CS 408 Graph Theory

Administrative details: www.cse.iitb.ac.in/∼ranade/408

Abhiram Ranade

January 7, 2021

Introduction

Graph theory is the study of **pairwise relationships** between **entities**.
Graph = (V,E), where
V = set of **vertices**, "Entities"
E = set of **edges**, edge = pair of vertices. $(u, v) \in E$: "entities u, v are related"

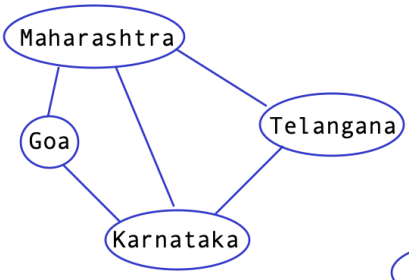
Undirected graph: Edge = unordered pair symmetric relationship
Directed Graph: Edge = ordered pair asymmetric relationship

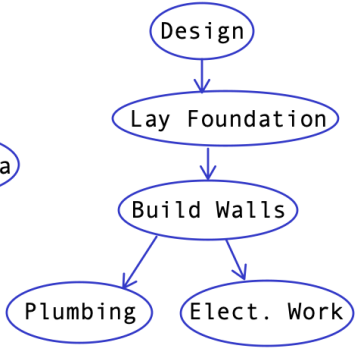
Example of Undirected Graph Relationship: States sharing a border
V = {Maharashtra, Goa, Karnataka, Telangana}
E = {(Mah.,Goa), (Mah., Tel.), (Mah., Kar.), (Kar.,Tel.), (Kar., Goa)}

Example of Directed Graph Relationship: Precedence between activities
V = {Design, Lay foundation, Build walls, Plumbing, Electrical work}
E = {(Design, Lay F.), (Lay F., Bld walls), (Bld walls, Plmb.), (Bld walls, Elect.)}

Pictorial representation of graphs

Vertices: Circles with name inside circle,
Undirected Edge (u, v) : line joining circles corresponding to u, v.
Directed edge (u, v) : Arrow from circle for u to circle for v.



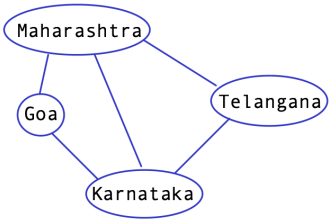


Many pictures possible for the same graph: connections important, not geometric placement.

Remarks

- Unless explicitly mentioned, "graph" means "undirected graph".
  Sometimes we allow edgeset E to be a multiset, i.e. the same pair may be listed several times. e.g. Representing two roads connecting the same pair of towns.
  We will call such graphs **"multigraphs"**.
- Unless explicitly mentioned, the two vertices in an edge need to be distinct.
  Occasionally we will allow an edge to connect the same vertex.
  Such an edge, say (u, u) is called a **(self) loop**.
- If $e = (u, v)$ is an edge, then u, v are said to be **adjacent**, and **endpoints** of e.
- An edge (u, v) is said to be **incident** on vertices u, v.
- If (u, v) is an edge in a (directed) graph, then the edge is said to be "from u to v".
- More examples of graphs
  - Transportation Networks
  - Family Trees
  - Organizational diagrams, e.g. who is whose boss

Graph Theoretic questions 1



When we make a map, we typically colour each region.

Regions sharing a border must get a different colour.
e.g. Maharashtra, Karnataka should get different colour.

What is the fewest number of colours needed to colour any map?

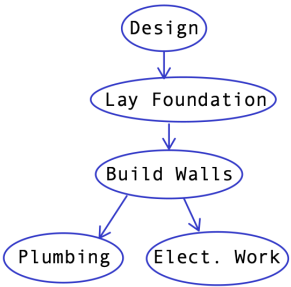
Instead of colouring the map, we can think of assigning colours to vertices in the graph, s.t. adjacent vertices have different colours.

Theorem: Any map that can be drawn on the surface of the earth, or the vertices of the associated graph, can be coloured using 4 colours.
Proof: Deep and difficult. But we will prove 5 colours suffice.

Our graph can be coloured using three colours. Mah: red, Kar: green, Kar,Tel: Blue.

Graph colouring models other real life problems too..

Graph Theoretic Questions 2



Suppose each vertex has an associated "duration".

Design: 30 days, Foundation: 5 days, Walls: 4 days, Plumbing: 2 days, Elect.: 2 days.

What is the minimum amount of time needed to complete all activities assuming enough people are available?

Note: If graph has edge (u, v) then v can start only after u finishes.

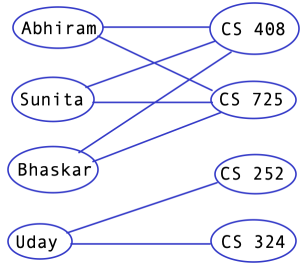
41 days for our graph

Possibly covered in Data Structures and Algorithms.

Graph Theoretic Questions 3

Each CSE professor says which courses she is willing to teach.
Goal: Maximize courses offered s.t. each professor teaches at most 1 course.

Vertices: One vertex for each course and One vertex for each professor.
Edges: Edge (u, v) is present if prof u can teach course v.



Max offered = 3

Goal in graph theoretic language: Select maximum number of edges such that at most one selected edge is incident on any vertex.

Such a collection of edges is called a **Matching**.

We are asking for a **Maximum (sized) Matching.**

Pause point

1. Suppose you are given information about students registered for different courses. You need to select examination slots for each course such that if two courses have a common student in their registrations then they must be scheduled in different slots. What is the minimum number of slots needed?
2. For every student you are given information about which pairs of students are willing to share rooms with. Assuming every room can accommodate only two students, how do you accommodate all students using the minimum number of rooms?

Solution:

1. Form a graph with a vertex for each course. Put an edge if the corresponding students share students. Find the minimum number of colours needed to colour this graph.
2. Form a graph with a vertex for each student, and edges (u, v) if students u, v are willing to share rooms. Find the maximum matching; allocate a room to each matched pair and single rooms to the rest.

Course Goals:

Understand how to express real life problems in the language of graphs.
Understand how to solve the problems.
Understand important graph properties.
Understand how to reason about graphs.

Arguments based on counting, mathematical induction.

Arguments based on matrices associated with graphs, their eigenvalues.

Rest of this lecture

Königsberg Bridge Problem

"Birth of Graph Theory"
Relevant in other problems, e.g. genome assembly

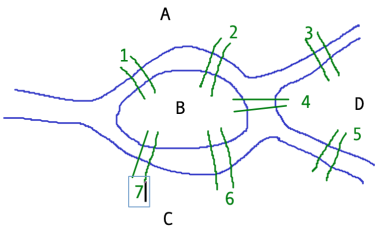
Formal Statement of the problem

Some terminology

Solution of the problem due to Euler.

The Königsberg Bridge Problem

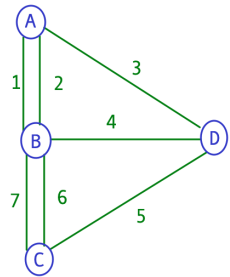
Königsberg is the old name of a city in Russia. It is on a river with islands and 7 bridges.



Puzzle: "Is it possible to start in any region A, B, C, D, cross each bridge exactly once and return where you started?"

Leonhard Euler solved this and its generalization in 1736. "Birth of graph theory"

Graph theoretic statement:



Vertex $\equiv$ Region. Edge $\equiv$ Bridge

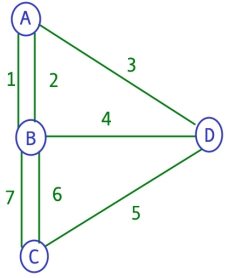
Connections important, not Geometry

"Start at any vertex and walk along each edge exactly once and return to the starting vertex."

Parallel edges : Multigraph

Some definitions and a theorem (Apply to Graphs/Multigraphs)

Degree of a vertex: number of edges incident on the vertex.



Degree(A) $= 3$
Degree(B) $= 5$

Even/odd vertex: Vertex with even/odd degree.
$\leftarrow$ all odd

Even graph: having all even vertices.

Walk: A sequence $v_1, e_1, v_2, e_2, \ldots, v_n$ s.t. e_i is incident on v_i, v_{i+1}
e.g. $A, 1, B, 7, C, 6, B, 6, C, 5, D$

Trail: Walk in which no edges repeat.

Path: Trail in which no vertex repeats.

u, v walk, trail, path A walk, path, trail in which first vertex is u, last is v.
A u, v walk, trail is **closed** if $u = v$.

A multigraph in which there is a u, v path for all vertices u, v is said to be **connected**.

Say a multigraph has a trail which passes through every edge and returns to the starting vertex. The multigraph and the trail are both called **Eulerian**.

Theorem: A connected multigraph is Eulerian if and only if it is even. Proof: Next.

Königsberg: not possible.

The necessary condition:

Theorem: If a connected multigraph is Eulerian, then it is even.

Proof: Let the multigraph have a trail $v_1, e_1, v_2, \ldots, e_{n-1}, v_n = v_1$, containing every edge exactly once.

The trail leaves v_1, then some k times returns to v_1 and leaves, and finally returns.
On each entry and exit it uses a unique edge.
So overall it must use an even number $(2k + 2)$ of edges.
Since all edges get used, the degree of v_1 must be $2k + 2$, i.e. even.

For any other vertex v: the trail enters and leaves v a total of some k' times.
So similarly the degree of v must be even. □

This proof is acceptable in the course.
But it is not very formal: entry, exit are not defined.
It is good to know how to prove more formally.

Next

On formal proofs

Day to day terms are analogies, e.g. a trail "enters".

Analogies help us in thinking.

But analogies are often not perfect. When we use them we need to be aware when they match the situation and where they break down.

If you want to be very careful: define terms precisely, then use. Formal proof

Key idea: Make up precise definitions guided by the analogies.

Think of the graph as being revealed as you walk along the trail.

Define the sequence of graphs that you will see, and consider how the degrees evolve.

The necessary condition: A more formal proof

Theorem: Suppose a connected multigraph G has a trail $T = v_1, e_1, v_2, \ldots, e_{n-1}, v_n$ with $v_n = v_1$, containing every edge exactly once. Then G must be even.

We make a claim about how G "is revealed". Then prove it by induction.

Proof for vertex v_1:

$H(k)$: In $G_k = (V, \{e_1, \ldots, e_k\})$, if $v_{k+1} = v_1$ then v_1 is even, else odd.

$H(n-1)$: In $G_{n-1} = (V, \{e_1, \ldots, e_{n-1}\})$, if $v_n = v_1$ then v_1 is even ... $\Rightarrow$ In G, v_1 is even.

Base case $H(1)$: In $G_1 = (V, \{e_1\})$ if $v_1 = v_2$ then v_1 is even, else odd.

Degree$(v_1) = 1$, odd. $v_1 \neq v_2$, So $H(1)$ holds.

Induction step: Suppose $H(k)$ holds for $k < n-1$.

Reqd: $H(k+1)$: In $G_{k+1} = (V, \{e_1, \ldots, e_{k+1}\})$, if $v_1 = v_{k+2}$, then v_1 is even, else odd.

Relevant part of T: $v_1, e_1, \ldots, v_k, e_k, v_{k+1}, e_{k+1}, v_{k+2}, \ldots$

Case 1: $H(k)$ **holds with** $v_{k+1} = v_1 \Rightarrow v_1$ is even in G_k. e_{k+1} is incident on $v_1 \Rightarrow v_1$ is odd in G_{k+1}. e_{k+1} is also incident on $v_{k+2} \neq v_{k+1} = v_1$. So $H(k+1)$ holds.

Case 2: $H(k)$ **holds with** $v_{k+1} \neq v_1 \Rightarrow v_1$ is odd in G_k.

If $v_{k+2} = v_1$, then degree of v_1 increases in G_{k+1} and becomes even. So $H(k+1)$ holds.

If $v_{k+2} \neq v_1$, then degree of v_1 in G_{k+1} remains odd. So $H(k+1)$ holds.

Proof for other vertices

Similar. Exercise.

Some more notation

We will write E(G) or V(G) to denote the edges of vertices of G.

If G has a u, v path, then u, v are said to be **connected**.
Set of all vertices connected to each other is a **connected component**.

Sufficiency

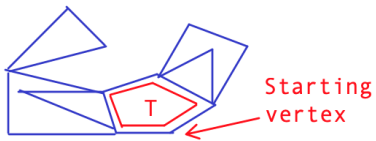
Theorem: If G is connected and even, then it is Eulerian.

Proof: Here is a recursive procedure to construct the required trail.

makeTrail($G = (V, E)$){

1. If $|E| = 0$ return ϕ.
2. Let T = trail starting from any vertex extended edge at a time while possible.
3. $G' = (V, E - E(T))$
4. for each component C of G'
5. T = merge(T, makeTrail(C))
6. Return T.

}



Precondition: G is even.
Postcondition: Closed trail returned.
Stmt 1 is the base case.

T must end at the starting vertex; if it ends at $v \neq$ starting vertex, v will need to have odd degree.

G' is even, because degree of each vertex reduces by an even number when we remove edges in T.

Each C is even, so makeTrail(C) will return a trail passing through all edges of C at returning to start.

"Magic of recursion"

Each C shares a vertex v with T. So makeTrail(C) can be merged with T at that vertex.

Pause point

1. Prove that a closed trail T contains an even number of edges incident on any vertex.
2. Suppose G' is obtained by removing the edges of a closed trail T from a connected even graph G Suppose C is a connected component in G'. Show that C, T share a vertex.
3. Suppose two closed trails have a common vertex v. Show that they can be merged into a single closed trail through all the vertices on the two trails.
4. Did we prove that our recursive procedure is correct?

1. Proof: Each edge on which the trail enters can be paired with an edge on which it leaves.
2. Let u, v be vertices in C, T resp. Since G is connected there is a u, v path P in G. Let w be the first vertex in P that is in T. If $w = u$ we are done; so assume otherwise. Now the edge e preceding w cannot belong to a different component C' of G' because then C, C' would be the same component. So e must be in C. So w must be in C as well.
3. Suppose v is the common vertex. Then start at v travel through the first, return to v, then travel through the second and return to v.
4. We showed that in successive recursive calls the arguments satisfies the precondition, and the number of edges decrease so the procedure will terminate. It should be clear that the top level call will be correct if the recursive calls return correctly. So proved.

Exercises

1. Prove that the number of odd vertices in any graph must be even.
2. Suppose 100 aspirants apply for 10 job positions. If each aspirant sends out 10 applications on the average, how many applications are there for any position?
3. Suppose a certain connected graph has a trail passing through all edges but not returning to the same vertex. What can you say about such graphs?
4. The **complement** of a graph $G = (V, E)$ is a graph $G' = (V, E')$ where E' contains exactly those pairs (u, v) such that $(u, v) \notin E$. Suppose a graph G is not connected. Prove that its complement will be connected or give a counter example.