increasingly common. Many researchers have suggested that laptop note taking is less effective than longhand note taking for learning. Prior studies have primarily focused on students' capacity for multitasking and distraction when using laptops. The present research suggests that even when laptops are used solely to take notes, they may still be impairing learning because their use results in shallower processing. In three studies, we found that students who took notes on laptops performed worse on conceptual questions than students who took notes longhand. We show that whereas taking more notes can be beneficial, laptop note takers' tendency to transcribe lectures verbatim rather than processing information and reframing it in their own words is detrimental to learning.

1. Why you are here? "Graphs are everywhere!"
2. Basics of graph theory
   - graph representations
   - notations & examples
   - types of graphs
3. Introduction to basic topological measures of a graph

# 1 Graphs are everywhere from bees to brains

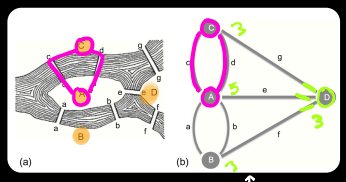
## HISTORY



Leonhard Euler (1707 - 1783)

mathematician, physicist, astronomer, geographer, logician, and engineer who laid the foundations of infinitesimal calculus and graph theory.

In 1735, he lived in Königsberg, which was built around 7 bridges. An unsolved problem around that time was whether it was possible to walk around the town via a route that crossed each bridge once and only once.




Euler solved this problem by representing the 4 land masses divided by the river as nodes and the 7 bridges as interconnecting edges.

This graph representation allowed him to prove that no more than 2 nodes (start & end points of the walk) should have an odd number of edges connecting them to the rest of the graph for such a walk to be possible.

"So is it possible to find a route around Königsberg city that crosses each and every bridge only once?

"Königsberg walk" is topologically prohibited.

"How can we use this proposition to build efficient/smart cities?"

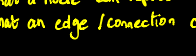
## BASIC DEFINITION

**Def** A graph is a pair $G=(V,E)$ of sets such that $E \subseteq |V| \times |V| = |V|^2$

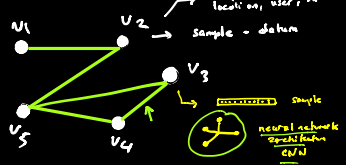
- $V$: set of nodes
- $E$: set of edges

elements of $E$ are 2-element subsets of $V$.

$e_1 = \{v_3, v_5\}$




- what a node can represent in a graph?
- what an edge / connection can represent in a graph?




## G-TAXONOMY

- A graph with vertex set $V$ is said to be a graph on $V$.
- The number of nodes of a graph $G$ is its order $\rightarrow |G|$.
- The number of edges of a graph $G$ is denoted by $\|G\|$.
- $G = (\emptyset, \emptyset) \Rightarrow$ empty
- A graph of order 0 or 1 is called trivial. $V = 0 \Rightarrow G = \emptyset$; $1 \Rightarrow |V| = 1$
- A vertex/node is incident with an edge $e$ if $v \in e$
  - $\rightarrow$ then $e$ is an edge at $v$.
- Two vertices are incident with an edge $e \in E$ are its end vertices. (ends)
- An edge joins its ends.
- The set of all edges at $v$ is denoted by $E(v)$.
- Two vertices are $(v_i, v_j)$ adjacent / neighbors if $\{v_i, v_j\}$ is an edge in $E$.
- Two edges are adjacent if they have an end in common.
- If all the nodes of $G$ are pairwise adjacent, then $G$ is complete.

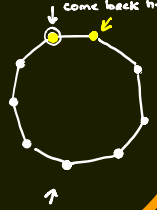

# FROM BEES TO BRAINS : GRAGHS

**Problem**: find the shortest tour which traverses all nodes of a graph $G$ only once.

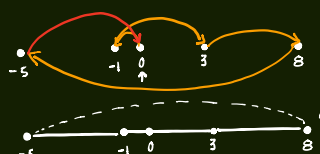
**Solution 1**:

"visit the nearest unvisited node"

start here, come back here



**Problem**: find the shortest tour which traverses all nodes of a graph $G$ only once.




- connect the closest pairs of nodes by a path

**Problem**: find the shortest tour which traverses all nodes of a graph $G$ only once.

- NP-hard problem
  - non-deterministic polynomial time 😕

Shortest for any graph.

- "Solutions" →

{"You think it is hard! I can solve it"

SCIENTIFIC REPORTS

OPEN

# Continuous Radar Tracking Illustrates the Development of Multi-destination Routes of Bumblebees

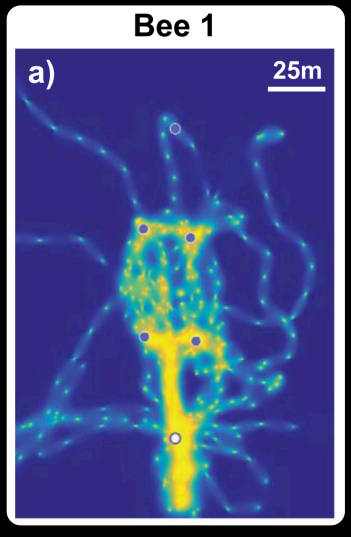
Received: 3 March 2017
Accepted: 28 November 2017
Published online: 11 December 2017

Joseph L. Woodgate[1,2], James C. Makinson[1,2], Ka S. Lim[2], Andrew M. Reynolds[2] & Lars Chittka[1,3]

Animals that visit multiple foraging sites face a problem, analogous to the Travelling Salesman Problem, of finding an efficient route. We explored bumblebees' route development on an array of five artificial flowers in which minimising travel distances between individual feeders conflicted with minimising overall distance. No previous study of bee spatial navigation has been able to follow animals' movement during learning; we tracked bumblebee foragers continuously, using harmonic radar, and examined the process of route formation in detail for a small number of selected individuals. On our array, bees did not settle on visit sequences that gave the shortest overall path, but prioritised movements to nearby feeders. Nonetheless, flight distance and duration reduced with experience. This increased efficiency was attributable mainly to experienced bees reducing exploration beyond the feeder array and flights becoming straighter with experience, rather than improvements in the sequence of feeder visits. Flight paths of all legs of a flight stabilised at similar rates, whereas the first few feeder visits became fixed early while bees continued to experiment with the order of later visits. Stabilising early sections of a route and prioritising travel between nearby destinations may reduce the search space, allowing rapid adoption of efficient routes.

The flight of bumble bees become more efficient over time

→ They learn how to solve the challenge to find a route that visits each destination (node) while travelling the shortest distance.




{"Want more evidence?"

VOL. 176, NO. 6 THE AMERICAN NATURALIST DECEMBER 2010

## Travel Optimization by Foraging Bumblebees through Readjustments of Traplines after Discovery of New Feeding Locations

Mathieu Lihoreau,[1] Lars Chittka,[1] and Nigel E. Raine[1,2,*]

1. Research Centre for Psychology, School of Biological and Chemical Sciences, Queen Mary University of London, Mile End Road, London E1 4NS, United Kingdom; 2. School of Biological Sciences, Royal Holloway University of London, Egham, Surrey TW20 OEX, United Kingdom

*Submitted March 24, 2010; Accepted August 10, 2010; Electronically published October 25, 2010*

*Online enhancements:* appendix figures.

**Science News** *from research organizations*

### Tiny brained bees solve a complex mathematical problem

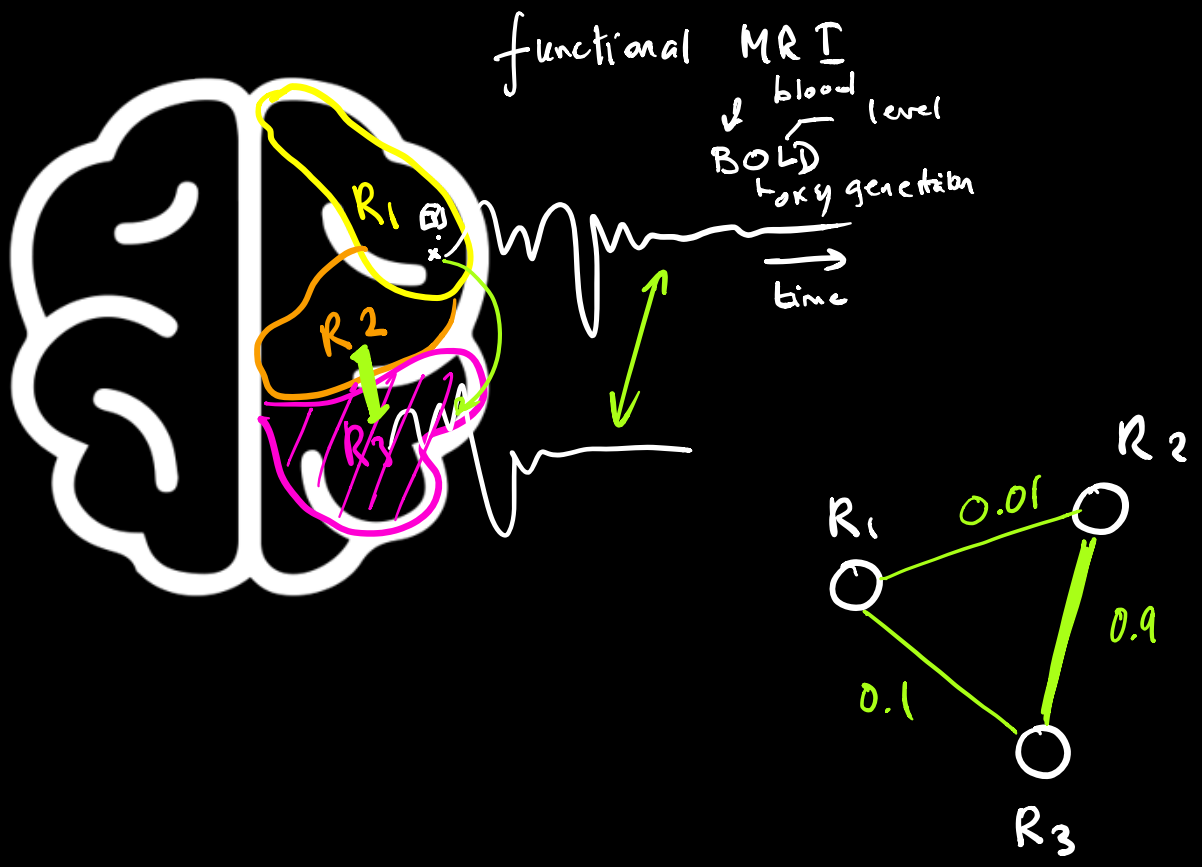
*Date:* October 25, 2010

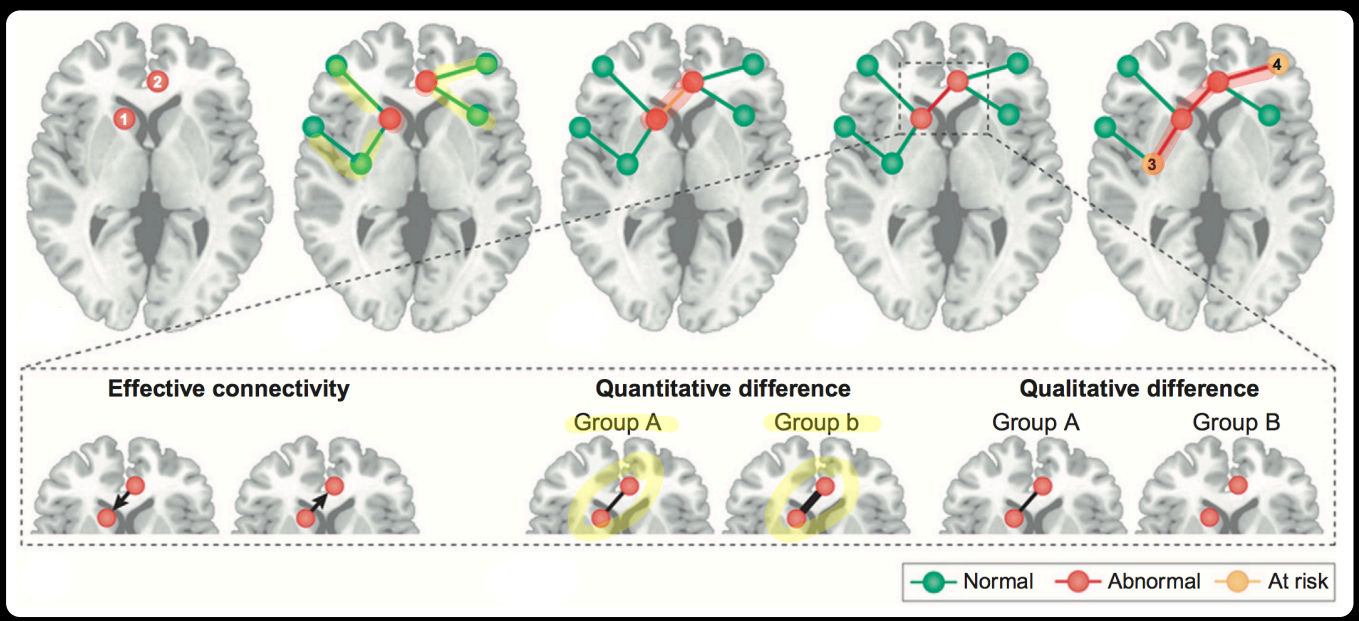
*Source:* University of Royal Holloway London

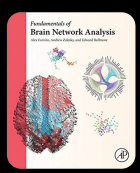
*Summary:* Bumblebees can find the solution to a complex mathematical problem which keeps computers busy for days. Scientists in the UK have discovered that bees learn to fly the shortest possible route between flowers even if they discover the flowers in a different order. Bees are effectively solving the 'Traveling Salesman Problem', and these are the first animals found to do this.

Dr Nigel Raine, from the School of Biological Sciences at Royal Holloway explains: "Foraging bees solve travelling salesman problems every day. They visit flowers at multiple locations and, because bees use lots of energy to fly, they find a route which keeps flying to a minimum."

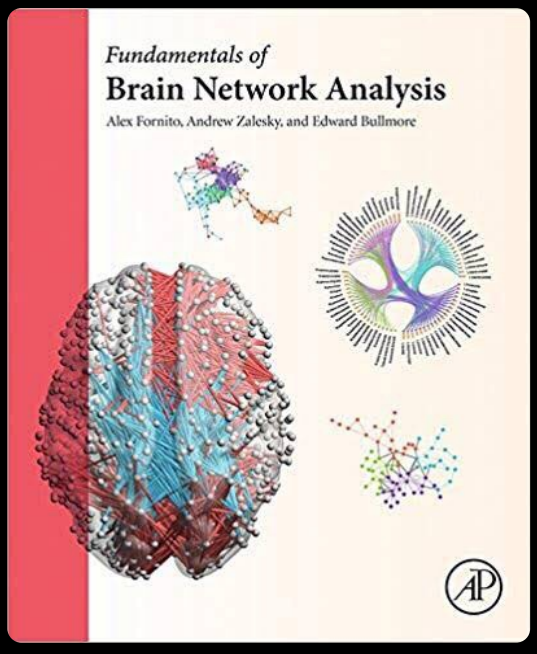
Dr Raine adds: "Despite their tiny brains bees are capable of extraordinary feats of behaviour. We need to understand how they can solve the Travelling Salesman Problem without a computer. What short-cuts do they use?'


functional MRI
blood level
BOLD
fMRI
time
R1
R2
R3
R1
R2
R3
0.01
0.9
0.1


1
2
3
4
Effective connectivity
Quantitative difference
Group A
Group b
Qualitative difference
Group A
Group B
Normal
Abnormal
At risk



fBNA.




ONE OF THE BEST BOOKS ON GRAPH THEORY & ANALYSIS WITH LOTS OF INTUITION AND A CLEAR AND SMOOTH FLOW OF NARRATIVE.

CHAPTER 3
Connectivity Matrices and Brain Graphs

CHAPTER 4
Node Degree and Strength

CHAPTER 5
Centrality and Hubs

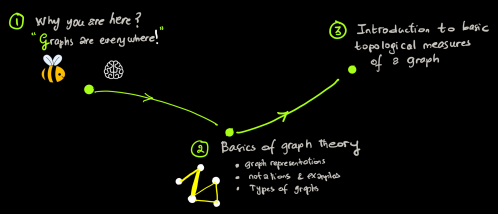
CHAPTER 6
Components, Cores, and Clubs

CHAPTER 6
Components, Cores, and Clubs

CHAPTER 7
Paths, Diffusion, and Navigation

CHAPTER 8
Motifs, Small Worlds, and Network Economy

CHAPTER 9
Modularity

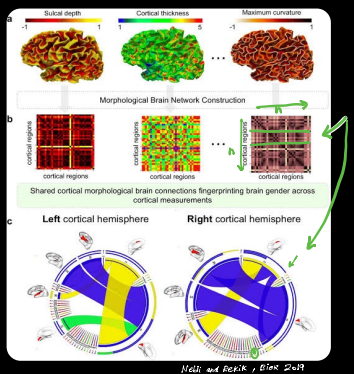


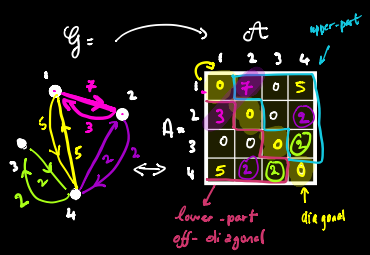

# 2 BASICS OF GRAPH THEORY

- Representation
- notations & operations
- graph types

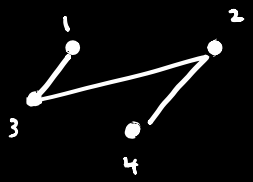
## Graph representation

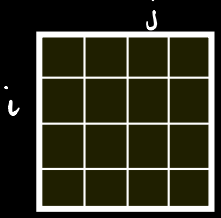
- Given a graph $G = (V, E)$ of order $n$ → its adjacency matrix $A \in \mathbb{R}^{n \times n}$ encodes the pairwise weights of its edges.
- Connectivity / adjacency matrix $A$ defines the pattern of pairwise adjacencies between nodes.



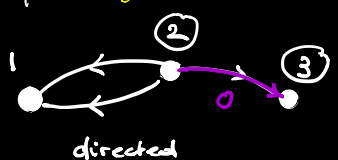



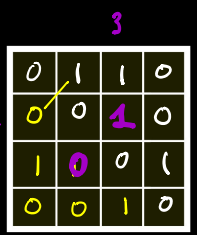

① if A is binary ⇒
$\forall i,j$ , $A_{ij} \in \{0,1\}$
- if A is symmetric ⇒ G is undirected



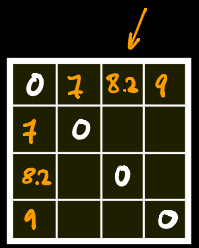



② if A is binary ⇒
$\forall i,j$ , $A_{ij} \in \{0,1\}$
- if A is asymmetric ⇒ G is directed.

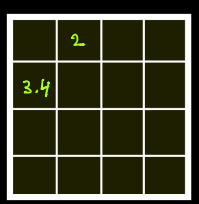


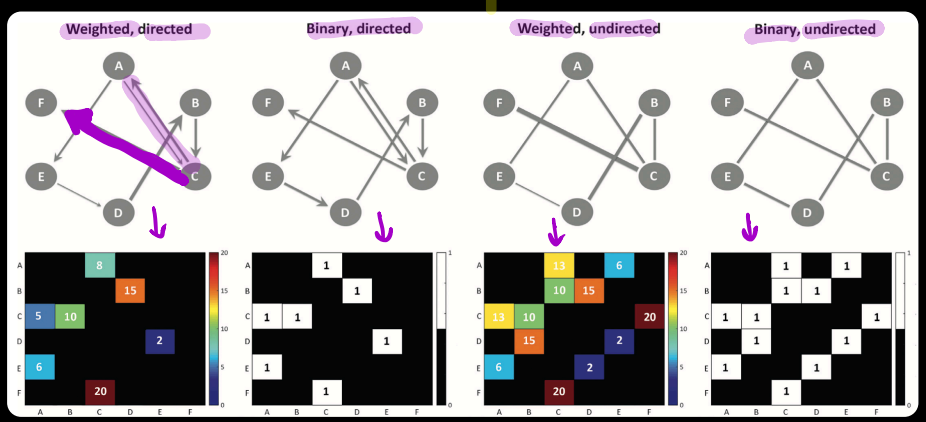



③ if A is weighted ⇒
$\forall i,j$ , $A_{ij} \in \mathbb{R}$
- if A is symmetric ⇒ undirected



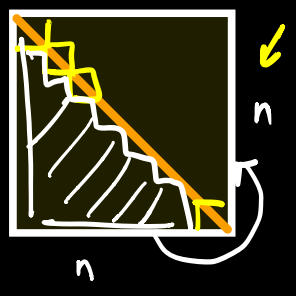
④ if A is weighted ⇒
$\forall i,j$ , $A_{ij} \in \mathbb{R}$
- if A is asymmetric ⇒ directed

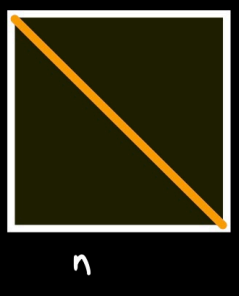




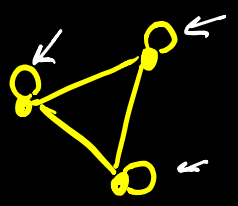
① What is the number of connections in a fully connected graph with n nodes (including self-connections)?

② What is the number of off-diagonal elements in the upper triangular part of $A_G$?

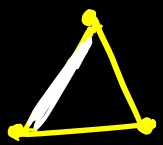
$A_G =$ 


$A_G =$ 


(undirected)
(directed)



$\frac{n(n-1)}{2} + n$



$\frac{n(n-1)}{2}$

$n \times n$

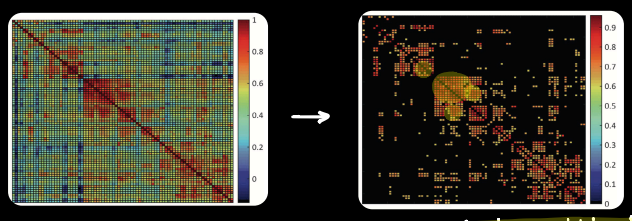
$n(n-1)$

## Graph basic operations

### Thresholding

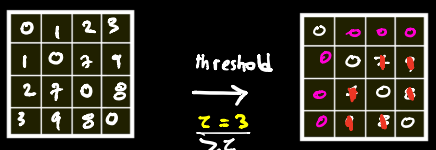
- Thresholding or sparsifying (if many values are removed >> 90%) at a specific cut-off threshold value $\tau$:

$$A^{\tau}_{ij} = A^{\tau}(i,j) = \begin{cases} A^{\tau}_{ij} & \text{if } A_{ij} > \tau \\ 0 & \text{otherwise} \end{cases}$$



☺ Thresholding a matrix to reduce the influence of low-weight and potentially spurious connections/edges on network/graph topology → remove some noise.

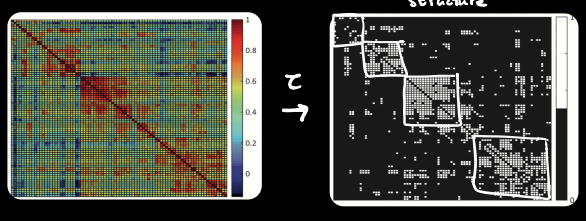
☺ Thresholding makes dense networks easier to process → it sparsifies the adjacency matrix by removing many edges ($\{e = \emptyset\}$)



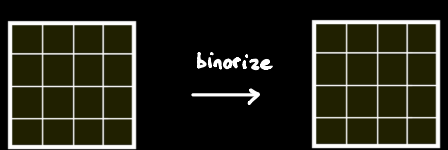

### BINARIZATION

- By applying a threshold to an adjacency matrix A, the remaining elements can be binarized such that:

$$A^{b}_{ij} = A^{b}(i,j) = \begin{cases} 1 & \text{if } A_{ij} > \tau \\ 0 & \text{otherwise} \end{cases}$$




☺ Analyses of binary adjacency matrices are principally concerned with understanding the topological patterns of connections between nodes, irrespective of variations in their weights (backbone).

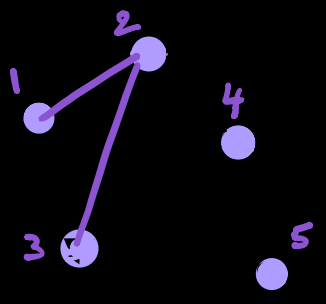


The adjacency matrix forms the basis of fundamental graph/network analysis.

## BASIC GRAGH PROPERTIES

* Number of edges $\|G\|$ in an undirected graph $G$?

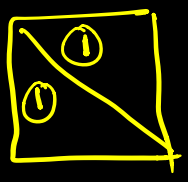
→ Sum all non-zero edges in the binarized network where $\begin{cases} A^b_{ij} = A_{ij} \\ \text{s.t. } A_{ij} \neq 0 \end{cases}$

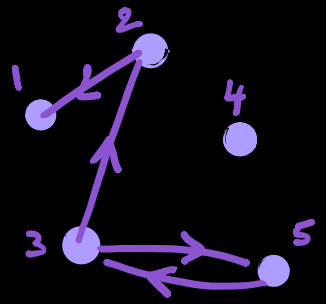



$$2\|G\| = \sum_{ij} A^b_{ij} \Rightarrow \|G\| = \frac{1}{2}\sum_{ij} A^b_{ij}$$

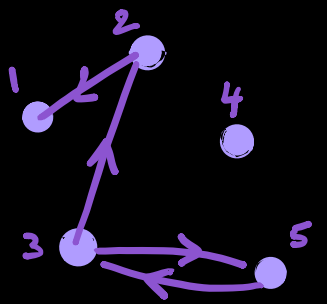
$2\|G\|$: number of edges

$\sum_{ij} A^b_{ij}$: binarized adjacency matrix




* Number of edges $\|G\|$ in a directed graph $G$?




$$\|G\| = \sum_{ij} A^b_{ij}$$

$\|G\|$: number of edges

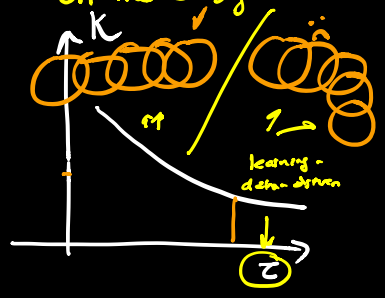
## * Graph connection density (connectance) K:

**Def:** The proportion of nonzero elements in A relative to the total possible number of connections that could be formed in G.

$|G|$

$\frac{n(n-1)}{2}$ (undirected) $=$ $n(n-1)$ (directed)

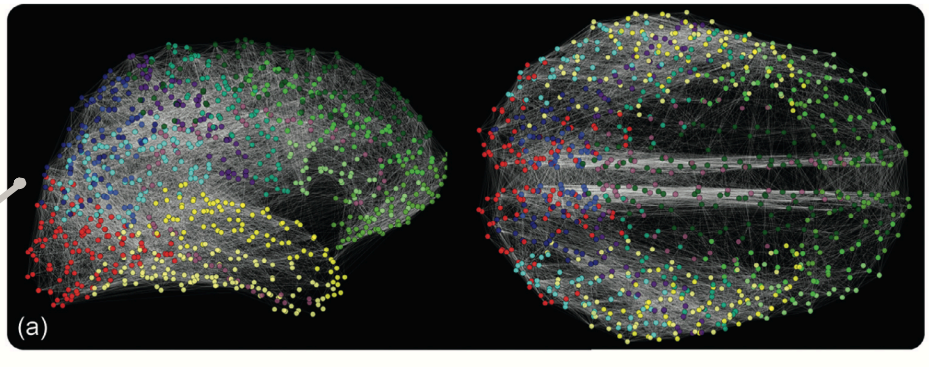
exclude self-connections on the diagonal



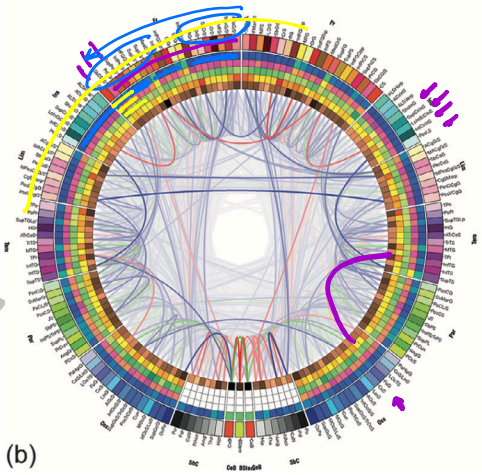

$$\begin{cases} K_d = \dfrac{\|G_d\|}{n(n-1)} \\ K_u = \dfrac{\|G_u\|}{\frac{n(n-1)}{2}} = \dfrac{2\,\|G\|_u}{n(n-1)} \end{cases}$$

## VISUALIZATION

brain graph (Gephi)




circular projection (connectogram) (circos)





a projection of fonctional connectivity graph with over 22k nodes & nearly 60M edges!

# GRAGH TYPES

## 1. Simple graph

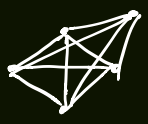
A graph is simple if it has no loops or multiple edges.


Loop
Self-connection
[An edge that connects up the same node]

Two edges that connect the same end points.
e.g. two routes connecting two cities.

## 2. Complete graph

A complete graph has each of its nodes connected to all other nodes.
→ every two nodes are adjacent



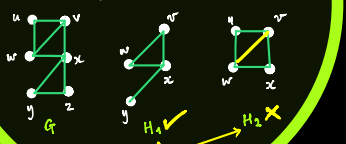
## 3. Subgraphs

↳ examine the structure of graphs that lie within a given graph.

- A graph H is called a subgraph of G if every vertex and edge of H belong to G. $V(H) \subseteq V(G)$ & $E(H) \subseteq E(G)$
- A graph G is a subgraph of itself.
→ all other subgraphs are proper subgraphs.
- If H is a subgraph of G having the same vertices as G, then H is a spanning subgraph of G.


## 4. Induced subgraph of G

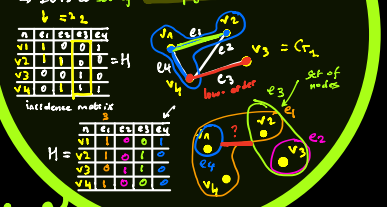
- Given a non-empty set $S \neq \phi$ of nodes of G, the subgraph $G[S]$ of G induced by S has vertex set S where two vertices u and v are adjacent if and only if (iff) u and v are adjacent in G.




## 5. Hypergraph

→ models high-order relationships among nodes.

A hypergraph H is a generalization of a graph in which an edge can join any number of vertices.
→ It is a set of non-empty subsets (or hyperedges).

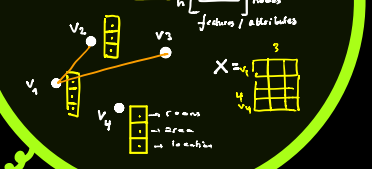



Feng et al. AAAI 2019

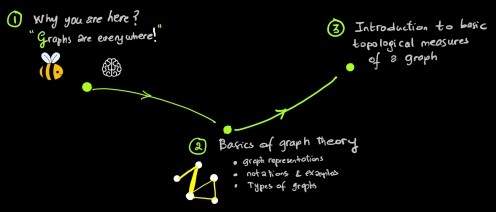
## 6. Annotated / decorated graphs

A graph G is annotated or decorated if its nodes can hold attributes.

$G = ((V, X), E)$
size n
size n × d
max size ?
nodes
features / attributes
n rooms
m areas
w locations



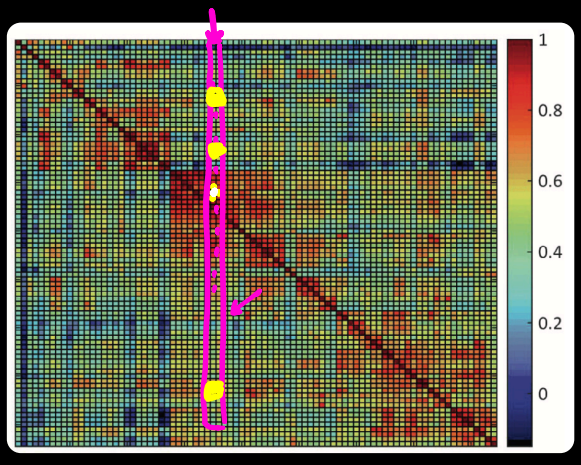

Murphy et al. 2016




# 3 INTRODUCTION TO BASIC TOPOLOGICAL MEASURES OF A GRAPH

{ What are the differences between an image and a graph ?

A



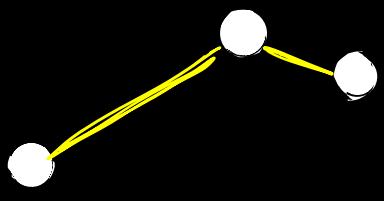

image

topological neighborhood

local neighborhood
= neighbors ?

y
x

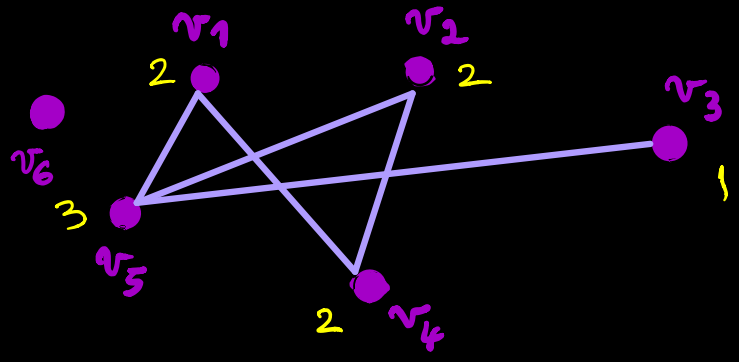
Spatial neighborhood



## Degree of a node

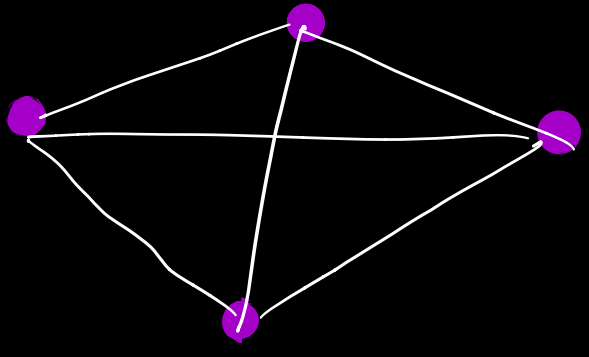
- Let $G = (V, E)$ be a (non-empty) graph. The set of neighbors of a node $v$ in $G$ is denoted by $N_G(v)$ or briefly $N(v)$.

  The degree (or valency) $d_G(v) = d(v)$ of a node $v$ is the number of edges at $v$. $\Rightarrow$ equal to the neighbors of $v$.




- A node of degree 0 is isolated.
- The number $\delta(G) = \min \{ d(v) \mid v \in V \}$ → minimum degree of $G$.
- The number $\Delta(G) = \max \{ d(v) \mid v \in V \}$ → maximum degree of $G$.
- If all nodes in $G$ have the same degree $k$, then $G$ is $k$-regular, or simply regular.

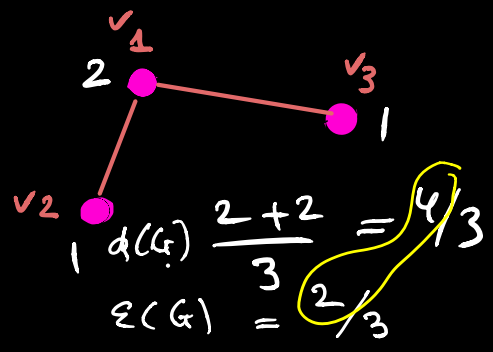
☺ Let's build a 3-regular graph (cubic graph):



3-regular complete

- The average degree of a graph $G$:

$$d(G) = \frac{1}{|V|} \sum_{v \in V} d(v)$$




- Clearly $\delta(G) \leq d(G) \leq \Delta(G)$
- The average degree quantifies globally what is measured locally by the node degrees: the number of edges of $G$ per node.
- Sometimes it is more convinient to express this ratio directly:

$$\varepsilon(G) = \frac{|E|}{|V|} = \frac{\|G\|}{|G|}$$

{ what is the relationship between $d(G)$ and $\varepsilon(G)$?

---

- If we sum up all the node degrees in $G$, we count every edge exactly twice:

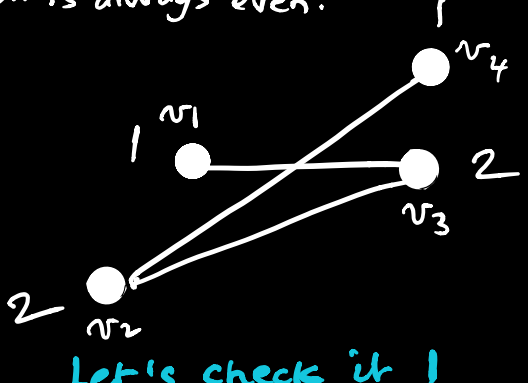
$$|E| = \frac{1}{2} \sum_{v \in V} d(v) = \frac{1}{2}\, d(G)\, |V|$$

$$\Rightarrow \quad \varepsilon(G) = \frac{|E|}{|V|} = \frac{1}{2}\, d(G)$$

## PROPOSITION 1.1

The number of nodes of odd degree in a graph is always even.

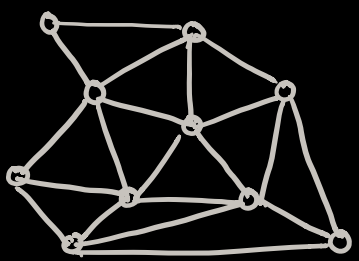
Proof ( so simple ! ) $|E| = \frac{1}{2} \sum_{v \in V} d(v)$ ($d(v)$: even)

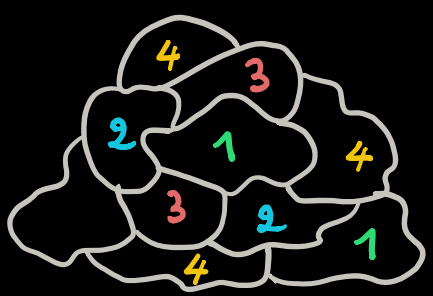


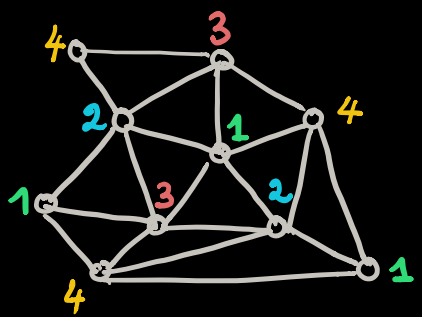

Let's check it !

# PB: GRAPH COLORING

In a map consisting of regions, can the regions be colored with four or fewer colors in such a way that every two adjacent regions are colored differently?



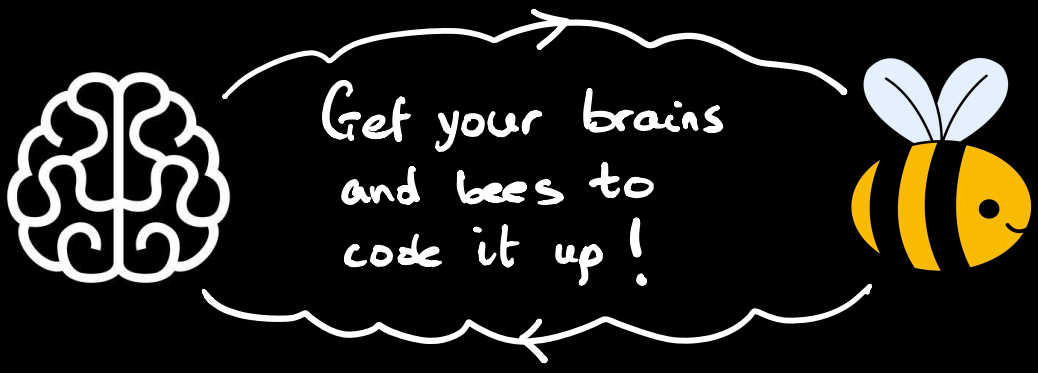



How to find the smallest number of colors K(G) such that all neighboring nodes in G have different colors ? → Design an algo which uses node degree to find K(G)?

↓

chromatic number




By *Jure Leskovec* STANFORD UNIVERSITY


- SNAP for C++
- SNAP for Python
- SNAP Datasets
- BIOSNAP Datasets
- What's new
- People
- Papers
- Projects
- Citing SNAP
- Links
- About
- Contact us

**Open positions**

Open research positions in **SNAP** group are available here.

# SNAP System

**S**tanford **N**etwork **A**nalysis **P**latform (**SNAP**) is a general purpose, high performance system for analysis and manipulation of large networks. *Graphs* consists of nodes and directed/undirected/multiple edges between the graph nodes. *Networks* are graphs with data on nodes and/or edges of the network.

The core SNAP library is written in C++ and optimized for maximum performance and compact graph representation. It easily scales to massive networks with hundreds of millions of nodes, and billions of edges. It efficiently manipulates large graphs, calculates structural properties, generates regular and random graphs, and supports attributes on nodes and edges. Besides scalability to large graphs, an additional strength of SNAP is that nodes, edges and attributes in a graph or a network can be changed dynamically during the computation.

SNAP was originally developed by Jure Leskovec in the course of his PhD studies. The first release was made available in Nov, 2009. SNAP uses a general purpose STL (Standard Template Library)-like library GLib developed at Jozef Stefan Institute. SNAP and GLib are being actively developed and used in numerous academic and industrial projects.

## Download

### Download SNAP

Download SNAP **HERE**!

Complete source code for the core SNAP and GLib libraries is available under the BSD license.

SNAP works on Windows with Visual Studio or Cygwin with GCC, Mac OS X, Linux and other Unix variants with GCC installed. SNAP is largely self-contained and has minimal dependency requirements on other packages.

## User Documentation

### Quick Introduction

SNAP.STANFORD.EDU