Lecture Notes

1 Graphs

This lecture is about *graphs*. These are mathematical objects used heavily in computer science. Often one can reduce a real-world problem to a purely mathematical statement about graphs. If the graph problem can be solved, then— in principle at least— the real-world problem is solved.

Here is an example. Among any six people on the planet, must there be three people who all know each other or three people who are all strangers?

To think about this problem, we could make a diagram where each person is represented by a dot. Two dots are connected by a solid line if the people know each other and by a dotted line if they do not. This amounts to a picture of a graph. An example is shown in Figure 1.

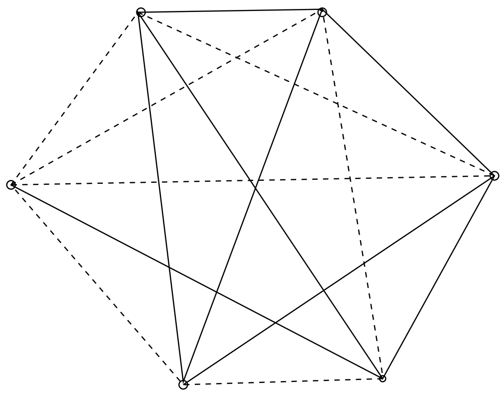


Figure 1: *The six circles represent people. A solid line between two circles means that the two people shook hands. A dotted line between two circles means that the two people did not shake hands. Is there any way to draw the diagram so that there is no solid triangle and no dashed triangle?*

If there is a triangle of solid lines, then three people all know each other. If there is a triangle of dotted lines, then three people are all strangers. The real-world question is now reduced to whether every such diagram contains either a solid triangle or a dashed triangle. This is a problem about graphs.

The following theorem states that every such diagram does contain either a solid triangle or a dashed triangle. That is, among any six people on the planet, there must be three people who all know each other or three people who are all strangers.

Theorem 1.1 If there are six dots and every pair is connected by either a dashed line or a solid line, then there is either a dashed triangle or a solid triangle.

Proof. Look at one of the dots d. There are five lines connected to it. Either at least three are solid or at least three or dotted. Suppose that three lines leading from the dot are solid. (The argument is the same in the other case.) These three lines lead to three more dots; call them x, y, and z.

Now there are two possibilities. First, if there is a solid line between any two of the points x, y, and z, then there is a solid triangle defined by these two points together with d. Otherwise, there are no solid lines between x, y, and z and so these three points define a dashed triangle. ■

This is problem is simple, but it has some big cousins. For example, how many people are necessary to ensure that some subset of *six* all know each other or are all strangers? The great mathematician Paul Erdös considered this problem so difficult that if aliens demanded an answer from humankind and all their computers within a year, he would recommend just throwing in the towel and attacking the aliens preemptively. Hey, it could happen.

2 Definitions

A nuisance in first learning graph theory is that there are so many definitions. They all correspond to intuitive ideas, but can take a long time to absorb. Worse, the same thing often has several names and can even have several equivalent definitions!

2.1 Simple Graphs

A *graph* is a pair of sets (V, E). The elements of V are called *vertices*. The elements of E are called *edges*. Each edge is a pair of distinct vertices.

Graphs are also sometimes called *networks*. Vertices are also sometimes called *nodes*. Edges are sometimes called *arcs*. Graphs can be nicely represented with a diagram of dots and lines as show in Figure 2

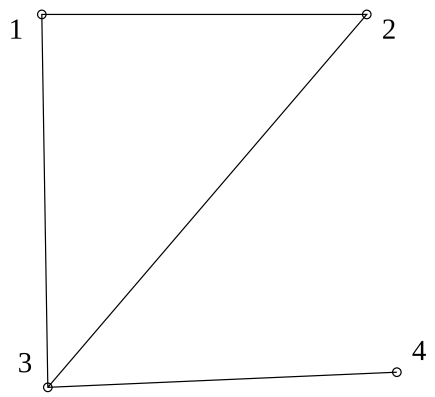


Figure 2: *This is a picture of a graph $G = (V, E)$. There are 4 vertices and 4 edges. The set of vertices V is $\{1, 2, 3, 4\}$. The set of edges E is $\{(1, 2), (1, 3), (2, 3), (3, 4)\}$. Vertex 1 is adjacent to vertex 2, but is not adjacent to vertex 4.*

As noted in the definition, each edge $(u, v) \in E$ is a pair of distinct vertices $u, v \in V$. Edge (u, v) is said to be *incident* to vertices u and v. Vertices u and v are said to be *adjacent* or *neighbors.* Phrases like, "an edge joins u and v" and "the edge between u and v" are comon.

A computer network is can be modeled nicely as a graph. In this instance, the set of vertices V represents the set of computers in the network. There is an edge (u, v) if there is a direct communication link between the computers corresponding to u and v.

2.2 Not Simple Graphs

There are actually many variants on the definition of a graph. The definition in the preceding section really only describes *simple graphs.* There are many ways to complicate matters.

Multigraphs

In a simple graph, there are either zero or one edges joining a pair of vertices. In a *multigraph*, multiple edges are permitted between the same pair of vertices.

Self-Loops

In a simple graph, each edge is a pair of distinct vertices. A *self-loop* is an edge that connects a vertex to itself. Figure 3 depicts a multigraph with self-loops.

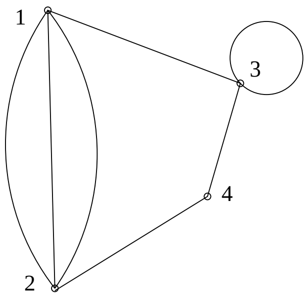


Figure 3: *This is a picture of a multigraph with a self-loop. In particular, there are three edges connecting vertices 1 and 2 and there is a self-loop on vertex 3.*

Directed Graphs

In a *directed graph* or *digraph*, edges are regarded not as lines, but as arrows. (The technical difference is that in a simple graph each edge is an unordered pair of vertices, whereas in a directed graph each edge is an ordered pair.) Figure 4 depicts a directed graph.

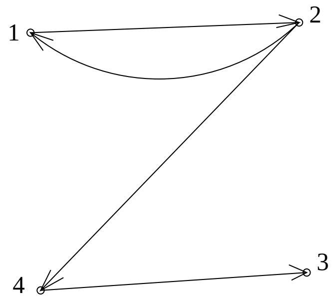


Figure 4: *This is a picture of a directed graph or digraph.*

Weighted Graphs

Sometimes it is useful to associate a number with each edge in a graph. These numbers are usually called *weights*. The weight of an edge (u, v) is often denoted $w(u, v)$. (If there is no edge between vertices u and v, then often $w(u, v)$ is infinity or zero.)

3 Graphs in the Real World

There are many real-world phenomena other than computer networks that can be described nicely by graphs. Here are some examples:

Airline Connections Here the vertices are airports and edges are flight paths. We could indicate the direction that planes fly along each flight path by using a directed graph. We could use weights to convey even more information. For example, $w(i, j)$ might be the distance between airports i and j, or the flying time between them, or even the air fare.

Precedence Constraints Suppose you have a set of jobs to complete, but some must be completed before others are begun. (For example, Atilla advises you always pillage *before* you burn.) Here the vertices are jobs to be done. Directed edges indicate constraints; there is a directed edge from job u to job v if job u must be done before job v is begun.

Program Flowchart Each vertex represents a step of computation. Directed edges between vertices indicate control flow.

Two-Player Game Tree All of the possibilities in a board game like chess can be represented in a graph. Each vertex stands for one possible board position. (This is a very big graph!) An example is shown in Figure 5.

Some graphs appear so frequently that they have names. The most important examples are shown in Figure 6.

4 Adjacency Matrices

We defined a graph in terms of a set of vertices and a set of edges. A graph can also be described by a matrix.

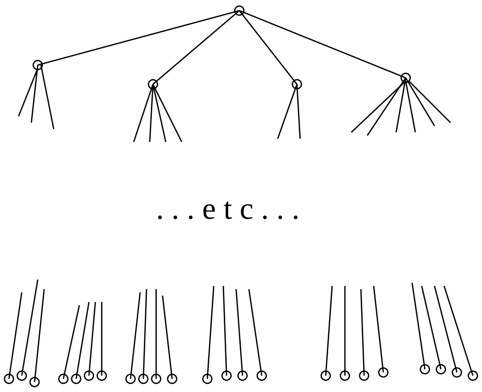


Figure 5: *This is a picture of a game tree. The topmost vertex represents the starting position. The edges descending from this vertex represent possible moves by the first player. The second tier vertices represent board positions after one move, and the edges below represent possible moves by the second player. A complete game can be represented by a path from the top vertex to a "leaf" vertex at the bottom of the diagram.*

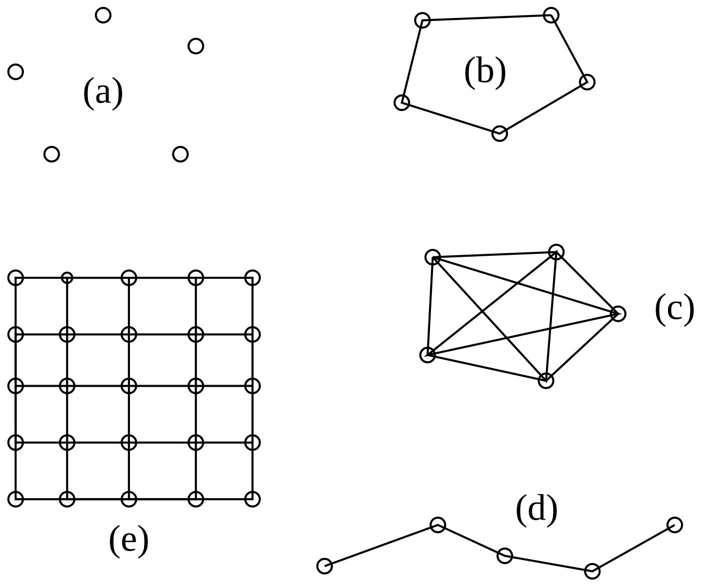
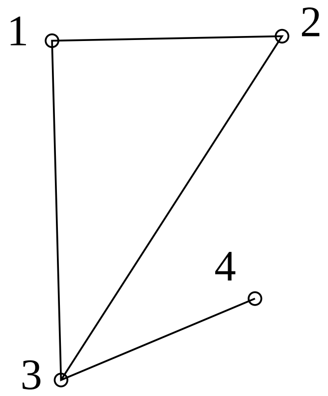


Figure 6: *The types of graph shown here are so common that they have names. (a) The empty graph on five vertices,* E_5*. An empty graph has no edges at all. (b) The cycle on five vertices,* C_5*. (c) The complete graph on five vertices,* K_5*. A complete graph has an edge between every pair of vertices. (d) A five-vertex line graph. (e) A* 5×5 *2-dimensional mesh.*

The *adjacency matrix* of a graph with n vertices is an $n \times n$ matrix. The entry in row i and column j of the matrix is a 1 if there is an edge connecting vertex i to vertex j and is 0 otherwise. An example is shown in Figure 7.



$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Figure 7: *This is a graph with 4 vertices and its associated 4x4 adjacency matrix. The entry in row i and column j of the matrix is a 1 if there is an edge (i, j) in the graph. All other entries are 0.*

Some variants of simple graphs can also be described well by matrices. For example, a weighted graph is naturally associated with a matrix that $w(i, j)$ as the entry in row i and column j.

5 Properties of Graphs

When one is confronted by a new graph– say in a dark alley– there is often a need to study properties at a higher-level than, "Is vertex u connected to vertex v?" Some questions that might arise are, "How many edges are incident to each vertex?", "Is the graph all in one piece or in several pieces?", "How can we color the vertices?" (The last may seem odd, but comes up surprisingly often; we'll see why in a minute.) This section discusses some of these higher-level properties of graphs.

5.1 Vertex Degree

The *degree* of a vertex is the number of incident edges. The degree of vertex v is often denoted $d(v)$. For example, referring to Figure 6, every vertex in an empty graph has degree 0, but every vertex in a cycle has degree 2. A look at the other examples suggests the following theorem:

Theorem 5.1 The sum of the degrees of the vertices in a graph is even.

Proof. Since every edge has two endpoints, every edge increases the sum of the degrees by two. This implies that the sum of the degrees is equal to twice the number of edges, and so is even. ■

Here is a puzzle that can be addressed with graphs. There is a party. Some people shake hands an even number of times and some shake an odd number of times. Show that an even number of people shake hands an odd number of times.

We can represent the party by a graph. (Yeah, right.) Each person is represented by a vertex. If two people shake hands, then there is an edge between the corresponding vertices. This reduces the problem to the following theorem:

Theorem 5.2 In every graph, there are an even number of vertices of odd degree.

Proof. By the preceding theorem, the sum the degrees of all vertices is even:

$$\begin{aligned}\sum_{v \in V} d(v) &= \mathit{even} \\ &= \sum_{v \in V, d(v)\ \mathit{odd}} d(v) + \sum_{v \in V, d(v)\ \mathit{even}} d(v)\end{aligned}$$

The second summation on the bottom line must be even, since it a sum entirely of even values. Since the whole expression on the bottom line must be even, the summation of odd values must contain an even number of terms. This implies that there are an even number of vertices with odd degree. ■

5.2 Chromatic Number

Time to discuss final exams. The MIT Schedules Office needs to assign a time slot for each final. This is not easy, because some students are taking several classes with finals, and a student can take only one test during a particular time slot. The Schedules Office wants to avoid all conflicts, but wants to make the exam period as short as possible.

This scheduling problem can be represented by a graph. Let each vertex represent a course. Put an edge between two vertices if there is some student that is taking both courses. Identify each possible time slot with a color. For example, Monday 9-12 is red, Monday 1-4 is blue, Tuesday 9-12 is green, etc.

If a student is taking two courses, then there is an edge between two vertices. If these vertices are the same color, then there is a conflict because the student has to take exams for the two courses at the same time. Therefore, the problem is to color each vertex of the graph so that no adjacent vertices have the same color and so that number of colors used is the minimum possible. An example is shown in Figure 8.

In general, the minimum number of colors needed to color the vertices of a graph G so that no two adjacent vertices are the same is called the *chromatic number* and is written $\chi(G)$[1]. For example, if G is the 3-cycle or triangle graph, then $\chi(G) = 3$.

In general, finding the chromatic number of a graph is very hard. However, we can at least put some upper bounds on the chromatic number. For example, the chromatic number of an n-vertex graph is certainly at most n, since every vertex could be assigned a different color.

[1]This is the Greek letter "chi" pronounced "key" by the ancient Greeks and "kie" by mathematicians and frats.

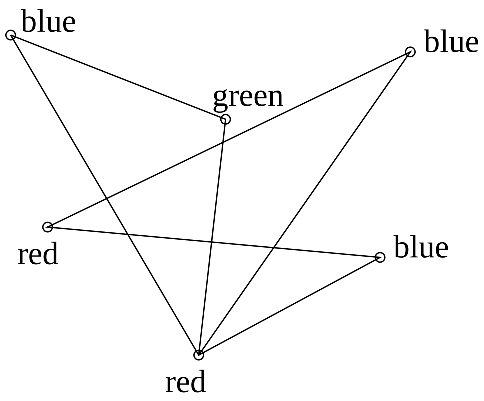


Figure 8: *This graph represents the exam scheduling problem. Each vertex stands for a course. An edge between two vertices indicates that a student is taking both courses and therefore the exams can not be scheduled at the same time. Each exam time slot is associated with a color. A schedule that creates no conflicts corresponds to a coloring of the vertices such that no adjacent vertices receive the same color.*

Suppose that we tried to do better by coloring vertices in an arbitrary order, but only using a new color only when forced. This strategy lies behind the following recursive algorithm. The input is a graph, and the output is a graph with appropriately colored vertices.

- Pick a vertex v. Remove v and all incident edges from the graph.
- If the graph is not empty, color the remainder recursively.
- Add v back to the graph. Color v differently from all neighbors, using a new color only if necessary.

Theorem 5.3 The preceding algorithm colors a graph with at most $p + 1$ colors where p is the maximum degree of any vertex.

This theorem implies, for example, that a graph with thousands of vertices, each of degree 3, requires at most 4 colors. The proof, is surprisingly easy:

Proof. The proof is by induction. Let $P(n)$ be the predicate that the preceding algorithm colors every graph in which every vertex has degree at most p using at most $p + 1$ colors.

In the base case, $P(1)$, there is a single vertex with degree zero. In this case, $p = 0$ and the algorithm requires $p + 1 = 1$ colors.

In the inductive step, assume $P(n)$ to prove $P(n + 1)$. Let G' be the graph obtained from G by removing vertex v and incident edges. No vertex in G' has degree greater than p, since removing a vertex and incident edges can not increase the degree of any other vertex. By induction, the algorithm (applied recursively) colors G' with at most $p + 1$ colors. Now we add back vertex v. Since v has at most p neighbors and there are $p + 1$ colors available, there is always one color left over for vertex v. Therefore, the algorithm colors G with at most $p + 1$ colors. ■

5.3 Connectivity

Is a graph all in one piece or composed of several pieces? To walk from one vertex to another, how many edges must one cross? Are there many different routes or just one? These are questions about connectivity.

Definitions

In a graph, a *path* from a vertex u to a vertex v is a sequence of edges $(u, x_1), (x_1, x_2), \ldots, (x_k, v)$. A path may contain the same edge or vertex multiple times.

In a *simple path*, no vertex or is crossed more than once. However, as a special case, the initial and final vertices may be the same. In other words, $u, x_1, x_2, \ldots, x_k$ are all distinct vertices, and $x_1, x_2, \ldots, x_k, v$ are all distinct vertices, but $u = v$ is permitted.

There is a path from u to v iff there is a path from v to u; one path is obtained from the other by reversing the sequence of edges. The same statement holds for simple paths.

A *circuit* is a path from a vertex back to itself, i.e. a sequence of edges $(u, x_1), (x_1, x_2), \ldots, (x_k, u)$. A *cycle* is a simple path from a vertex back to itself.

To summarize, a path is to a circuit as a simple path is to a cycle; both of the former may contain the same edge or vertex multiple times, but this is ruled-out in the latter.

A graph is *connected* if there is a path between every pair of distinct vertices. For example, referring back to Figure 6, the empty graph is disconnected, but all others shown are connected.

A *subgraph* of a graph $G = (V, E)$ is a graph $G' = (V', E')$ such that $V' \subseteq V$ and $E' \subseteq E$. For example, a triangle is a subgraph of K_4 (the complete graph on four vertices), but not of C_4 (a cycle with four vertices).

A *connected component* of a graph is a subgraph that is connected and maximal. This is, if any more vertices are added to the subgraph, then it is no longer connected. For example, every connected graph has exactly one connected component. The empty graph on n vertices has n connected components. A graph with three connected components is shown in Figure 9.

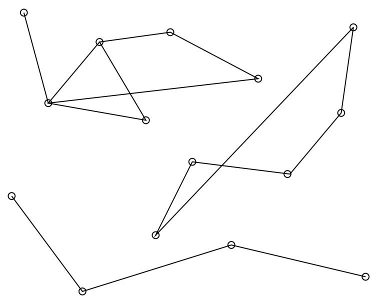


Figure 9: *This is a picture of a graph with 3 connected components.*

A False Theorem about Connectivity

If a graph is connected, then every vertex must be connected to some other vertex. Is the converse of this statement true? If every vertex is connected to some other vertex, then is the graph connected? The answer is no. In fact, the graph with three connected components showin in Figure 9 is a counterexample. So what is wrong with the following proof?

False Theorem 5.4 If every vertex in a graph is adjacent to another vertex, then the graph is connected.

Nothing helps a false proof like a good picture; see Figure 10.

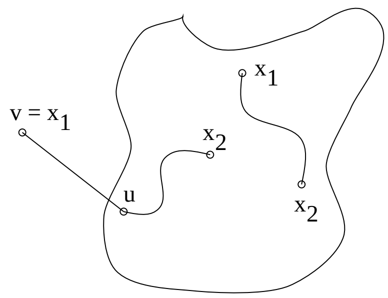


Figure 10: *This picture accomanies the false proof. Two situations are depicted. In one, vertices x_1 and x_2 both lie among the first n vertices, and so there is a connecting path by induction. In the second, $v = x_1$ and x_2 is among the first n vertices. In this case there is a connecting path because there is an edge from v to u and a path from u to x_2 by induction.*

Proof. The proof is by induction. Let $P(n)$ be the predicate that if every vertex in an n-vertex graph is adjacent to another vertex, then the graph is connected. In the base case, $P(1)$ is trivially true because there only one vertex.

In the inductive step, we assume $P(n)$ to prove $P(n+1)$. Start with an n-vertex graph; this is connected by induction. Now add a vertex v. By assumption v is connected to one of the first n vertices; call that one u.

Now we must show that for every pair of distinct vertices x_1 and x_2, there is a path between them. If both x_1 and x_2 are among the first n vertices, then there is a path by induction. Otherwise, one of the vertices is v (say x_1) and the other is among the first n vertices. These are connected by the path from x_1 to u to x_2 as shown in the figure. ■

The error is in the early statement "Start with an n-vertex graph; this is connected by induction." The induction hypothesis does not say that every n-vertex graph is connected, but only, "*if every vertex in an n-vertex graph is adjacent to another vertex,* then the graph is connected".

The core problem here is called "buildup error" and is a common source of mistakes in inductive proofs.

Buildup error occurs when the induction hypothesis $P(n)$ is itself an implication of the form $A(n) \Rightarrow B(n)$. In this instance, $A(n)$ is the statement, "every vertex in an n-vertex graph is adjacent to another vertex" and $B(n)$ is the statement, "the graph is connected".

In the inductive step of a correct proof, we assume $A(n) \Rightarrow B(n)$ in order to prove $A(n+1) \Rightarrow B(n+1)$. In an erroneous proof, we assume $B(n)$ in order to prove $A(n+1) \Rightarrow B(n+1)$. This is precisely what happened; we assumed, "the graph is connected" rather than, "if every vertex in an n-vertex graph is adjacent to another vertex, then the graph is connected".

A True Theorem about Connectivity

If a graph has too few edges, then there can not be a path between every pair of vertices. More generally, a graph with a very small number of edges ought to have many connected components. (Remember, "many connected components" does not mean "very connected"; rather, it means "broken into many pieces"!) The following theorem generalizes these observations.

Theorem 5.5 A graph with k edges and n vertices has at least $n - k$ connected components.

For example, a graph with 0 edges and n vertices has at least $n - 0 = n$ connected components. A graph with 100 vertices and 35 edges must have 100 - 35 = 65 connected components.

The theorem is proven by induction on k. This may seem odd, because for $k > n$, the theorem says that the number of connected components is greater than some negative number! This is useless, but certainly true, so there is no harm.

Proof. The proof is by induction.

Let $P(k)$ be the predicate that an n-vertex graph with k edges has at least $n - k$ connected components. In the base case, $P(0)$ holds because an n-vertex graph with 0 edges does have at least $n - 0 = n$ components.

In the inductive step, assume that an n-vertex graph with k edges has at least $n - k$ connected components to prove that an n-vertex graph with $k + 1$ edges has at least $n - k - 1$ connected components. Let G be a graph with n vertices and $k + 1$ edges. Remove one edge e. The resulting graph has n vertices and k edges. By induction there are at least $n - k$ connected components.

There are two possibilities when we replace the edge e. First, if e is incident to two vertices in the same connected component, then the number of connected components is unchanged at $n - k$. Otherwise, if e is incident to vertices in two different connected components, then the two become a single connected component. This reduces the number of connected components by one to $n-k-1$. In both cases, the number of connected components is at least $n - k - 1$ as claimed. ■

If a graph has fewer than $n - 1$ edges, then the theorem implies it has more than one connected component. Therefore, a graph must have at least $n - 1$ edges to be connected. In fact, there are graphs with $n - 1$ edges and a single connected component; the line graph of Figure 6 is an example.