Chapter 1

Introduction

The first chapter is an introduction, including the formal definition of a graph and many terms we will use throughout. More importantly, however, are examples of these concepts and how you should think about them. As a first nontrivial use of graph theory, we explain how to solve the "Instant Insanity" puzzle.

1.1 A first look at 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:

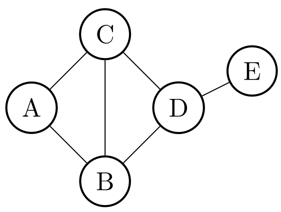


Figure 1.1.1: A graph

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.

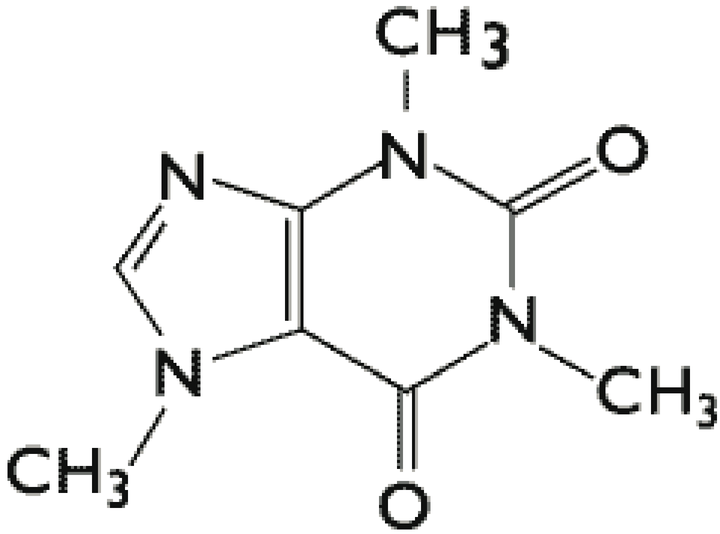


Figure 1.1.2: A Caffeine Molecule, courtesey Wikimedia Commons

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 1.1.3. 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 1.1.4. Consider the water molecule, which consists of a single oxygen atom, connected to two hydrogen atoms. 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. As another example, in organic chemistry, there are often "double bonds", instead of just one.

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 Basic examples and concepts

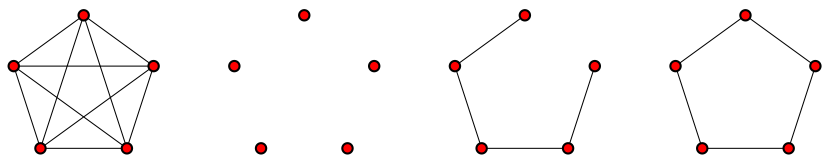
Several simple graphs that are frequently referenced have specific names.

Definition 1.1.5. The complete graph K_n is the graph on n vertices, with an edge between any two distinct vertices.

Definition 1.1.6. The empty graph E_n is the graph on n vertices, with no edges.

Definition 1.1.7. The path graph P_n is the graph on n vertices $\{v_1, \ldots, v_n\}$ with edges $\{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}\}$.

Definition 1.1.8. The cycle graph C_n is the graph on n vertices $\{v_1, \ldots, v_n\}$ with edges $\{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}, \{v_n, v_1\}\}$.



K_5 E_5 P_5 C_5

Figure 1.1.9: Basic graphs

Definition 1.1.10. 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 1.1.11. The empty graph and complete graph are complements of each other; $K_n^c = E_n$

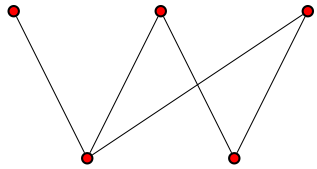
The path graph P_4 and its complement are shown below:

P_4 P_4^c

It commonly occurs that there are two different types of vertices, and the edges only go between vertices of the two types. For example, we may be modelling a company, and one type of vertices may represent the employees, and another type of vertices could represent the different jobs that need done, with an edge between a worker and a job if the worker is qualified to do that particular job. We call these graphs *bipartite*.

Definition 1.1.12. A graph $\mathbf{G}$ is *bipartite* if its vertices can be coloured red and blue so that every edge goes between a red vertex and a blue vertex.

Example 1.1.13. The graph below is bipartite.



As another example, note that the cycle graph C_4 is bipartite – we can colour vertices 1 and 3 red, and vertices 2 and 4 blue. But the cycle graph C_3 is *not* bipartite: as the two colours are interchangable, we can assume we coloured vertex 1 red; then since it is adjacent to both 2 and 3, those vertices must both be blue, but they're adjacent to each other, which violates the definition of bipartite. More generally, we have:

Lemma 1.1.14. *The cycle graph C_n is bipartite if and only if n is even.*

Proof. Let's try to colour the vertices of C_n red and blue so that adjacent vertices have different colour. Without loss of generality, we may assume that v_1 is coloured blue. Then since it is adjacent to v_1, v_2 must be coloured red. Continuing in this way, we see that v_k is blue for odd k and red for even k. But v_n is adjacent to v_1, and so these will have different colours precisely when n is even. □

Lemma 1.1.15. *A graph $\mathbf{G}$ is bipartite if and only if $\mathbf{G}$ has no subgraphs that are isomorphic to C_{2k+1}*

Proof. First, note that if $\mathbf{G}_2$ is a subgraph of $\mathbf{G}_1$, and $\mathbf{G}_1$ is bipartite, then so is $\mathbf{G}_2$: colouring the vertices of $\mathbf{G}_1$ red and blue in particular colours the vertices of $\mathbf{G}_2$ as well. Hence, we see if that $\mathbf{G}$ has a subgraph isomorphic to C_{2k_1}, which isn't bipartite by the previous lemma, then $\mathbf{G}_1$ can't be bipartite, either.

In the other direction, we assume that $\mathbf{G}$ has no subgraphs isomorphic to C_{2k+1}; we need to prove that $\mathbf{G}$ is bipartite. Again, let's try to colour the vertices of $\mathbf{G}$ red and blue so that adjancent vertices have different colours. Choose a vertex v of $\mathbf{G}$, without loss of generality we may assume that v is coloured blue; then all vertices adjacent to v – i.e., those vertices at distance 1 from v – are coloured red. The vertices adjacent to those must be blue, and the ones adjacent to those must be red, alternating out. This suggests trying to colour all vertices at odd distance from v red, and those vertices at even distance from v blue. We will show that if this colouring has two vertices of the same colour that are adjacent, then $\mathbf{G}$ has an odd cycle.

Assume that u and w are two vertices coloured red that are adjacent. Since u is red, the distance from v to u is odd – there is a path $v = v_0 - v_1 - \cdots - v_{2k+1} = u$. Similarly, there is an odd length path from v to w: $v = w_0 - w_1 - \cdots - w_{2\ell+1} = w$. Taking the union of these two subgraphs and the edge connecting u and w, we get a closed walk consisting of $(2k+1)+(2\ell+1)+1 = 2k+2\ell+3$ edges, which is odd. This walk may repeat some vertices and edges, but if it does we can split it into two smaller walks, one of which must have odd length, and eventually we must get a closed walk of odd length that doesn't repeat any vertices.

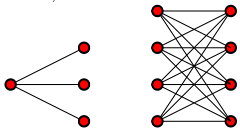
The case that u and w are both coloured blue is completely analogous, except we will be merging two paths with an even number of edges and one extra edge to get a path with odd length. □

A special type of bipartite graph is the complete bipartite graphs $K_{m,n}$, which are the simple graphs that have as many edges as possible while still being bipartite.

Definition 1.1.16. The *complete bipartite graph* $K_{m,n}$ is the graph with m red vertices and n blue vertices, and an edge between very red vertex and every blue vertex.

Example 1.1.17. The complete bipartite graph $K_{2,2}$ is isomorphic to C_4.

The graphs $K_{1,3}$ and $K_{4,4}$ are drawn below.



$K_{1,3}$ $K_{4,4}$

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.

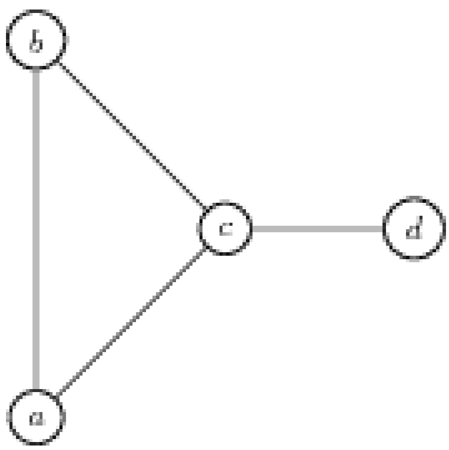


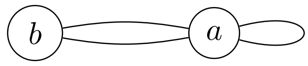
Figure 1.2.3: The graph K

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.

The graph G to the right has two vertices, a and b, and three edges, two between a and b, and a loop at a. Vertex a has degree 4, and vertex b has degree 2.



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

1.2.3 Handshaking lemma and first applications

To motivative the Handshaking Lemma, we consider the following question. Suppose there seven people at a party. Is it possible that everyone at the party knows exactly three other people?

We can model the situation a graph, with vertices being people at the party, and an edge between two vertices if the corresponding people know each other. The question is then asking for the existence of a graph with seven vertices so that every vertex has degree three. It is then natural to attempt to solve the problem by trying to draw such a graph. After a few foiled attempts, we begin to suspect that it's not possible, but doing a case-by-case elimination of all the possibilities is daunting. It's easier to find a reason why we can't draw such a graph.

We will do this as follows: suppose that everyone at the party who knows each other shakes hands. How many handshakes will occur? On the one hand, from the definitions this would just be the number of edges in the graph. On the other hand, we can count the number of handshakes working person-by-person: each person knows three other people, and so is involved in three handshakes. But each handshake involves two people, and so if we count $7 * 3$ we've counted each handhsake twice, and so there should be $7 * 3/2 = 10.5$ handshakes happening, which makes no sense, as we can't have half a handshake. Thus, we have a contradiction, and we conclude such a party isn't possible.

Euler's handshaking Lemma is a generalization of the argument we just made to an arbitrary graph.

Theorem 1.2.9. *(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. □

We have seen already seen one use of Euler's handshaking Lemma, but it will be particularly useful in Chapter 3, when we study graphs on surfaces.

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

The "same" graph can be drawn in the plane in multiple different ways. For instance, the two graphs below are each the "cube graph", with vertices the 8 corners of a cube, and an edge between two vertices if they're connected by an edge of the cube:

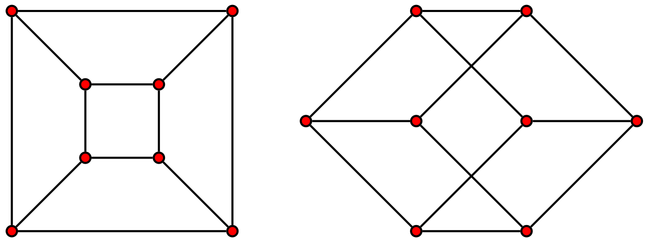


Figure 1.3.1: Two drawings of the cube graph

Example 1.3.2. It is not hard to see that the two graphs above are both drawings of the cube, but for more complicated graphs it can be quite difficult at first glance to tell whether or not two graphs are the same. For instance, there are many ways to draw the Petersen graph that aren't immediately obvious to be the same.

This animated gif created by Michael Sollami for this Quanta Magazine article on the Graph Isomorphism problem illustrates many different such drawings in a way that makes the isomorphisms apparent.

Definition 1.3.3. 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 1.3.4.

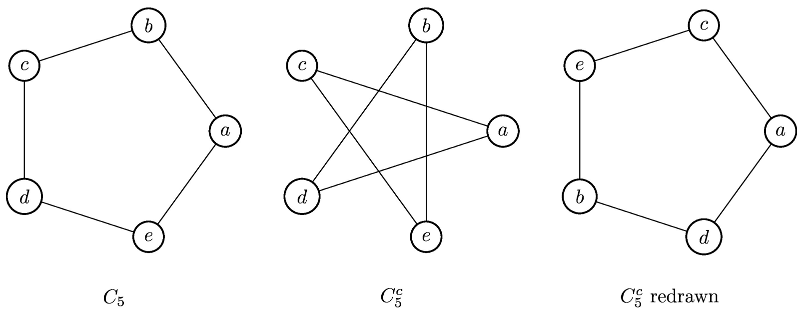


Figure 1.3.5: C_5 is isomorphic to its complement C_5^c

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

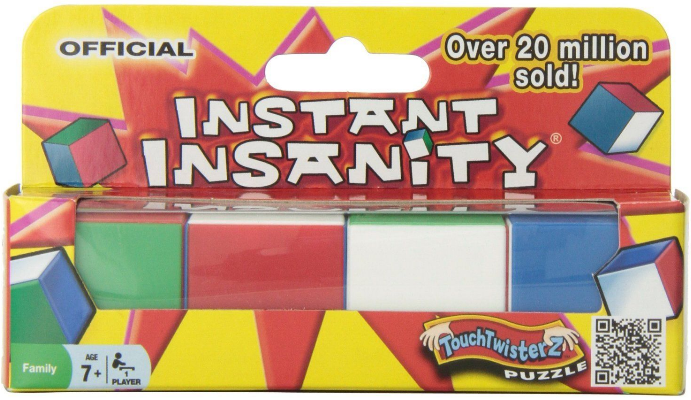


Figure 1.4.1: Instant Insanity Package

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.

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.

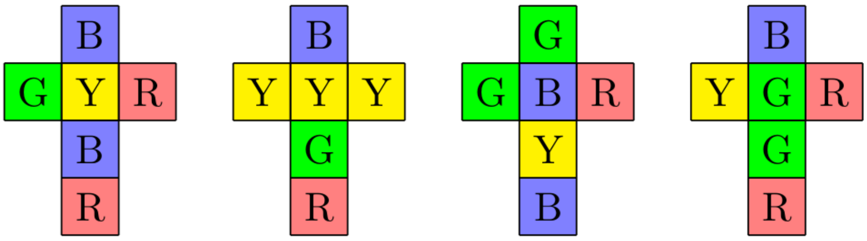


Figure 1.4.2: An impossible set of cubes

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:

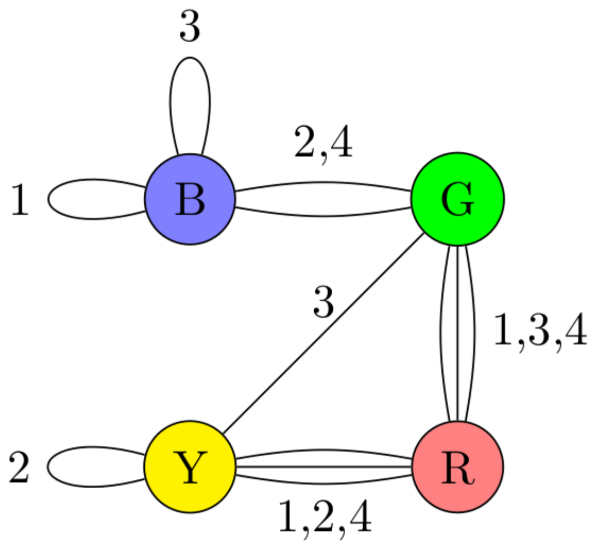


Figure 1.4.3: The graph constructed from the cubes in Figure 1.4.2

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:

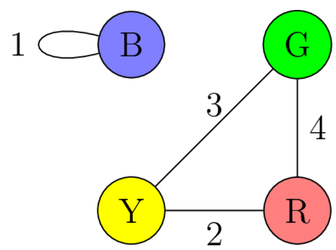


Figure 1.4.4: A subgraph for a solution for one pair of faces

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:

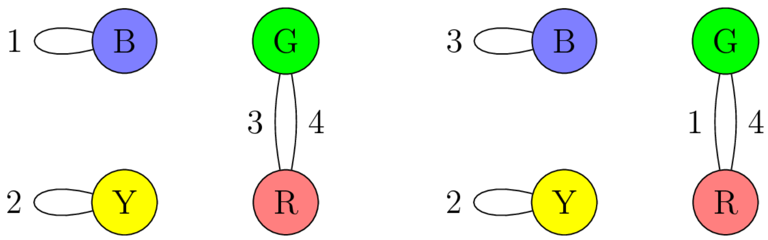


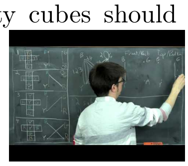
Figure 1.4.5: Two more subgraphs for a partial solutions

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
www.youtube.com/watch?v=GsbhRfjaaN8

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.

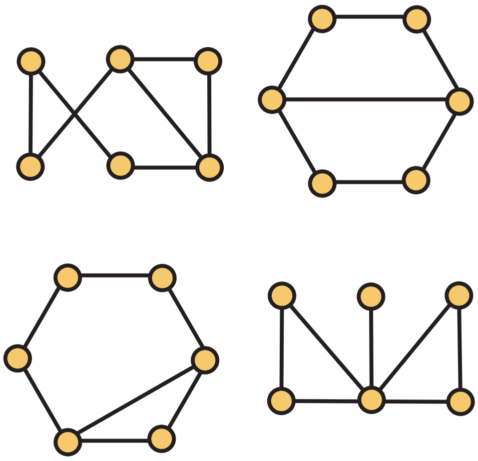


Figure 1.5.1: Are these graphs isomorphic?

4. Let **G** be a simple graph with n vertices and degree sequence $a_1, a_2, \ldots, a_n$. What's the degree sequence of its complement $\mathbf{G}^c$?

5. Let G be the graph with graph with vertices consisting of the 10 three element subsets of $\{a, b, c, d, e\}$, and two vertices adjacent if they share exactly one element. So, for example, the two vertices $v = \{a, c, e\}$ and $w = \{b, c, d\}$ are adjacent, but neither v or w is adjacent to $u = \{a, b, c\}$.
Draw G in a way that shows it is isomorphic to the Petersen graph.
Now let H be the graph with vertices consisting of the 10 *two* element subsets of $\{a, b, c, d, e\}$, and two vertices adjacent if they share *no* elements. Without drawing H, write down an isomorphism between G and H. Hint: There's a "natural" bijection between the two and three element subsets of $\{a, b, c, d, e\}$

6. 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.

7. 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

8. Consider the standard Instant Insanity puzzle, with four cubes and four colours. Explain why one would expect there to be 331,776 different cube configurations. Further explain why there would be fewer configurations if any cubes are coloured with symmetries.
In the text, we solve the puzzle by finding certain pairs of subgraphs. Assuming that none of the cubes are coloured symmetrically, explain why each pair of subgraphs corresponds to at least 8 different cube configurations that are actually solutions, and why, depending on the isomorphism type of the subgraphs found, there may be more solutions.

9. Variations of the Insant Insanity puzzle increase the number of cubes and the number of colours. Explain how to modify our graph theoretic solution to solve the puzzle when we have n cubes, each face of which is coloured one of n colours, and we want to line up the cubes so that each of the top, bottom, front and rear strings of cubes displays each of the n colours exactly once.

10. Use the method from the previous question to solve the following set of six cubes, marketed under the name "Drive ya crazy", where each face is coloured either blue, cyan, green, orange, red, or yellow.

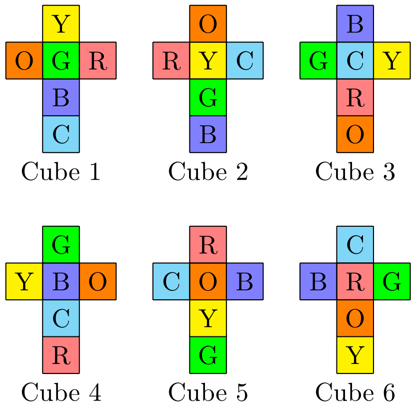


Figure 1.5.2: The six cubes from "Drive Ya crazy"