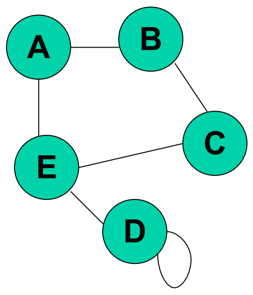
# COMP 102: Excursions in Computer Science
Lecture 11: Graphs

Instructor: Joelle Pineau (jpineau@cs.mcgill.ca)

Class web page: www.cs.mcgill.ca/~jpineau/comp102

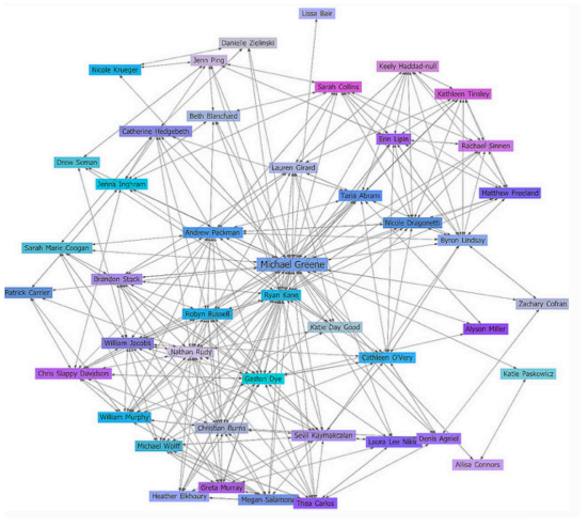
## Quick Review of Graphs

- A **graph** is an abstract representation defined by a pair **(N, E)**, where
  - **N** is a collection of nodes (or objects)
  - **E** is a collection of pairs of nodes, called edges (representing the relations between the objects.)
- What is a path? What is the path length?
- What is an adjacency matrix?
- What is the difference between directed and undirected graphs?
- What is a cycle? What is a tree?



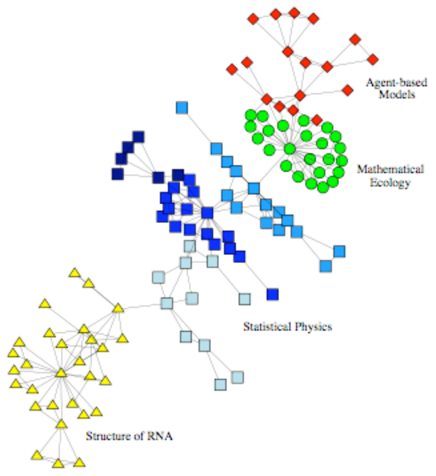
## Example: A friendship network

- Graphs are sometimes also called networks.
- Graph analysis tool on Facebook to analyze patterns of friendships.
  - Nodes: people
  - Edges: friendships
- Could annotate the types of relationships.



## Example: Scientific collaborations

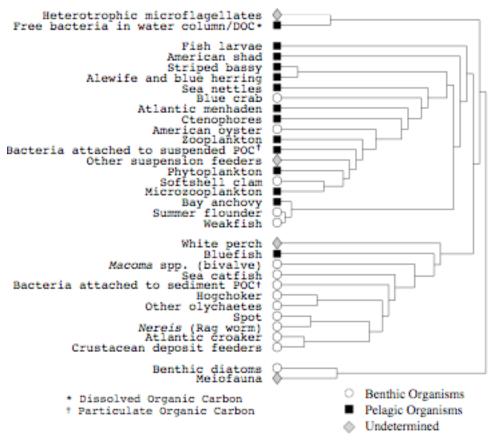
- Nodes correspond to scientists in residence at the Santa Fe Institute in 1999-2000, and their collaborators.
- An edge is drawn between a pair of scientists if they coauthored one or more articles during this time period.
- The research topics are shown as different colours. These are identified automatically using a *clustering* algorithm.



*http://arxiv.org/pdf/cond-mat/0112110v1*

## Example: Food web

- Nodes correspond to the most prevalent marine organisms living in the Chesapeake Bay (USA).
- An edge is drawn between a pair if one of the organisms eats the other.
- Graph suggests there are two well-defined communities.
- These correspond quite closely to pelagic organisms (those that live near the surface) and benthic organisms (those that live near the bottom).



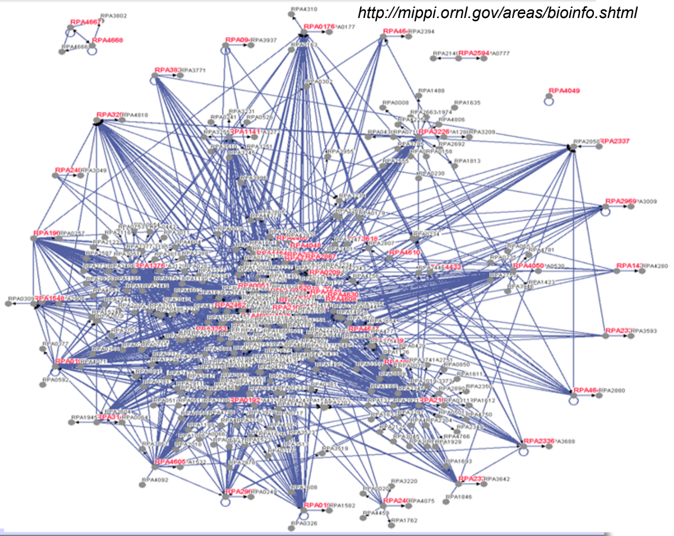
*http://arxiv.org/pdf/cond-mat/0112110v1*

## Example: Protein-protein interaction network

*http://mippi.ornl.gov/areas/bioinfo.shtml*

Interaction of protein molecules from the perspective of biochemistry and signal transduction.

E.g. *R. Palustris* protein-protein interaction network.

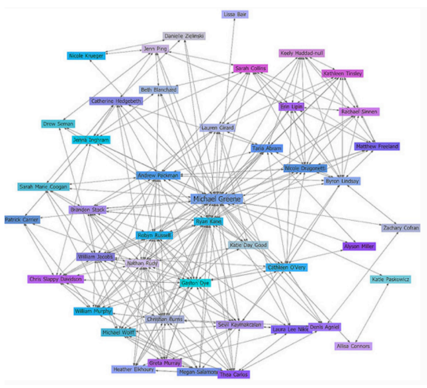


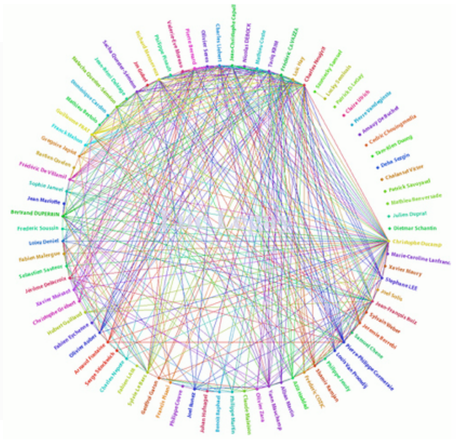
## Aside

- **Question**: How should we display a graph (nodes and edges) such that the information is interpretable for a human reader?
- This is the problem of graph **visualization**.
- This is a hard problem! Especially for graphs with many nodes and edges.

## Example: A friendship network

You can visualize the network differently, to see certain patterns.





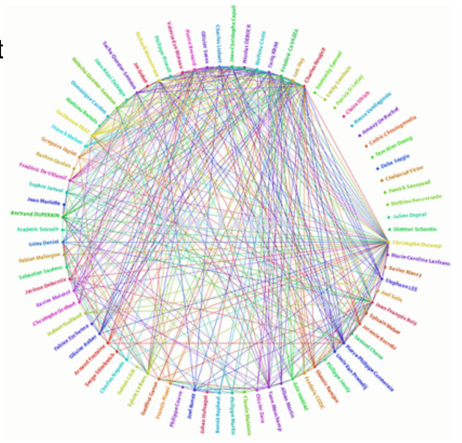
## Aside

- **Question**: How should we display a graph (nodes and edges) such that the information is interpretable for a human reader?
- This is the problem of graph **visualization**.
- This is a hard problem! Especially for graphs with many nodes and edges.
- Many people interested in finding good graph drawing algorithms to automatically generate an image of a graph, given its adjacency matrix.
- Nothing more on this problem in this course. But interesting techniques in graph theory and computer graphics for this.

## Back to work

So many questions we could ask of these graphs!

E.g. How do we find the shortest path between two given nodes?



Time to think about **SEARCHING**!

## Searching over Graphs

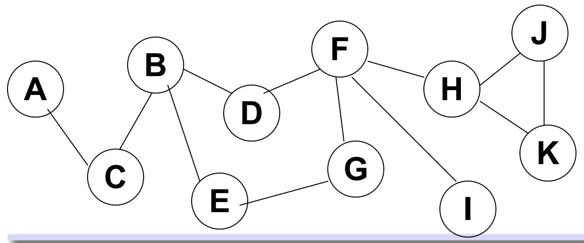
- Your graph is defined by a set of nodes and an adjacency matrix.
- You also need to know the start node and the end node.
- The goal is to explore all possible paths and return the shortest one.

Warning! Need to be **systematic** about the order in which you explore these paths.

## Breadth-first search

- Start at some node n. — Say we start with F.
- Explore all the neighbors of n. — Explore D, G, I and H.
- Then explore all the unvisited neighbours of the neighbours of n. — B, E, K, J
- Then visit unvisited neighbours of those. — C
- Continue until no more unvisited nodes remain. — A

| | |
|---|---|
| Visitation order: | F, D, G, I, H, B, E, K, J, C, A |
| Visitation path: | F - D - F - G - F - I - F - H - F - D - B - D - F - G - E - G - F - H - K - H - J - H - F - D - B - C - A |



## Comments on breadth-first search

- Breadth-first search explores the graph layer by layer.
  E.g. For web-browsing, all n-away links are explored.
  **IMPORTANT**:
  - Need to decide before-hand on the order of neighbours (e.g. clockwise)
  - Need to keep track of nodes you've already explored.

- **Pro**: Good algorithm if you want to find the shortest path between the start node, and another node. (As soon as you find that node, you know you have found the shortest path to it.)

- **Con**: Often requires a lot of backtracking ( = visitation path goes through visited nodes again and again.)

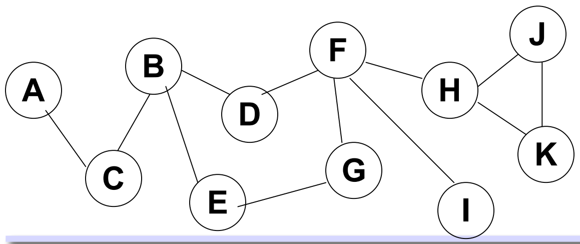
**Can we avoid all this backracking?**

## Depth-first search

- Start at some node n. Say we start with F.
- Then explore the first unvisited neighbour of n (call this n'). Explore D.
- Then explore the first unvisited neighbour n', and so on until you hit a node with no unexplored neighbours. B, C, A
- Then backtrack 1 level to explore the next unvisited neighbour. E, G, etc.

  Visitation order: F, D, B, C, A, E, G, I, H, K, J

  Visitation path: F - D - B - C - A - C - B - E - G - E - B - D - F - I - F - H - K - J



## Comments on depth-first search

- Depth-first search explores graph by going deeper whenever possible.

  E.g. For web-browsing, always click on 1st link until you hit a dead-end.

  **IMPORTANT**:
  - Need to decide before-hand on the order of neighbours (e.g. clockwise)
  - Need to keep track of nodes you've already explored.

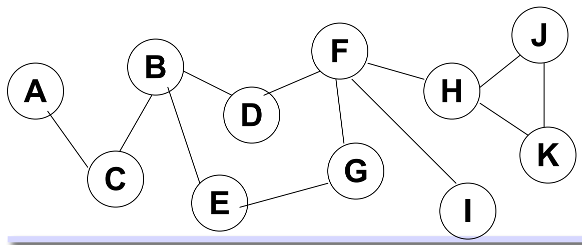
- **Pro**: Usually uses much less backtracking to explore the full graph than breadth-first search. How much less depends on neighbourhood ordering (sometimes lucky, sometimes not)

- **Con**: Not guaranteed to find the shortest path, unless you explore the full graph.
  - E.g. After 3 rounds, found path to "E": F-D-B-E, which is longer than F-G-E.

## Can we try a Best-first search?

| | |
|---|---|
| • Start at some node n. | Say we start with F. |
| • Pick a score function. | Say score = alphabetical order. |
| • Add its neighbours to the list of candidate nodes. | Add D(=4), G(=7), I(=9), H(=8). |
| • Pick candidate node with best score. | Pick D. |
| • Add its neighbours to the list of candidate nodes. | Add B(=2). |
| • Continue until no more unexplored nodes. | Pick B, Add C(=3) and E(=5), etc. |

| | |
|---|---|
| Exploration order: | F, D, B, C, A, E, G, H, I, J, K |
| Candidate list: | D, G, I, H, B, C, E, A, K, J |



## Comments on best-first search

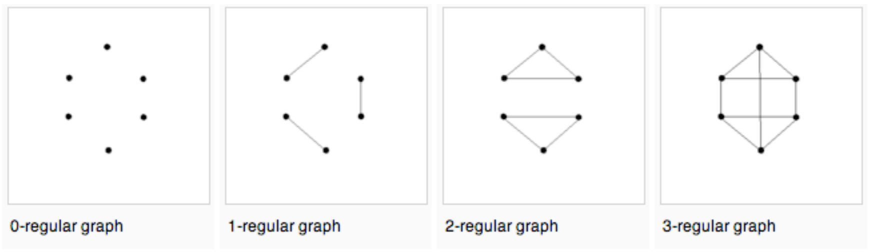
- Best-first search explores graph by according to priority order.
  E.g. For web-browsing, always explore link with highest PageRank.
  **IMPORTANT**:
  - Need to have a score function, which can be calculated for each node.
  - Need to keep track of candidate nodes.
- **Pro**: Usually much faster to reach a goal node (e.g. let's say we stop when we reach "A".)
- **Con**: No advantage if you want to explore the full graph.

## Graph Topologies

- **Topology** = The arrangement in which the nodes of a graph are connected to each other.
- Common types of graphs:
  - **Regular graph**
  - **Complete graph**
  - **Random graph**

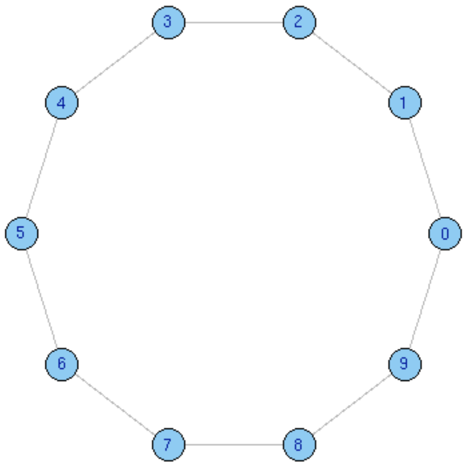
## Regular graph

- Main characteristic: Each node has same number of neighbours.



*http://en.wikipedia.org/wiki/Regular_graph*

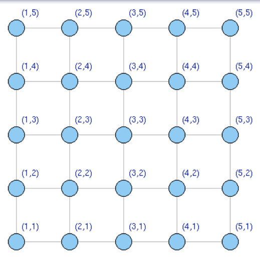
## Special regular graph: the Ring



*http://geza.kzoo.edu/~csardi/module/html/*

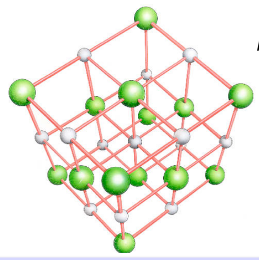
## Special regular graph: the Lattice

- This is a common topology to model road networks (in 2-D).



*http://geza.kzoo.edu/~csardi/module/html/*

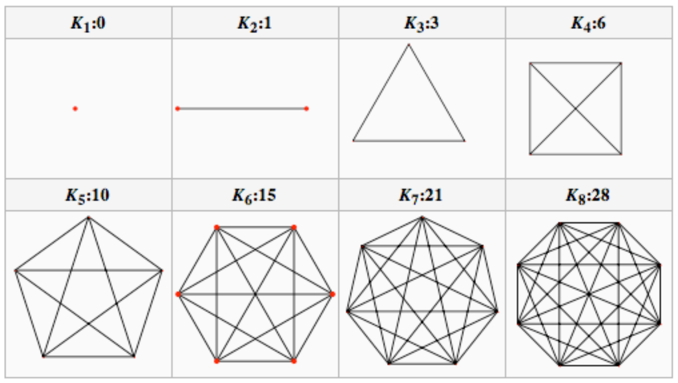
- Also common for molecular diagrams (in 3-D).



*http://www.dkimages.com*

## Complete graph (also a regular graph)

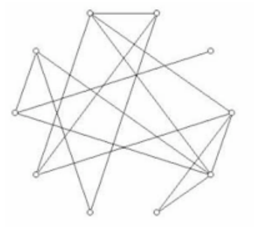
- Main characteristic: All pairs of nodes are connected by an edge.



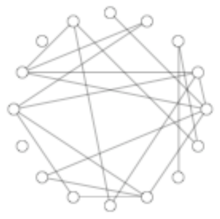
*http://en.wikipedia.org/wiki/Complete_graph*

## Random graph

- Basic construction: Start with a set of nodes. With probability *p,* randomly add an edge between any pair of nodes.
- Graph is denoted *G(n,p)*, where *n* is the number of nodes and *p* is the probability of a pairwise connection.



*http://epress.anu.edu.au/cs/html/ch05s03.html*



*http://aps.arxiv.org/PS_cache/cond-mat/pdf/0007/0007235v2.pdf*

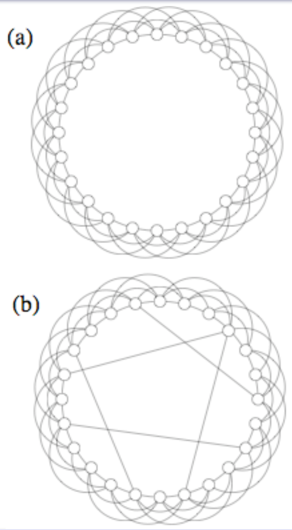
## Graphs in the real world

- Think back to our examples:
  - Montreal metro system.
  - Friendship networks.
  - Roadmap of a city.
  - The internet.
  - A maze.
- Most biological, technological and social graphs/networks are not exactly regular, complete or random.
- **Next**: explore a special class of graphs.

## Small-world networks

- A small-world network is a mix of a regular graph and a random graph.
- Simple construction:
  - Start with a ring made of *n* nodes and *k* edges per node.
  - Wire the *k* edges as for a regular graph.
  - With probability *p*, re-wire each edge to another random node.

*http://aps.arxiv.org/PS_cache/cond-mat/pdf/0303/0303516v1.pdf*



## Characteristics of a small-world network

- Key parameters:
  - *n* controls the size of the graph (= number of nodes)
  - *k* controls the degree of connectedness (e.g. if *k=n* then we have a complete graph.
  - *p* controls the trade-off between “regular” (*p=0*) and “random” (*p=1*)

**This correctly models many real-life networks!**

## Example: Model the spread of an infectious disease

Consider a population of *n* individuals, connected according to a given topology.

Basic model:

- On day 1: a number *b* of individuals are infected.
- On day 2 (and subsequent days)*:* we see the effect of that infection
  - Each infected individual can infect each of its neighbours with probability *h*.
  - Each infected individual is cured with probability *g*.

We could complicate this model significantly, e.g.

- Individuals have probability of dying from the disease.
- Individuals develop immunity so can't get the disease more than once.
- Individuals take a variable number of days to develop symptoms after contagion.

## Analysis of the model

Now we can ask lots of interesting questions!

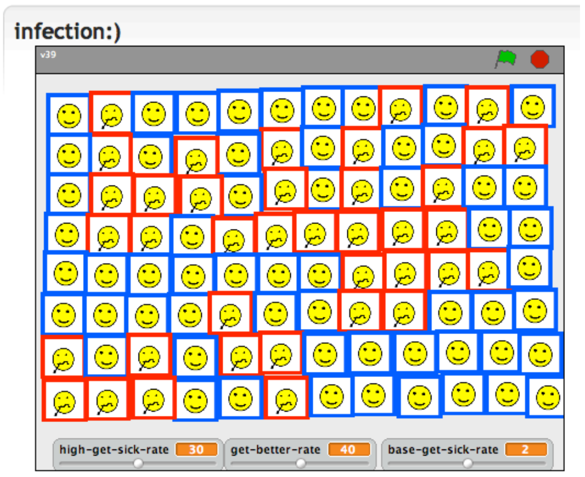
- For what values of infection rate *h* and remission probability *g* can we keep infection rate at less than 10% of the population?
- What is the critical base rate *b* at which the disease infects half of *n* in less than a week?
- What is the impact of the graph topology on the spread of a disease?
- What is the impact of a specific intervention strategy (e.g. through manipulating *h*) on the spread of the disease?

**How do we get these results?**

## Simulating a graph

- Need to simulate our graph, to capture the change of state in the infected population.
- What do you remember about finite-state machines?
  - States + Transition graph. Use this here!
- Pick values for *n*, *b*, *g*, and *h*.
- The state is described by a separate variable, $n_i$ = *{healthy, infected}* for each node.
- The transition graph expresses the effect of the infection.

## Scratch simulation



http://scratch.mit.edu/projects/zevbo/1372318

## Take-home message

- Main searching algorithms for graphs: Breadth-first search, Depth-first search, Best-first search.
  - Know the steps of each algorithm, and the pros/cons for each.
- Characteristics of the basic types of graphs (regular, complete, random).
- Definition and characteristics of small-world networks.
- Understand how graphs and finite-state machines can be combined to simulate real-world phenomena.