Contents

Chapter 1

Introduction

The first chapter is an introduction, including the formal definition of a graph and many terms we will use throughout, but more importantly, examples of these concepts and how you should think abotu them.

1.1 A first look a graphs

1.1.1 The idea of a graph

First and foremost, you should think of a graph as a certain type of picture, containing dots and lines connecting those dots, like so:

We will typically use the letters G, H, or Γ (capital Gamma) to denote a graph. The "dots" or the graph are called *vertices* or *nodes*, and the lines between the dots are called *edges*. Graphs occur frequently in the "real world", and typically how to show how something is connected, with the vertices representing the things and the edges showing connections.

- *Transit networks:* The London tube map is a graph, with the vertices representing the stations, and an edge between two stations if the tube goes directly between them. More generally, rail maps in general are graphs, with vertices stations and edges representing line, and road maps as well, with vertices being cities, and edges being roads.

- *Social networks:* The typical example would be Facebook, with the vertices being people, and edge between two people if they are friends on Facebook.

- *Molecules in Chemistry:* In organic chemistry, molecules are made up of different atoms, and are often represented as a graph, with the atoms being vertices, and edges representing covalent bonds between the vertices.

That is all rather informal, though, and to do mathematics we need very precise, formal definitions. We now provide that.

1.1.2 The formal definition of a graph

The formal definition of a graph that we will use is the following:

Definition 1.1.1. A *graph* G consists of a set $V(G)$, called the *vertices* of G, and a set $E(G)$, called the *edges* of G, of the two element subsets of $V(G)$

Example 1.1.2. Consider the water molecule. It has three vertices, and so $V(G) = \{O, H1, H2\}$, and two edges $E(G) = \big\{\{O, H1\}, \{O, H2\}\big\}$

This formal definition has some perhaps unintended consequences about what a graph is. Because we have identified edges with the two things they connect, and have a set of edges, we can't have more than one edge between any two vertices. In many real world examples, this is not the case: for example, on the London Tube, the Circle, District and Picadilly lines all connect Gloucester Road with South Kensington, and so there should be multiple edges between those two vertices on the graph.

Another consequence is that we require each edge to be a two element subset of $V(G)$, and so we do not allow for the possibility of an edge between a vertex and itself, often called a *loop*.

Graphs without multiple edges or loops are sometimes called *simple graphs*. We will sometimes deal with graphs with multiple edges or loops, and will try to be explicit when we allow this. Our default assumption is that our graphs are simple.

Another consequence of the definition is that edges are symmetric, and work equally well in both directions. This is not always the case: in road systems, there are often one-way streets. If we were to model Twitter or Instragram as a graph, rather than the symmetric notion of friends we would have to work with "following". To capture these, we have the notion of a *directed graph*, where rather than just lines, we think of the edges as arrows, pointing from one vertex (the source) to another vertex (the target). To model Twitter or Instagram, we would have an ege from vertex a to vertex b if a followed b.

1.1.3 Named graphs, and basic examples and concepts

Several simple graphs that are frequently referenced have specific names.

Definition 1.1.3. The complete graph K_n is the graph on n vertices, with an edge between any two distinct vertices.

Definition 1.1.4. The empty graph E_n is the graph on n vertices, with no edges.

Definition 1.1.5. The cycle graph C_n is the graph on n vertices $\{v_1, \dots, v_n\}$ with edges $\{\{v_1, v_2\}, \{v_2, v_3\}, \dots, \{v_{n-1}, v_n\}, \{v_n, v_1\}\}$.

Definition 1.1.6. The complelement of a simple graph G, which we will denote G^c, and is sometimes written $\overline{G}$, is the graph with the same vertex set as G, but $\{v, w\} \in E(G^c)$ if and only if $\{v, w\} \notin E(G)$; that is, there is an edge between v and w in G^c if and only if there is not an edge between v and w in G

Example 1.1.7. The empty graph and complete graph are complements of each other; $K_n^c = E_n$

1.2 Degree and handshaking

1.2.1 Definition of degree

Intuitively, the *degree* of a vertex is the "number of edges coming out of it". If we think of a graph G as a picture, then to find the degree of a vertex $v \in V(G)$ we draw a very small circle around v, the number of times the G intersects that circle is the degree of v. Formally, we have:

Definition 1.2.1. Let G be a simple graph, and let $v \in V(G)$ be a vertex of G. Then the *degree of* v, written $d(v)$, is the number of edges $e \in E(G)$ with $v \in e$. Alternatively, $d(v)$ is the number of vertices v is adjacent to.

Example 1.2.2.

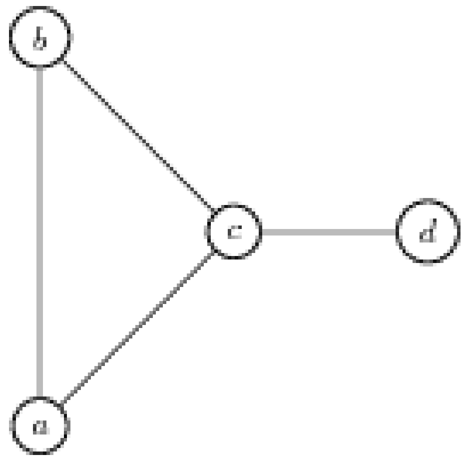


Figure 1.2.3: The graph K

In the graph K shown in Figure 1.2.3, vertices a and b have degree 2, vertex c has degree 3, and vertex d has degree 1.

Note that in the definition we require G to be a simple graph. The notion of degree has a few pitfalls to be careful of G has loops or multiple edges. We still want to the degree $d(v)$ to match the intuitive notion of the "number of edges coming out of v" captured in the drawing with a small circle. The trap to beware is that this notion no longer agrees with "the number of vertices adjacent to v" or the "the number of edges incident to v"

Example 1.2.4.

1.2.2 Extended example: Chemistry

In organic chemistry, molecules are frequently drawn as graphs, with the vertices being atoms, and an edge betwen two vertices if and only if the corresponding atoms have a covalent bond between them (that is, they share a vertex).

Example 1.2.5 (Alkanes)**.**

The location of an element on the periodic table determines the valency of the element – hence the degree that vertex has in any molecule containing that graph:

- Hydrogen (H) and Fluorine (F) have degree 1
- Oxygen (O) and Sulfur (S) have degree 2
- Nitrogen (N) and Phosphorous (P) have degree 3
- Carbon (C) has degree 4

Usually, most of the atoms involved are carbon and hydrogen. Carbon atoms are not labelled with a C, but just left blank, while hydrogen atoms are left off completely. One can then complete the full structure of the molecule using the valency of each vertex. On the exam, you may have to know that Carbon has degree 4 and Hydrogen has degree 1; the valency of any other atom would be provided to you

Graphs coming from organic chemistry do not have to be simple – sometimes there are double bonds, where a pair of carbon atoms have two edges between them.

Example 1.2.6.

If we know the chemical formula of a molecule, then we know how many vertices of each degree it has. For a general graph, this information is known as the degree sequence

Definition 1.2.7 (Degree sequence)**.** The degree sequence of a graph is just the list (with multiplicity) of the degrees of all the vertices.

The following sage code draws a random graph with 7 vertices and 10 edges, and then gives its degree sequence. You can tweak the code to generate graphs with different number of vertices and edges, and run the code multiple times, and the degree sequence should become clear.

```
vertices = 7
edges = 10
g = graphs.RandomGNM(vertices,edges)
g.show()
print g.degree_sequence()
```

Knowing the chemical composition of a molecule determines the degree sequence of its corresponding graph. However, it is possible that the same set of atoms may be put together into a molecule in more than one different ways. In chemistry, these are called *isomers*. In terms of graphs, this corresponds to different graphs that have the same degree sequence.

An important special case is the constant degree sequence.

Definition 1.2.8 (Regular graphs)**.** A graph Γ is *d-regular*, or *regular of degree d* if every vertex $v \in \Gamma$ has the same degree d, i.e. $d(v) = d$.

As a common special case, a regular graph where every vertex has degree three is called *trivalent*, or *cubic*.

Some quick examples:

1. The cycle graph C_n is two-regular
2. The complete graph K_n is $(n-1)$-regular
3. The Petersen graph is trivalent

Exercise 1.2.9.

1.2.3 Handshaking lemma and first applications

Theorem 1.2.10. *(Euler's handshaking Lemma)*

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

Proof. We count the "ends" of edges two different ways. On the one hand, every end occurs at a vertex, and at vertex v there are $d(v)$ ends, and so the total number of ends is the sum on the left hand side. On the other hand, every edge has exactly two ends, and so the number of ends is twice the number of edges, giving the right hand side. □

Euler's handshaking lemma will be particularly useful in Section 3, when we study graphs on surfaces, but already it still has some amusing corollaries.

1.3 Graph Isomorphisms

Generally speaking in mathematics, we say that two objects are "isomorphic" if they are "the same" in terms of whatever structure we happen to be studying. The symmetric group S_3 and the symmetry group of an equilateral triangle D_6 are isomorphic. In this section we briefly briefly discuss isomorphisms of graphs.

1.3.1 Isomorphic graphs

It is possible to draw the same graph in the plane in many different ways – e.g., the pentagon C_5, and its complement the five sided star C_5^c are actually "the same", as indicated by the following labeling of the vertices:

Definition 1.3.1. An isomorphism $\varphi : G \to H$ of simple graphs is a biject $\varphi : V(G) \to V(H)$ between their vertex sets that preserves the number of edges between vertices. In other words, $\varphi(v)$ and $\varphi(w)$ are adjacent in H if and only if v and w are adjancent in G.

Example 1.3.2. This animated gif shows several graphs isomorphic to the Petersen graph, and demonstrates that they are isomorphic. Animated gif by Michael Sollami for this Quanta Magazine article on the Graph Isomorphism problem

Example 1.3.3.

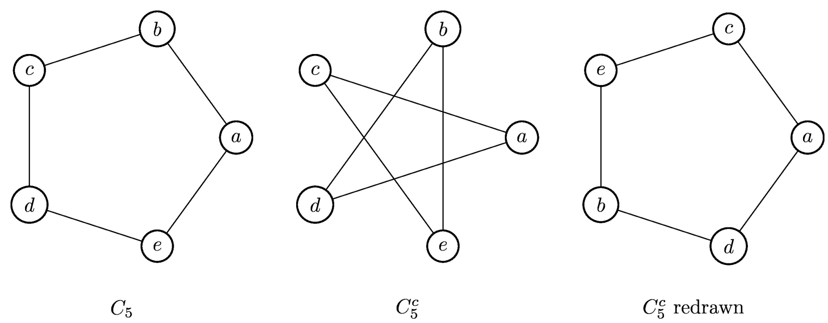


Figure 1.3.4: C_5 is isomorphic to its complement C_5^c

The cycle graph on 5 vertices, C_5 is isomorphic to its complement, C_5^c. The cycle C_5 is usually drawn as a pentagon, and if we were then going to naively draw C_5^c we would draw a 5-sided star. However, we could draw C_5^c differently as shown, making it clear that it is isomorphic to C_5, with isomorphism $\varphi : C_5 \to C_5^c$ defined by $\varphi(a) = a, \varphi(b) = c, \varphi(c) = e, \varphi(d) = b, \varphi(e) = d$.

Although solving the graph isomorphism problem for general graphs is quite difficult, doing it for small graphs by hand is not too bad and is something you must be able to do for the exam. If the two graphs are actually isomorphic, then you should show this by exhibiting an isomrophism; that is, writing down an explicit bijection between their vertex sets with the desired properties. The most attractive way of doing this, for humans, is to label the vertices of both copies with the same letter set.

If two graphs are not isomorphic, then you have to be able to prove that they aren't. Of course, one can do this by exhaustively describing the possibilities, but usually it's easier to do this by giving an obstruction – something that is different between the two graphs. One easy example is that isomorphic graphs have to have the same number of edges and vertices. We'll discuss some others in the next section

1.3.2 Heuristics for showing graphs are or aren't isomorphic

Another, only slightly more advanced invariant is the degree sequence of a graph that we saw last lecture in our discussion of chemistry.

If $\varphi : G \to H$ is an isomorphism of graphs, than we must have $d(\varphi(v)) = d(v)$ for all vertices $v \in G$, and since isomorphisms are bijections on the vertex set, we see the degree sequence must be preserved. However, just because two graphs have the same degree sequences does not mean they are isomorphic.

Slightly subtler invariants are number and length of cycles and paths.

1.3.3 Cultural Literacy: The Graph Isomorphism Problem

This section, as all "Cultural Literacy" sections, is information that you may find interesting, but won't be examined.

The graph isomorphism problem is the following: given two graphs G and H, determine whether or not G and H are isomorphic. Clearly, for any two graphs G and H, the problem is solvable: if G and H both of n vertices, then there are $n!$ different bijections between their vertex sets. One could simply examine each vertex bijection in turn, checking whether or not it maps edges to edges.

The problem is interesting because the naive algorithm discussed above is not very efficient: for large n, $n!$ is absolutely huge, and so in general this algorithm will take a long time. The question is, is there are a faster way to do check? How fast can we get?

The isomorphism problem is of fundamental importance to theoretical computer science. Apart from its practical applications, the exact difficulty of the problem is unknown. Clearly, if the graphs are isomorphic, this fact can be easily demonstrated and checked, which means the Graph Isomorphism is in NP.

Most problems in NP are known either to be easy (solvable in polynomial time, P), or at least as difficult as any other problem in NP (NP complete). This is not true of the Graph Isomorphism problem. In November of last year, Laszlo Babai announced a quasipolynomial-time algorithm for the graph isomorphism problem – you can read about this work in this great popular science article.

1.4 Instant Insanity

So far, our motivation for studying graph theory has largely been one of philosophy and language. Before we get too much deeper, however, it may be useful to present a nontrivial and perhaps unexpected application of graph theory; an example where graph theory helps us to do something that would be difficult or impossible to do without it.

1.4.1 A puzzle

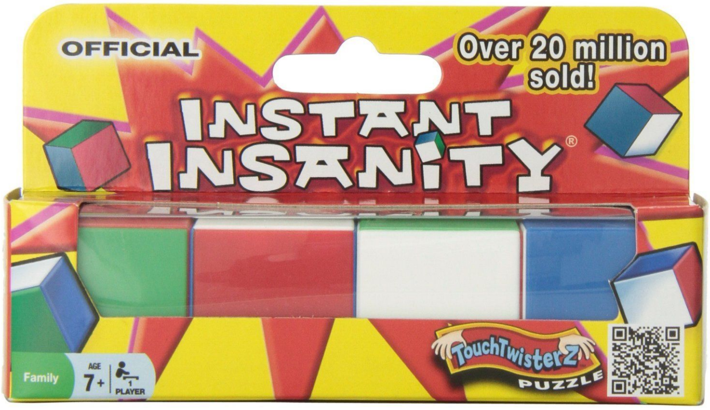


Figure 1.4.1: Instant Insanity Package

There is a puazzle marketed under the name "Instant Insanity", one version of which is shown above. The puzzle is sometimes called "the four cubes problem", as it consists of four different cubes. Each face of each cube is painted one of four different colours: blue, green, red or yellow. The goal of the puzzle is to line the four cubes up in a row, so that along the four long edges (front, top, back, bottom) each of the four colours appears eactly once.

Depending on how the cubes are coloured, this may be not be possible, or there may be many such possibilities. In the original instant insanity, there is exactly one solution (up to certain equivalences of cube positions). The point of the cubes is that there are a large number of possible cube configurations, and so if you just look for a solution without being extremely systematic, it is highly unlikely you will find it.

But trying to be systematic and logical about the search directly is quite difficult, perhaps because we have problems holding the picture of the cube in our mind. In what follows, we will introduce a way to translate the instant insanity puzzle into a question in graph theory. This is obviously in no way necessary to solve the puzzle, but does make it much easier. It also demonstrates the real power of graph theory as a visualization and thought aid.

Exercise 1.4.2.

There are many variations on Instant Insanity, discussions of which can be found here and here. There's also a commercial for the game.

1.4.2 Enter graph theory

It turns out that the important factor of the cubes is what color is on the opposite side of each face. Suppose we want face one facing forward. Then we have four different ways to rotate the cube to keep this the same. The same face will always appear on the opposite side, but we can get any of the remaining four faces to be on top, say.

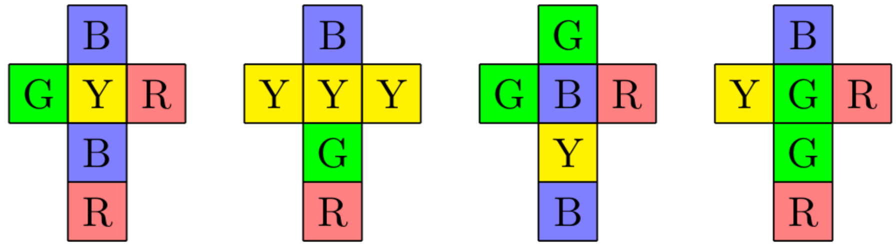


Figure 1.4.3: An impossible set of cubes

Let us encode this information in a graph. The vertices of the graph will be the four colors, B, G, R and Y. We will put an edge between two colors each time they appear as opposite faces on a cube, and we will label that edge with a number 1-4 denoting which cube the two opposite faces appear. Thus, in the end the graph will have twelve edges, three with each label 1-4. For from the first cube, there will be a loop at B, and edge between G and R, and an edge between Y and R. The graph corresponding to the four cubes above is the following:

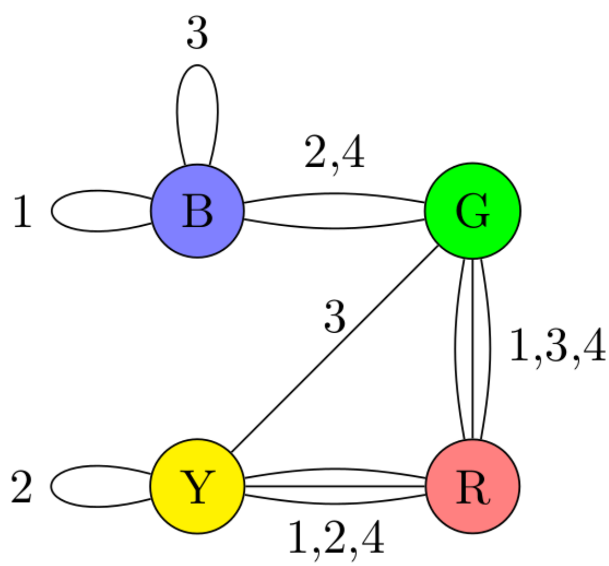


Figure 1.4.4: The graph constructed from the cubes in Figure 1.4.3

1.4.3 Proving that our cubes were impossible

We now analyze the graph to prove that this set of cubes is not possible.

Suppose we had an arrangement of the cubes that was a solution. Then, from each cube, pick the edge representing the colors facing front and back on that cube. These four edges are a subgraph of our original graph, with one edge of each label, since we picked one edge from each cube. Furthermore, since we assumed the arrangement of cubes was a solution of instant insanity, each color appears once on the front face and once on the back. In terms of our subgraph, this translates into asking that each vertex has degree two.

We can get another subgraph satisfying these two properties by looking at the faces on the top and bottom for each cube and taking the corresponding edges. Furthermore, these two subgraphs do not have any edges in common.

Thus, given a solution to the instant insanity problem, we found a pair of subgraphs S_1, S_2 satisfying:

1. Each subgraph S_i has one edge with each label 1,2,3,4

2. Every vertex of S_i has degree 2

3. No edge of the original graph is used in both S_1 and S_2

As an exercise, one can check that given a pair of subgraphs satisfying 1-3, one can produce a solution to the instant insanity puzzle.

Thus, to show the set of cubes we are currently examining does not have a solution, we need to show that the graph does not have two subgraphs satisfying properties 1-3.

To do, this, we catalog all graphs satisfying properties 1-2. If every vertex has degree 2, either:

1. Every vertex has a loop

2. There is one vertex with a loop, and the rest are in a triangle

3. There are two vertices with loops and a double edge between the other two vertices

4. There are two pairs of double edges

5. All the vertices live in one four cycle

6. A subgraphs of type 1 is not possible, because G and R do not have loops.

For subgraphs of type 2, the only triangle is G-R-Y, and B does have loops. The edge between Y-G must be labeled 3, which means the loop at B must be labeled 1. This means the edge between G and R must be labeled 4 and the edge between Y and R must be 2, giving the following subgraph:

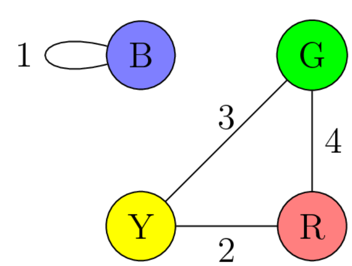


Figure 1.4.5: A subgraph for a solution for one pair of faces

For type 3, the only option is to have loops at B and Y and a double edge between G and R. We see the loop at Y must be labeled 2, one of the edges between G and R must be 4, and the loop at B and the other edge between G and R can switch between 1 and 3, giving two possibilities:

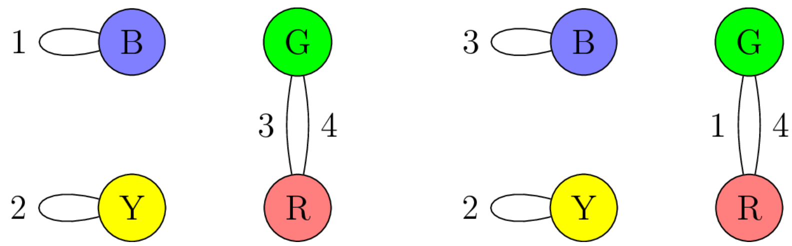


Figure 1.4.6: Two more subgraphs for a partial solutions

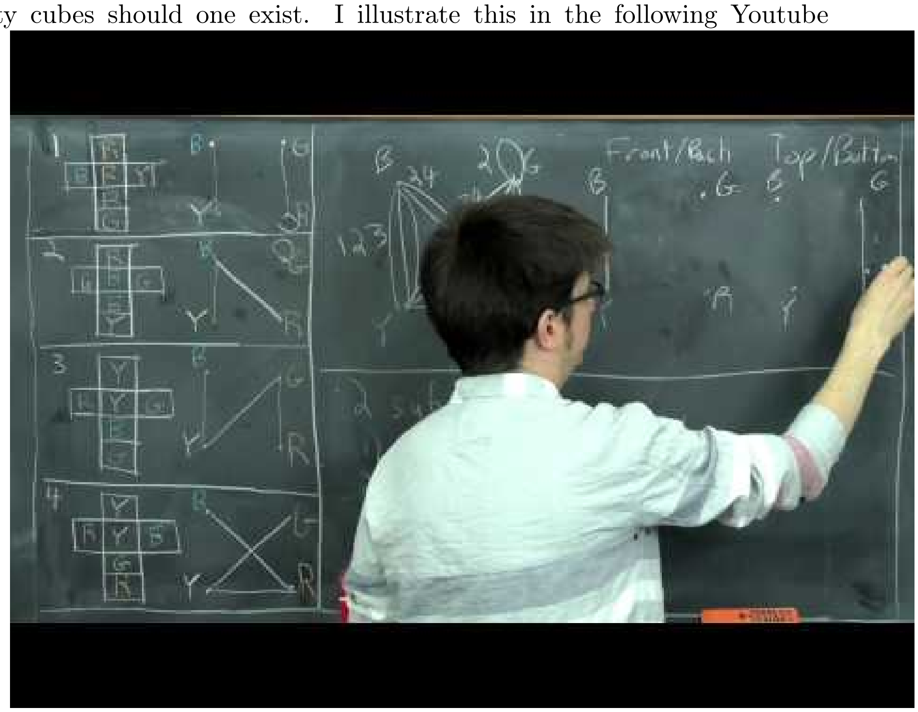
For subgraphs of type 4, the only option would be to have a double edge between B and G and another between Y and R; however, none of these edges are labeled 3 and this option is not possible.

Finally, subgraphs of type 5 cannot happen because B is only adjacent to G and to itself; to be in a four cycle it would have two be adjacent to two vertices that aren't itself.

This gives three different possibilities for the subgraphs SiSi that satisfy properties 1 and 2. However, all three possibilities contain the the edge labeled 4 between G and R; hence we cannot choice two of them with disjoint edges, and the instant insanity puzzle with these cubes does not have a solution.

1.4.4 Other cube sets

The methods above also give a way to find the solution to a set of instant insanity cubes should one exist. I illustrate this in the following Youtube video:



Other cube sets

1.5 Exercises

1. For each of the following sequences, either give an example of such a graph, or explain why one does not exist.

(a) A graph with six vertices whose degree sequence is $[5, 5, 4, 3, 2, 2]$

(b) A graph with six vertices whose degree sequence is $[5, 5, 4, 3, 3, 2]$

(c) A graph with six vertices whose degree sequence is $[5, 5, 5, 5, 3, 3]$

(d) A simple graph with six vertices whose degree sequence is $[5, 5, 5, 5, 3, 3]$

2. For the next Olympic Winter Games, the organizers wish to expand the number of teams competing in curling. They wish to have 14 teams enter, divided into two pools of seven teams each. Right now, they're thinking of requiring that in preliminary play each team will play seven games against distinct opponents. Five of the opponents will come from their own pool and two of the opponents will come from the other pool. They're having trouble setting up such a schedule, so they've come to you. By using an appropriate graph-theoretic model, either argue that they cannot use their current plan or devise a way for them to do so.

3. Figure 1.5.1 contains four graphs on six vertices. Determine which (if any) pairs of graphs are isomorphic. For pairs that are isomorphic, give an isomorphism between the two graphs. For pairs that are not isomorphic, explain why.

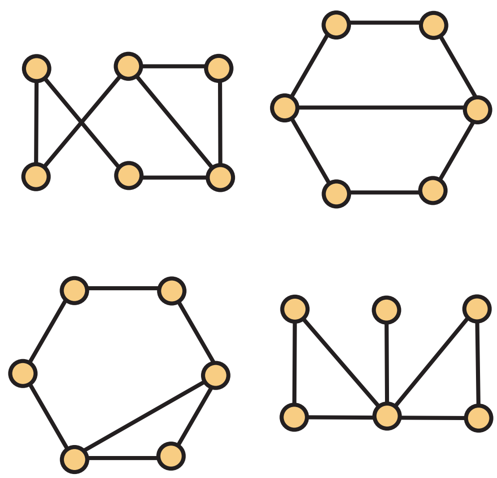


Figure 1.5.1: Are these graphs isomorphic?

4. Recall that G^c denotes the complement of a graph G. Prove that $f : G \to H$ is an isomorphism of graphs if and only if $f : G^c \to H^c$ is an isomorphism.

5. Determine the number of non-isomorphic simple graphs with seven vertices such that each vertex has degree at least five.

Hint. Consider the previous exercise

Chapter 2

Walks

In this chapter we investigate walks in graphs. We first look at some basic definitions and examples, we discuss Dijkstra's algorithm for finding the shortest path between two points in a weighted graph, and we discuss the notions of Eulerian and Hamiltonian graphs.

2.1 Walks: the basics

If the edges in a graph Γ represent connections between different cities, it is obvious to strart planning longer trips that compose several of these connections. The notion of a *walk* formally captures this definition; the formal notions of *path* and *trail* further ask that we not double back on ourselves or repeat ourselves in certain formally defined ways.

Once we've done that, we investigate what it means for a graph to be connected or disconnected.

2.1.1 Walks and connectedness

Before we see the formal definition of a walk, it will be useful to see an example:

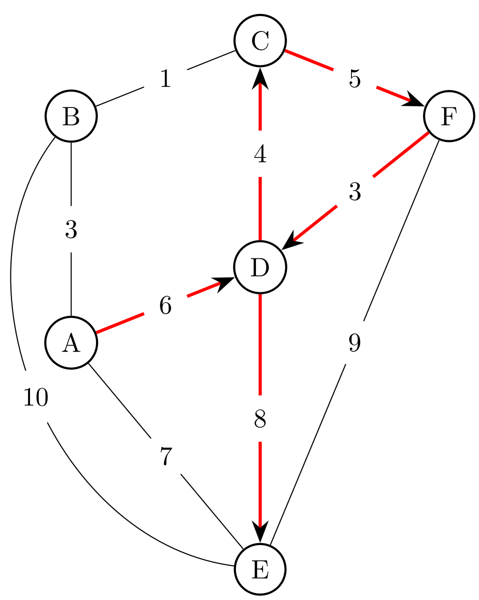


Figure 2.1.1: Example of a walk

In the graph shown, the vertices are labelled with letters, and the edges are labelled with numbers, and we have a walk highlighted in red, and with arrowtips drawn on the edges. Starting from vertex A, we can take edge 6 to vertex D, and then edge 5 to vertex C, edge 5 to vertext F, edge 3 back to vertex D, and finally edge 8 to vertex E. Intuitively,then, a walk strings together several edges that share vertices in between. Makign that formal, we have the following.

Definition 2.1.2 (Walk)**.** *walk* in a graph Γ is a sequence

$$v_0, e_1, v_1, e_2, v_2, \ldots, v_{n-1}, e_n, v_n$$

where the v_i are vertices, the e_j are edges, and the edge e_j goes between vertices v_{j-1} and v_j.

We say that the walk is between vertices $a = v_0$ and $b = v_n$

With this notation for a walk, Example Figure 2.1.1, the walk shown would be written $A, 6, D, 4, C, 5, F, 3, D, 8, E$. The visual representation of the walk on the graph is vastly more intuitive, the written one feeling cumbersome in comparison.

The definition of walk above contains some extra information. If we just know the sequence of edges we can reconstruct what the vertices have to be (assuming we have at least two edges in the walk). Alternatively, if the graph Γ does not have multiple edges, it is enough to just know the vertices v_i, but if Γ has multiple edges that just knowing the vertices does not determine the walk.

Definition 2.1.3 (Connected)**.** We say a graph Γ is *connected* if for any two vertices v, w, there is a walk from v to w in Γ.

Definition 2.1.4 (Disjoint union)**.** Given two graphs Γ_1 and Γ_2, the *disjoint union* $\Gamma_1 \sqcup \Gamma_2$ is obtained by taking the disjoint union of both the vertices and edges of Γ_1 and Γ_2. So $\Gamma_1 \sqcup \Gamma_2$ consists of a copy of Γ_1 and a copy of Γ_2, with no edges in between the two graphs.

Definition 2.1.5 (Disconnected)**.** A graph Γ is *disconnected* if we can write $\Gamma = \Gamma_1 \sqcup \Gamma_2$ for two proper (i.e., not all of Γ) subgraphs Γ_1 and Γ_2.

We now have a definition for what it means for a graph to be connected, and another for what it means for a graph to be disconnected. From everday usage of this words, we would certainly hope that a graph is disconnected if and only if it is not connected. However, it is not immediately clear from the definitions as written that this is the case.

Lemma 2.1.6. *The following are equivalent:*

1. *. Γ is connected*

2. *Γ is not disconnected*

Proof. 1 implies 2: Supppose that Γ is connected, and let $v, w \in V(\Gamma)$; we want to show that there is a walk from v to w.

Define $S \subset V(\Gamma)$ to be the set of all vertices $u \in V(\Gamma)$ so that there is a walk from v to u; we want to show that $w \in S$.

First, observe that there are no edges from S to $V(\Gamma) \setminus S$. Suppose that e was an edge between $a \in S$ and $b \in \Gamma \setminus S$. Since $a \in S$, by the definition of S

there is a walk $v = v_0 v_1 v_2 \cdots v_m = a$ from v to a. We can add the edge e to the end of the walk, to get a walk from v to b, and hence by definition $b \in S$.

Now suppose that $w \notin S$. Then S and $V(\Gamma) \setminus S$ are both nonempty, and by the above there are no edges between them, and so Γ is not connected, a contradiction.

To prove 2 implies 1, we prove the contrapositive. If Γ is not connected, then there are two vertices $v, w \in V(\Gamma)$ so that there is no walk from v to w.

Suppose that $\Gamma = \Gamma_1 \sqcup \Gamma_2$, and pick $v \in V(\Gamma_1), w \in V(\Gamma_2)$. Any walk from v to w starts in $V(\Gamma_1)$ and ends in $V(\Gamma_2)$, and so at some point there must be an edge from a vertex in Γ_1 to a vertex in Γ_2, but there are no such edges
□ □

2.1.2 Types of Walks

Many questions in graph theory ask whether or not a walk of a certain type exists on a graph: we introduce some notation that will be needed for these questions.

Definition 2.1.7. We say a walk is *closed* if it starts and ends on the same vertex; i.e., $v_0 = v_n$. The *length* of a walk is n, the number of edges in it. The *distance* between two vertices v and w is the length of the shortest walk from v to w, if one exists, and ∞ if one does not exist.

It is sometimes convenient to have terminology for walks that don't back-track on themselves:

Definition 2.1.8.

1. If the edges e_i of the walk are all distinct, we call it a *trail*
2. If the vertices v_i of the walk are all distinct (except possibly $v_0 = v_m$), we call the walk a *path*. The exception is to allow for the possibility of closed paths.

Lemma 2.1.9. *Let $v, w \in V(\Gamma)$. The following are equivalent:*

1. *There is a walk from v to w*
2. *There is a trail from v to w*
3. *There is a path from v to w.*

As is often the case, the formal write-up of the proof makes something that can seem very easy intuitively look laborious, so it's worth anlysing it briefly for our example walk $A - D - C - F - D - E$ from Figure 2.1.1. This walk is not a path as it repeats the vertex D; however, we may simply remove the triangle $D - C - F - D$ from the walk to get the trail $A - D - E$. this idea is what works in general.

Proof. It is immediate from the definitions that 3 implies 2 which implies 1, as any path is a trail, and any trail is a walk.

That 1 implies 3 is intuitively obvious: if you repeat a vertex, then you've visited someplace twice, and weren't taking the shortest route. Let's make this argument mathematically precise.

Suppose we have a walk $v = v_0, e_1, \ldots, e_m, v_m = w$ that is not a path. Then, we must repeat some vertex, say $v_i = v_k$, with $i < k$. Then we can cut out all the vertices and edges between v_i and v_k to obtain a new walk

$$v = v_0, e_1, v_1, \ldots, e_i, v_i = v_k, e_{k+1}, v_{k+1}, e_{k+2}, v_{k+2}, \ldots, v_m$$

Since $i < k$, the new walk is strictly shorter than our original walk. Since the length of a walk is finite, if we iterate this process the result must eventually terminate. That means all our vertices are distinct, and hence is a path. $\square$

2.2 Eulerian Walks

In this section we introduce the problem of Eulerian walks, often hailed as the origins of graph theroy. We will see that determining whether or not a walk has an Eulerian circuit will turn out to be easy; in contrast, the problem of determining whether or not one has a Hamiltonian walk, which seems very similar, will turn out to be very difficult.

2.2.1 The bridges of Konigsburg

The city of Konigsberg (now Kaliningrad) was built on two sides of a river, near the site of two large islands. The four sectors of the city were connected by seven bridges, as follows (picture from Wikipedia):

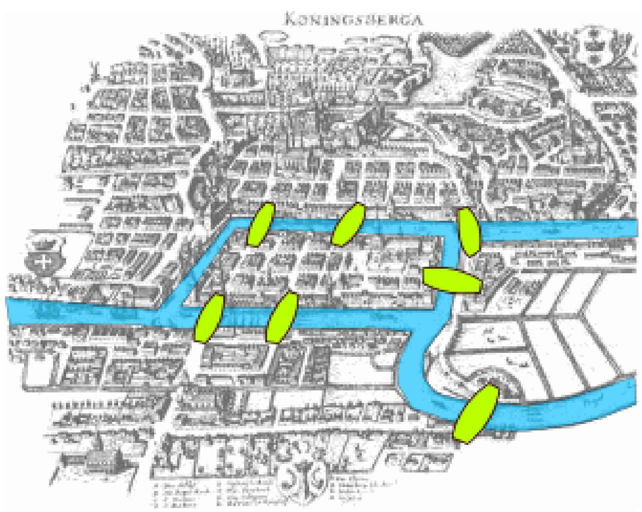


Figure 2.2.1: The city of Konigsburg in Euler's time

A group of friends enjoyed strolling through the city, and created a game: could they take a walk in the city, crossing every bridge exactly once, and return to where they started from? They couldn't find such a walk, but they couldn't prove such a walk wasn't possible, and so they wrote to the mathematician Euler, who proved that such a walk is not possible.

2.2.2 Eulerian Walks: definitions

We will formalize the problem presented by the citizens of Konigsburg in graph theory, which will immediately present an obvious generalization.

We may represent the city of Konigsburg as a graph Γ_K; the four sectors of town will be the vertices of Γ_K, and edges between vertices will represent the bridges (hence, this will not be a simple graph).

Then, the question reduces to finding a closed walk in the graph that will uses every edge *exactly* once. In particular, this walk will not use any edge more than once and hence will be a trail.

Exercise 2.2.2.

Definition 2.2.3. Let G be a graph. An *Eulerian cycle* is a closed walk that uses every edge of G exactly once.

If G has an Eulerian cycle, we say that G is *Eulerian*.

If we weaken the requirement, and do not require the walk to be closed, we call it an Euler path, and if a graph G has an Eulerian path but not an Eulerian cycle, we say G is *semi-Eulerian*

The question of the walkers of Konigsburg is then equivalent to asking if the graph Γ_K is Eulerian.

2.2.3 The first theorm of graph theory

Theorem 2.2.4. *A connected graph Γ is Eulerian if and only if every vertex of Γ has even degree*

Proof. □

Remark 2.2.5. Note that it does *not* say: "A graph Γ is Eulerian if and only if it is connected and every vertex has even degree." This statement in quotation marks is false, but for "stupid" reasons. If Γ is Eulerian, and E_n is the graph with n vertices wit no edges, then $\Gamma \sqcup E_n$ is Eulerian but not connected. These are the only examples of such graphs.

Theorem 2.2.6. *A connected graph Γ is semi-Eulerian if and only if it has exactly two vertices with odd degree.*

Proof. A minor modification of our argument for Eulerian graphs shows that the condition is necessary. Suppose that Γ is semi-Eulerian, with Eulerian path $v_0, e_1, v_1, e_2, v_3, \ldots, e_n, v_n$. Then at any vertex other than the starting or ending vertices, we can pair the entering and leaving edges up to get an even number of edges.

However, at the first vertex v_0 the path cleaves along e_1 the first time but never enters it accordingly, so that v_0 has an odd degree; similarly, at v_n the path enters one final time along e_n without leaving, and so v_n also has an odd degree.

To see the condition is sufficient we could also modify the argument for the Eulerian case slightly, but it is slicker instead to *reduce* to the Eulerian case. Suppose that Γ is connected, and that vertices v and w have odd degree and all other vertices of Γ have even degree. Then we can construct a new graph Γ' by adding an extra edge $e = vw$ to Γ. Then Γ' is connected and every vertex has even degree, and so it has an Eulerian cycle. Deleting the edge e that we added from this cycle gives an Eulerian path from v to w in Γ. □

2.3 Hamiltonian cycles

We now introduce the concept of Hamiltonian walks. Though on the surface the question seems very similar to determining whether or not a graph is Eulerian, it turns out to be much more difficult.

Definition 2.3.1. A graph is *Hamiltonian* if it has a closed walk that uses every vertex exactly once; such a path is called a *Hamiltonian cycle*

First, some very basic examples:

1. The cycle graph C_n is Hamiltonian.
2. Any graph obtained from C_n by adding edges is Hamiltonian

3. The path graph P_n is *not* Hamiltonian.

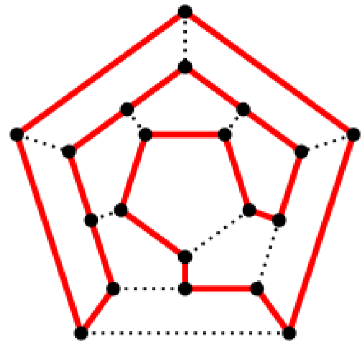


Figure 2.3.2: Image of the Icosian game from the Puzzle Musuem; image of the solution from Wikipedia

The term Hamiltonian comes from William Hamiltonian, who invented (a not very successful) board game he termed the "icosian game", which was about finding Hamiltonian cycles on the dodecahedron graph (and possibly its subgraphs)

2.4 Exercises

1. The questions in this exercise pertain to the graph **G** shown in Figure 2.3.1.

(a) What is the degree of vertex 8?

(b) What is the degree of vertex 10?

(c) How many vertices of degree 2 are there in **G**? List them.

(d) Find a cycle of length 8 in **G**.

(e) What is the length of a shortest path from 3 to 4?

(f) What is the length of a shortest path from 8 to 7?

(g) Find a path of length 5 from vertex 4 to vertex 6.

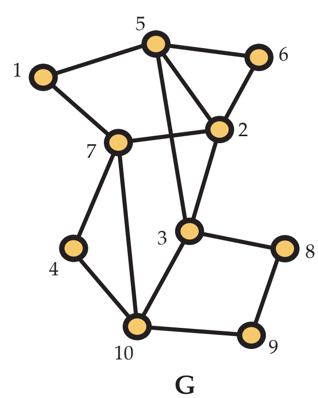


Figure 2.4.1: A graph

2. Draw a graph with 8 vertices, all of odd degree, that does not contain a path of length 3 or explain why such a graph does not exist.

3. Find an eulerian circuit in the graph **G** in Figure 2.3.2 or explain why one does not exist.

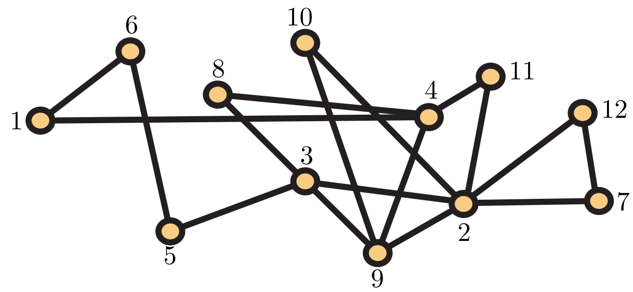


Figure 2.4.2: A graph **G**

4. Consider the graph **G** in Figure **??**. Determine if the graph is eulerian. If it is, find an eulerian circuit. If it is not, explain why it is not. Determine if the graph is hamiltonian. If it is, find a hamiltonian cycle. If it is not, explain why it is not.

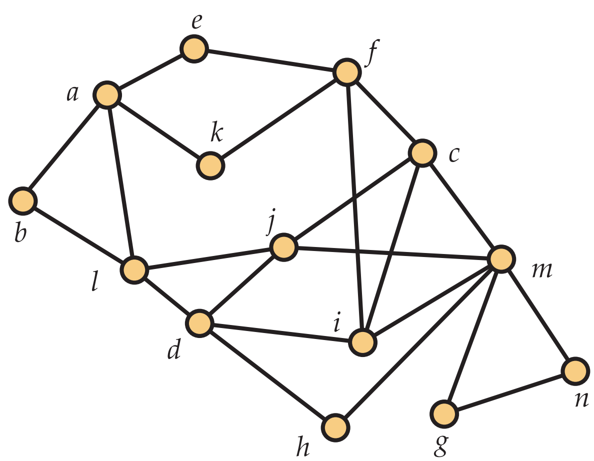


Figure 2.4.3: A graph **G**