Graphs – Introduction (theory) – 1

MATH 2740 – Mathematics of Data Science – Lecture 15

Julien Arino

julien.arino@umanitoba.ca

Department of Mathematics @ University of Manitoba

Fall 202X

The University of Manitoba campuses are located on original lands of Anishinaabeg, Ininew, Anisininew, Dakota and Dene peoples, and on the National Homeland of the Red River Métis. We respect the Treaties that were made on these territories, we acknowledge the harms and mistakes of the past, and we dedicate ourselves to move forward in partnership with Indigenous communities in a spirit of Reconciliation and collaboration.

Outline

Why use graphs/networks?

Binary relations

Undirected graphs

Why use graphs/networks?

Binary relations

Undirected graphs

Graphs versus networks

Mostly a terminology difference:

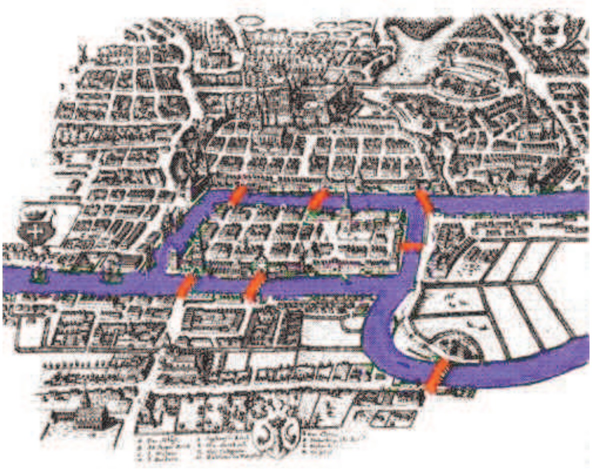
- graphs in the mathematical world
- networks elsewhere

I will mostly say *graphs* (this is a math course) but might oscillate

Beware: language is not consistent, so make sure you read the definitions at the start of whatever source you are using

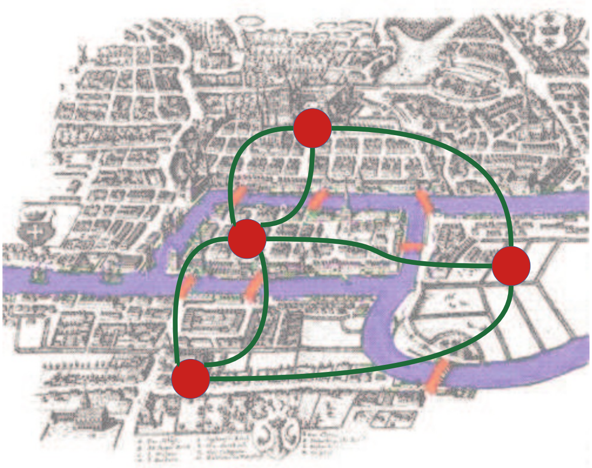
The genesis of graphs – Euler's bridges of Königsberg

Cross the 7 bridges in a single walk without recrossing any of them?



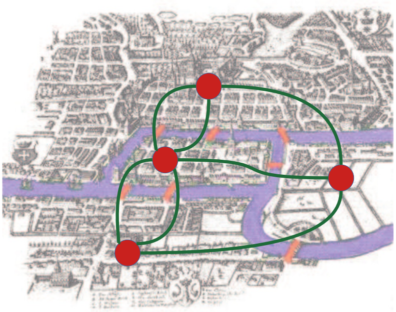
The genesis of graphs – Euler's bridges of Königsberg

Cross the 7 bridges in a single walk without recrossing any of them?



The genesis of graphs – Euler's bridges of Königsberg

Cross the 7 bridges in a single walk without recrossing any of them?

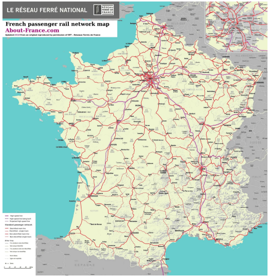


Mathematical problem

Is it possible to find a *trail* containing all *edges* of the graph?

Finding a cycle with all vertices

Salesperson must visit some cities (vertices) for their job. Can they plan a round trip using trains enabling them to visit each specified city exactly once?



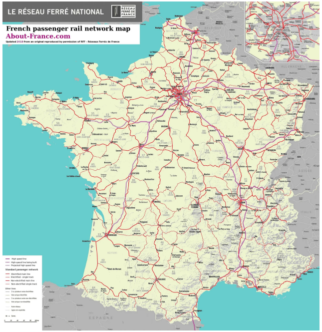
- 2 vertices are connected iff a line connects the cities and does not pass through any other city

Mathematical problem

Is it possible to find a cycle containing all graph vertices?

How far is it to “train” through *n* cities?

What is the minimal length of train travel needed to visit *n* cities (vertices)?



- all cities are connected; each edge has a value assigned to it (the distance)

Mathematical problem

What is the minimal spanning tree associated to the graph?

Graphs/networks encode relations

Graphs are used in a variety of contexts because they encode *relations* between objects

Many objects in the world have relations... so graphs are quite easy to find

We will see many examples later, for now we cover the mathematical background

Graphs vs digraphs vs multigraphs vs multidigraphs vs ...

Name-wise and notation-wise, this domain is a bit of a mess

- The vertex set V is essentially the only constant
- *Undirected graph* $G = (V, E)$, where E are the *edges*
- *Undirected multigraph* $G_M = (V, E)$
- *Directed graph* (or *digraph*) $G = (V, A)$, where A are the *arcs*
- *Directed multigraph* (or *multidigraph*) $G_M = (V, A)$
- Any of the above is called a *graph* and is denoted $G = (V, X)$, when we seek generality

And just to confuse the whole thing more: we often say *graph* for *unoriented graph*

Why use graphs/networks?

Binary relations

Undirected graphs

Binary relation

Definition 1 (Binary relation)

- A **binary relation** is an arbitrary association of elements of one set with elements of another (maybe the same) set
- A binary relation over the sets X and Y is defined as a subset of the Cartesian product $X \times Y = \{(x, y) | x \in X, y \in Y\}$
- $(x, y) \in R$ is read "x is R-related to y" and is denoted xRy
- If $(x, y) \notin R$, we write "not xRy" or $x \not{R} y$

Definition 2 (Properties of binary relations)

A binary relation R over a set X is

- **Reflexive** if $\forall x \in X$, xRx
- **Irreflexive** if there does not exist $x \in X$ such that xRx
- **Symmetric** if $xRy \Rightarrow yRx$
- **Asymmetric** if $xRy \Rightarrow y \not R x$
- **Antisymmetric** if xRy and $yRx \Rightarrow x = y$
- **Transitive** if xRy and $yRz \Rightarrow xRz$
- **Total** (or **complete**) if $\forall x, y \in X$, xRy or yRx

Definition 3 (Equivalence relation)

A relation that is reflexive ($\forall x \in X$, xRx), symmetric ($xRy \Rightarrow yRx$) and transitive (xRy and $yRz \Rightarrow xRz$) is an **equivalence relation**

Definition 4 (Partial order)

A relation that is reflexive ($\forall x \in X$, xRx), antisymmetric (xRy and $yRx \Rightarrow x = y$) and transitive (xRy and $yRz \Rightarrow xRz$) is a **partial order**

Definition 5 (Total order)

A partial order that is total ($\forall x, y \in X$, xRy or yRx) is a **total order**

- Why use graphs/networks?
- Binary relations
- **Undirected graphs**

The `igraph` library

Throughout these slides, we use the package `igraph`

I illustrate the functions that can be used to study some of the mathematical notions I introduce

I use mostly examples from the `igraph` documentation

Undirected graphs

- **Undirected graph**
- Degree of a vertex
- Isomorphic graphs
- Subgraphs, unions of graphs

Graph

Intuitively: a graph is a set of points, and a set of relations between the points

The points are called the *vertices* of the graph and the relations are the *edges* of the graph

We can also think of the relations as being one directional, in which case the relations are the *arcs* of the digraph (a contraction of “directed graph”)

Graph, vertex and edge

Definition 6 (Graph)

An **undirected graph** is a pair $G = (V, E)$ of sets such that

- V is a set of points: $V = \{v_1, \ldots, v_p\}$
- E is a set of 2-element subsets of V: $E = \{\{v_i, v_j\}, \{v_i, v_k\}, \ldots, \{v_n, v_p\}\}$ or $E = \{v_i v_j, v_i v_k, \ldots, v_n v_p\}$

Definition 7 (Vertex)

The elements of V are the **vertices** (or nodes, or points) of the graph G. V (or $V(G)$) is the vertex set of the graph G

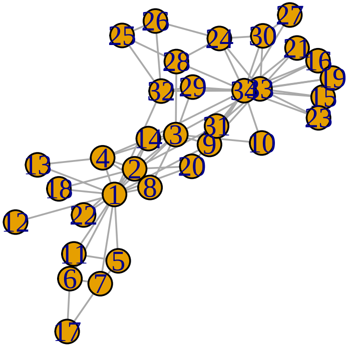
Definition 8 (Edge)

The elements of E are the **edges** (or lines) of the graph G. E (or $E(G)$) is the edge set of the graph G

Setting up graphs in `igraph`

```r
G <- make_empty_graph()
G <- make_graph(edges = c(1, 2, 1, 5), n = 10,
                directed = FALSE)
G <- make_graph("Zachary")
plot(G, main = "Data on a Karate club")
```

Data on a Karate club



Order and Size

Definition 9 (Order of a graph)

The number of vertices in G is the **order** of G. Using the notation $|V(G)|$ for the *cardinality* of $V(G)$,

$$|V(G)| = \text{order of G}$$

Definition 10 (Size of a graph)

The number of edges in G is the **size** of G,

$$|E(G)| = \text{size of G}$$

- A graph having order p and size q is called a $(p, q)-$graph
- A graph is finite if $|V(G)| < \infty$

Some simple measures

```
V(G)

## + 34/34 vertices, from b79af2b:
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 2
## [26] 26 27 28 29 30 31 32 33 34

head(E(G))

## + 6/78 edges from b79af2b:
## [1] 1--2 1--3 1--4 1--5 1--6 1--7

gorder(G)

## [1] 34

gsize(G)

## [1] 78
```

Incident – Adjacent

Definition 11 (Incident)

- A vertex v is **incident** with an edge e if $v \in e$; then e is an edge at v
- If $e = uv \in E(G)$, then u and v are each incident with e
- The two vertices incident with an edge are its ends
- An edge $e = uv$ is incident with both vertices u and v

Definition 12 (Adjacent)

- Two vertices u and v are **adjacent** in a graph G if $uv \in E(G)$
- If uv and uw are distinct edges (i.e. $v \neq w$) of a graph G, then uv and uw are adjacent edges

Incident vertices

```
incident(G, 1)

## + 16/78 edges from b79af2b:
##  [1] 1-- 2 1-- 3 1-- 4 1-- 5 1-- 6 1-- 7 1-- 8 1-- 9 1--11 1--12 1--13 1-
## [13] 1--18 1--20 1--22 1--32

incident_edges(G, c(1, 2))

## [[1]]
## + 16/78 edges from b79af2b:
##  [1] 1-- 2 1-- 3 1-- 4 1-- 5 1-- 6 1-- 7 1-- 8 1-- 9 1--11 1--12 1--13 1-
## [13] 1--18 1--20 1--22 1--32
##
## [[2]]
## + 9/78 edges from b79af2b:
## [1] 1-- 2 2-- 3 2-- 4 2-- 8 2--14 2--18 2--20 2--22 2--31
```

Adjacent vertices

```r
adjacent_vertices(G, v = 1)
```

```
## [[1]]
## + 16/34 vertices, from b79af2b:
##  [1]  2  3  4  5  6  7  8  9 11 12 13 14 18 20 22 32
```

```r
adjacent_vertices(G, v = c(1, 2))
```

```
## [[1]]
## + 16/34 vertices, from b79af2b:
##  [1]  2  3  4  5  6  7  8  9 11 12 13 14 18 20 22 32
##
## [[2]]
## + 9/34 vertices, from b79af2b:
## [1]  1  3  4  8 14 18 20 22 31
```

Definition 13 (Multiple edge)

Multiple edges are two or more edges connecting the same two vertices within a multigraph

Definition 14 (Loop)

A **loop** is an edge with both the same ends; *e.g.* $\{u, u\}$ is a loop

Definition 15 (Simple graph)

A **simple graph** is a graph which contains no loops or multiple edges

Definition 16 (Multigraph)

A **multigraph** is a graph which can contain multiple edges or loops

Testing for these properties

```
any_multiple(G)

## [1] FALSE

any_loop(G)

## [1] FALSE

is_simple(G)

## [1] TRUE
```

Graph and binary relations

A simple graph G can be defined in term of a vertex set V and a binary relation over V that is

- irreflexive ($\forall u \in V, u \not R u$)
- symmetric ($\forall u, v \in V, uRv \implies vRu$)

The set of edges $E(G)$ is the set of symmetric pairs in R

If R is not irreflexive, the graph is not simple

Undirected graphs

- Undirected graph
- **Degree of a vertex**
- Isomorphic graphs
- Subgraphs, unions of graphs

Definition 17 (Degree of a vertex)

Let v be a vertex of $G = (V, E)$.

- The number of edges of G incident with v is the **degree** of v in G
- The number of edges of G at v is the **degree** of v in G
- The degree of v in G is noted $d_G(v)$ or $deg_G(v)$

Theorem 18

Let G be a $(p, q)-$graph with vertices $v_1, \ldots, v_p$, then

$$\sum_{i=1}^{p} d_G(v_i) = 2q$$

Degree

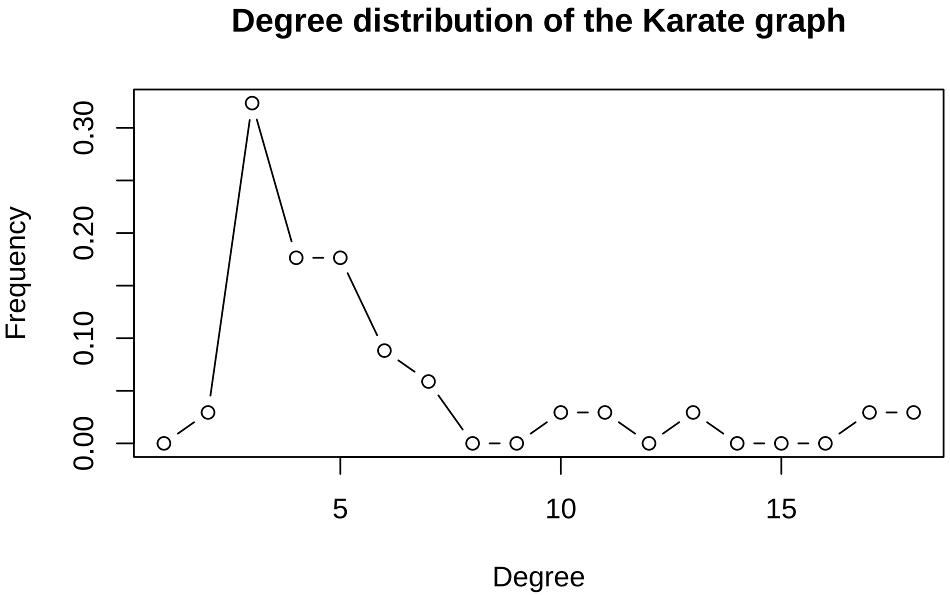
```r
degree(G)
```

```
##  [1] 16  9 10  6  3  4  4  4  5  2  3  1  2  5  2  2  2  2  2  3  2  2  2
## [26]  3  2  4  3  4  4  6 12 17
```

```r
degree_distribution(G)
```

```
##  [1] 0.00000000 0.02941176 0.32352941 0.17647059 0.17647059 0.08823529
##  [7] 0.05882353 0.00000000 0.00000000 0.02941176 0.02941176 0.00000000
## [13] 0.02941176 0.00000000 0.00000000 0.00000000 0.02941176 0.02941176
```

```r
plot(degree_distribution(G),
     type = "b",
     xlab = "Degree", ylab = "Frequency",
     main = "Degree distribution of the Karate graph")
```



Degree distribution of the Karate graph
Frequency
0.00
0.10
0.20
0.30
5
10
15
Degree

Definition 19 (Odd vertex)

A vertex is an **odd vertex** is its degree is odd

Definition 20 (Even vertex)

A vertex is called **even vertex** is its degree is even

Theorem 21

Every graph contains an even number of odd vertices

Illustration of recent results

Theorem 18 states that a $(p, q)-$graph has $\sum_{i=1}^{p} d_G(v_i) = 2q$

```
sum(degree(G)) == 2*length(E(G))

## [1] TRUE
```

Theorem 21 states that every graph contains an even number of odd vertices

```
sum(pracma::mod(degree(G),2))

## [1] 12
```

(`mode(x,2)` returns 1 if x is odd so `sum` counts how many are odd)

Regular graph

Definition 22 (Regular graph)

If all the vertices of G have the same degree k, then the graph G is k-regular

```
length(unique(degree(G))) == 1
```

```
## [1] FALSE
```

Undirected graphs

- Undirected graph
- Degree of a vertex
- **Isomorphic graphs**
- Subgraphs, unions of graphs

Isomorphic graphs

Definition 23 (Isomorphic graphs)

Let $G_1 = (V(G_1), E(G_1))$ and $G_2 = (V(G_2), E(G_2))$ be two graphs. G_1 and G_2 are **isomorphic** if there exists an isomorphism ϕ from G_1 to G_2, that is defined as an injective mapping $\phi : \; V(G_1) \to V(G_2)$ such that two vertices u_1 and v_1 are adjacent in $G_1 \iff$ the vertices $\phi(u_1)$ and $\phi(v_1)$ are adjacent in G_2

If ϕ is an isomorphism from G_1 to G_2, then the inverse mapping ϕ^{-1} from $V(G_2)$ to $V(G_1)$ also satisfies the definition of an isomorphism. As a consequence, if G_1 and G_2 are isomorphic graphs, then

- G_1 is isomorphic to G_2
- G_2 is isomorphic to G_1

Theorem 24

The relation "is isomorphic to" is an equivalence relation on the set of all graphs

Theorem 25

If G_1 and G_2 are isomorphic graphs, then the degrees of vertices of G_1 are exactly the degrees of vertices of G_2

Testing isomorphicity

Create two isomorphic graphs by permuting the vertices of the first, then test if they are isomorphic

```
g1 <- sample_pa(30, m = 2, directed = FALSE)
g2 <- permute(g1, sample(vcount(g1)))
# should be TRUE
isomorphic(g1, g2)

## [1] TRUE
```

Undirected graphs

- Undirected graph
- Degree of a vertex
- Isomorphic graphs
- Subgraphs, unions of graphs

Subgraph

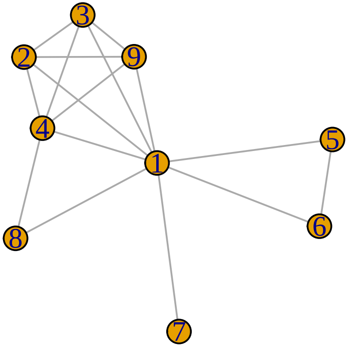
Definition 26 (Subgraph)

Let $G = (V, E)$ be a graph. A graph $H = (V(H), E(H))$ is a **subgraph** of G if $V(H) \subseteq V$ and $E(H) \subseteq E$

Extracting a subgraph

```
G_sub = induced_subgraph(G, c(1:5, 11:14))
plot(G_sub, main = "A subgraph of the Karate graph")
```

A subgraph of the Karate graph



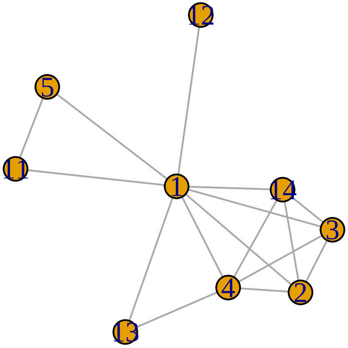
Naming vertices

Note that vertices (and as a consequence, edges) will be relabelled, so `G_sub` has vertices labelled 1 to 9

You can name vertices if you want to avoid this

```
V(G)$name = 1:length(V(G))
G_sub = induced_subgraph(G, c(1:5, 11:14))
plot(G_sub, labels = V(G)$name,
     main = "Subgraph of the Karate graph preserving names")
```

Subgraph of the Karate graph preserving names



Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs

Definition 27 (Union of G_1 and G_2)

$G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$

Definition 28 (Intersection of G_1 and G_2)

$G_1 \cap G_2 = (V_1 \cap V_2, E_1 \cap E_2)$

Definition 29 (Disjoint graphs)

If $G_1 \cap G_2 = (\emptyset, \emptyset) = \emptyset$ (empty graph) then G_1 and G_2 are **disjoint**

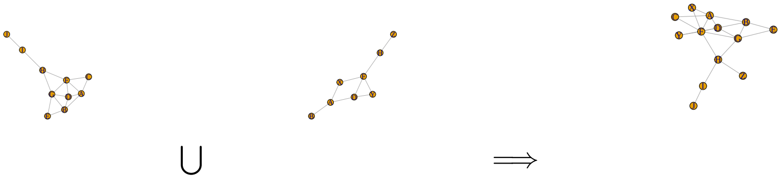
Definition 30 (Complement of G_1)

The **complement** $\bar{G}_1$ of G_1 is the graph on V_1, with the edge set $E(\bar{G}_1) = [V_1]^2 \setminus E_1$ ($e \in E(\bar{G}_1) \iff e \notin E_1$)

Union

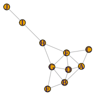
```
net1 <- graph_from_literal(
  D - A:B:F:G, A - C - F - A, B - E - G - B, A - B, F - G,
  H - F:G, H - I - J
)
net2 <- graph_from_literal(D - A:F:Y, B - A - X - F - H - Z, F - Y)
print_all(net1 %u% net2)

## IGRAPH 9551db4 UN-- 13 21 --
## + attr: name (v/c)
## + edges from 9551db4 (vertex names):
##  [1] I--J H--Z H--I G--H G--E F--X F--Y F--H F--C F--G B--E B--G A--X A--
## [16] A--B D--Y D--G D--F D--B D--A
```

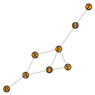


Intersection of two graphs

```
G_inter = net1 %s% net2
plot(G_inter)
```



$\cap$



$\Longrightarrow$