

4 Graph Theory

Throughout these notes, a **graph** G is a pair (V, E) where V is a set and E is a set of unordered pairs of elements of V . The elements of V are called vertices and the elements of E are called edges. We typically denote by $V(G) = V$ the vertex set of G and $E(G) = E$ the edge set of G . If $u, v \in V(G)$, then u and v are adjacent if $\{u, v\} \in E(G)$. We refer to u and v as the endpoints of the edge $\{u, v\}$. It is convenient to draw $V(G)$ as a set of points in the plane and each edge as a curve between its endpoints in the plane, as in Figure 1(a). A **multigraph** is a pair (V, E) where V is a set and E is a multiset of unordered pairs of elements of V – we allow E to contain repetitions of the same unordered pair. In other words, more than one edge can join two vertices. Sometimes we even allow loops on a vertex – see Figure 1(b). A directed graph – **digraph** for short – is a pair (V, E) where V is a set and E is a set of ordered pairs of G . The elements of E are called arcs, and we draw them in the plane as curves with an arrow indicating the direction of the arc – see Figure 1(c).

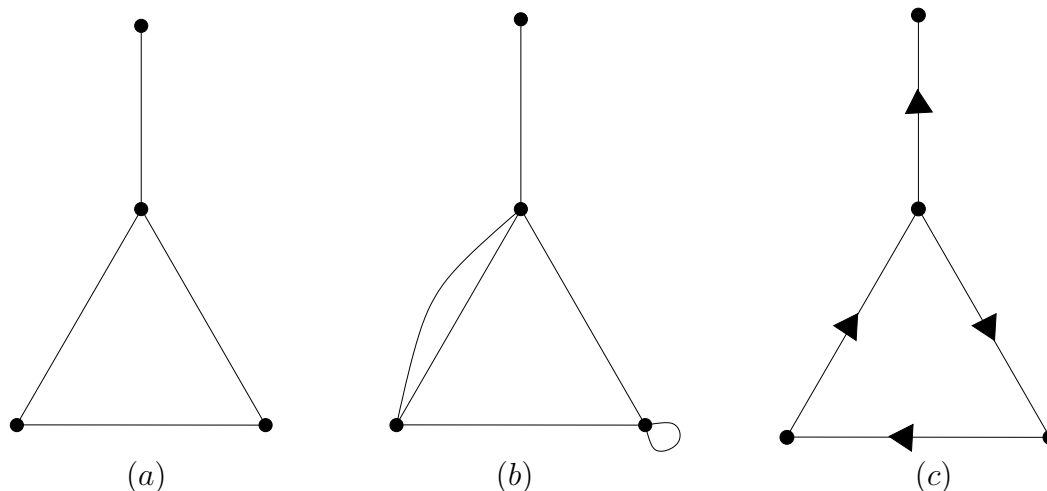


Figure 1 : Graphs, multigraphs and digraphs.

If the vertices of the leftmost figure are labelled $\{1, 2, 3, 4\}$ in clockwise order from the topmost vertex, then the edge set of that graph is $\{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 2\}\}$. Under the same labelling, the multigraph in Figure 1(b) would have edge multiset $\{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 2\}, \{4, 2\}, \{3, 3\}\}$. Finally, the digraph in Figure 1(c) would have arc set $\{(2, 1), (2, 3), (3, 4), (4, 2)\}$.

4.1 Graph theoretic terminology

In the next few short sections, we describe the notation that will be used in the remainder of the notes to refer to various properties of graphs.

4.1.1 Degrees and Neighbourhoods

The [neighborhood](#) of v in G is denoted $\Gamma_G(v)$, and is the set of vertices of G which are adjacent to v . The [degree](#) of a vertex v in a graph G is the number of edges incident with v , and is denoted $d_G(v)$, or just $d(v)$ for short. We write $\delta(G)$ and $\Delta(G)$ for the smallest and largest degree of a vertex in G , respectively. For example, if G is the graph in Figure 1(a), then $\delta(G) = 1$ and $\Delta(G) = 3$. The neighborhood of a set $X \subset V(G)$, which we denote by $\Gamma(X)$, is the set of vertices of $V(G) \setminus X$ with at least one neighbor in X . For sets $X, Y \subset V(G)$, we write $e(X, Y)$ for the set of edges of G with one endpoint in X and the other endpoint in Y .

An important fact involving the degrees of a graph G , which we will use on numerous occasions, is the [handshaking lemma](#):

Lemma 1 *For any graph $G = (V, E)$,*

$$\sum_{v \in V} d_G(v) = 2|E|.$$

Proof \triangleright When we add up the degrees of vertices of G , every edge of G is counted twice, once for each of its endpoints. ■

An important consequence of the handshaking lemma is that the number of vertices of odd degree in any graph must be even – otherwise the sum on the left above would be odd. So, for example, if you try to draw a graph on five vertices in which every vertex has degree three, you would fail. We refer to graphs where all the degrees are the same, say all degrees are equal to r , as r -regular graphs. The handshaking lemma gives an easy way to count the number of edges in a graph: it is just half the sum of the degrees of the vertices.

4.1.2 Basic classes of graphs

Let K_n denote the complete graph on n vertices: this is the graph consisting of all $\binom{n}{2}$ possible edges on n vertices. A bipartite graph is a graph G such that $V(G)$ can be partitioned into two sets A and B such that each edge of G has one endpoint in A and one endpoint in B . The sets A and B are called the parts of G . Let $K_{s,t}$ denote

the complete bipartite graph with s vertices in one part and t vertices in the other: this graph consists of all st possible edges between a set of size s and a set of size t . We refer to $K_{1,t}$ as a star. The n -dimensional cube, or n -cube, is the graph whose vertex set is the set of binary strings of length n , and whose edge set consists of pairs of strings differing in one position. A tree is a connected graph without cycles, and a forest is a vertex-disjoint union of trees and isolated vertices. The vertices of degree one in a tree or forest are often called leaves. A cycle is a connected graph all of whose vertices have degree two, and a path is a tree with exactly two vertices of degree one. Those vertices are sometimes called the endpoints of the paths, and the other vertices of the path are called internal vertices. Two paths are internally disjoint if they share no internal vertices.

The number of edges in K_n is $\binom{n}{2}$, and by the handshaking lemma, we can verify this: every vertex in K_n is joined to all $n - 1$ other vertices, so every vertex has degree $n - 1$. By the handshaking lemma

$$|E(K_n)| = \frac{1}{2} \sum_{v \in V} d(v) = \frac{1}{2} \sum_{v \in V} (n - 1) = \frac{1}{2} n(n - 1) = \binom{n}{2}.$$

For the n -dimensional cube, Q_n , there are 2^n vertices. Each vertex v is joined to n other vertices – namely flip one position in the string v to get strings adjacent to v , and there are n possible positions to do a flip. So every vertex of the n -cube has degree n , and

$$|E(Q_n)| = \frac{1}{2} \sum_{v \in V} d(v) = \frac{1}{2} \sum_{v \in V} n = \frac{1}{2} 2^n n = n2^{n-1}.$$

4.1.3 Subgraphs

If H and G are graphs and $V(H) \subset V(G)$ and $E(H) \subset E(G)$, then H is called a subgraph of G . To denote that H is a subgraph of G , we write $H \subset G$. If in addition $V(H) = V(G)$ then H is called a spanning subgraph of G . If $G = (V, E)$ is a graph and $X \subset V$ and $L \subset E$, then $G - X$ denotes the graph whose vertex set is $V \setminus X$ and whose edge set consists of all edges disjoint from X . Also $G - L$ denotes the graph $(V, E \setminus L)$. In the case that $X = \{x\}$, we write $G - x$ instead of $G - X$ and if $L = \{e\}$ we write $G - e$ instead of $G - \{e\}$. For example, if G is the graph in Figure 1(a) and x is the vertex of degree three in Figure 1(a), and y is the vertex of degree one, then $G - x$ consists of a disjoint union of K_1 and K_2 , and $G - y$ is K_3 .

4.1.4 Walks and Paths

A **walk** in a graph $G = (V, E)$ is an alternating sequence of vertices and edges, whose first and last elements are vertices, and such that each edge joins the vertices immediately preceding it and succeeding it in the sequence. For example,

$$a\{a, d\}d\{d, e\}e\{e, a\}a\{a, d\}d$$

is a walk in the graph in Figure 3. If the vertices of a walk are all distinct, then it is referred to as a **path**. The length of a walk is the number of steps taken in the walk. The first and last vertices of a walk are called end vertices and the remaining vertices are called internal vertices. For example, the walk given above is not a path, but $a\{a, d\}d\{d, e\}e$ is a path. The walk above has length four. For convenience, we often omit the edges and list the vertices in order along the path, such as ade instead of $a\{a, d\}d\{d, e\}e$. If the endpoints of a walk are u and v , then we say the walk is a uv -walk. If the first and last vertex of the walk are the same, then the walk is referred to as a closed walk. A cycle is a closed walk with the same number of edges as vertices – every vertex of a cycle has degree two. For example, in Figure 3, the different cycles are $adea$, $abea$, $bcdeb$, $adeba$, $adcba$, $aedcba$, and $aebcda$.

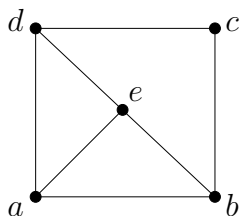


Figure 3 : Walks, paths and cycles.

4.2 Connected graphs

A graph is connected if any pair of vertices in the graph are joined by at least one path. If a graph is not connected, we say it is disconnected. The components of a graph $G = (V, E)$ are the maximal (i.e. with as many edges as possible) connected subgraphs of G . The graphs in Figure 1(a) and Figure 3 are connected, whereas the graph below has three components.

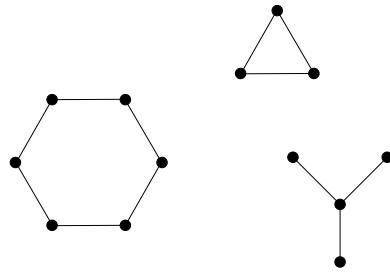


Figure 4 : Components.

The simplest type of connected graph is a tree: by definition, a tree is a connected graph without cycles – a connected acyclic graph. To describe the structure of trees, we define the notion of a bridge. A bridge of a graph G is an edge $e \in E(G)$ such that $G - e$ has more components than G . For example, in Figure 1(a), the top edge e is a bridge since G has one component but $G - e$ has two components. It is easy to spot the bridges of a graph, using the following lemma:

Lemma 2 *An edge $e \in E(G)$ is a bridge if and only if e is not contained in any cycle in G .*

Proof \triangleright If e is contained in a cycle C of G , then $C - e$ is a path joining the ends of e . But that means $G - e$ is connected, so e could not have been a bridge. \blacksquare

Since a tree has no cycles, every edge of a tree must be a bridge. We can now characterize which graphs are trees in a few ways.

Proposition 3 *Each of the following is equivalent to a graph G being a tree:*

1. *The graph G is connected and acyclic.*
2. *The graph G is connected and every edge of G is a bridge.*
3. *The graph G is connected and has $|V(G)| - 1$ edges.*

Proof \triangleright Clearly Proposition 3.1 is the definition of G being a tree. Since a connected graph is acyclic if and only if every edge of the graph is a bridge, by the last lemma, Proposition 3.1 and Proposition 3.2 are equivalent. We proved Proposition 3.1 implies Proposition 3.3 by strong induction on the number of vertices in the tree, so it remains to show that Proposition 3.3 implies Proposition 3.1. To see that, if G is connected with $|V(G)| - 1$ edges, we remove an edge of any cycle and that does not disconnect G , by Lemma 2. We continue removing edges of G in cycles until all the cycles are gone. But then the remaining graph T is connected and acyclic, so must be a tree.

Since it has $|V(G)|$ vertices, we know it must have $|V(G)| - 1$ edges. But G itself has $|V(G)| - 1$ edges, so $G = T$. ■

The last part of the proof of this proposition is important. It says that in any connected graph G , while there is a cycle, pick an edge of the cycle and remove it. By Lemma 2, we did not disconnect the graph, so if we repeat this procedure we eventually obtain a spanning subgraph of G which is acyclic and connected – a tree. We call this a [spanning tree](#) of the graph. In general, a graph has many spanning trees.

Proposition 4 *Any connected graph contains a spanning tree.*

The proof gives a fairly quick way to find a spanning tree of a graph: search for a cycle and remove an edge of the cycle, and repeat until there are no cycles left.

4.3 Block Decomposition*

We just gave three equivalent characterizations of trees. In general, we would like to describe how to build connected graphs. The main result will be the block decomposition theorem. We require some definitions. A [cut vertex](#) of a graph G is a vertex x such that $G - x$ is disconnected. A [block](#) of a graph is a maximal connected subgraph with no cut vertex – a subgraph with as many edges as possible and no cut vertex. So a block is either K_2 or is a graph which contains a cycle. For example in a tree, every block is K_2 . The block decomposition of a graph is just the set of all the blocks of the graph. An example of a block decomposition is shown below.

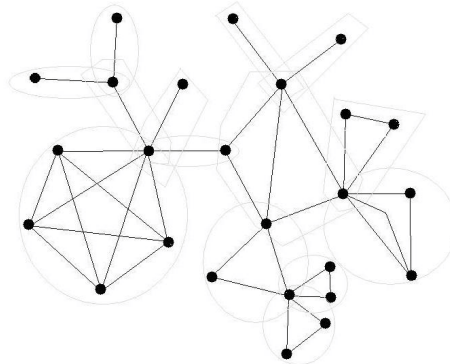


Figure 5 : Blocks.

In the picture, there are fourteen blocks. Seven of the blocks are K_2 , four of the blocks are triangles, one of the blocks is K_5 , and there are two other blocks. The block decomposition theorem says that block decompositions of graphs have a “tree-like structure”. To make this precise, given a graph G , we form a new graph \mathcal{B} where the vertices of \mathcal{B} consist of all cut vertices of G and also all blocks of G , and where a block is joined to all cut vertices of G contained in it. An example of this graph is shown below for the figure above:

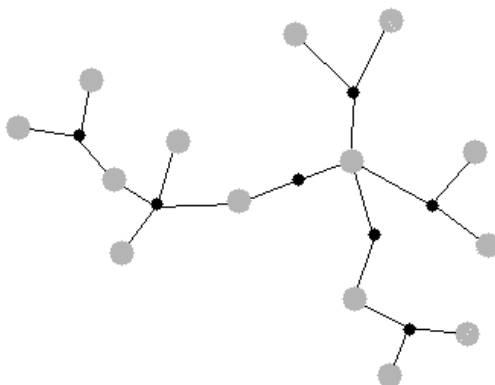


Figure 6 : The graph \mathcal{B} .

In the figure, the black vertices represent cut vertices of G , and the grey vertices represent blocks of G . Here is the block decomposition theorem:

Theorem 5 *Let G be a connected graph. Then \mathcal{B} is a tree.*

Proof \triangleright By adding edges inside the blocks of G , we do not change \mathcal{B} , so we can assume every block of G is a complete graph. Since G is connected, clearly \mathcal{B} is connected too. Now we show \mathcal{B} has no cycles. The vertices of a cycle $\mathcal{C} \subseteq \mathcal{B}$ are alternately blocks of G and cut vertices of G , by definition of \mathcal{B} . This is shown in the figure below:

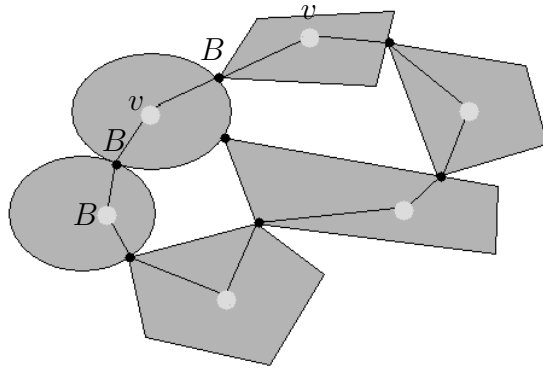


Figure 7 : Cycle in \mathcal{B} .

In the figure, the blocks are shown as grey dots and labelled B and the cut vertices are black dots labelled v . Let the cut vertices of G in order along \mathcal{C} be $v_0, v_1, \dots, v_k, v_0$. Then $v_0v_1v_2 \dots v_kv_0$ is a cycle $C \subseteq G$. If $B \in \mathcal{C}$, then $B \cup C$ is a subgraph of G consisting of the complete graph B together with the cycle C containing an edge of B and at least one edge not in B . Therefore $B \cup C$ has no cut vertex, and must be a block of G . However, this contradicts the definition that B is block. ■

Using this lemma, we give a first example of a structure theorem in graph theory. Define a [theta graph](#) to be any graph consisting of the union of three internally disjoint paths between two points.

Proposition 6 *Let G be a connected graph containing no theta graph. Then every block of G is a cycle or K_2 and G is a tree of cycles and K_2 s, as shown in Figure 8 below.*

Proof \triangleright Let B be a block of G . If $B \neq K_2$, then B contains a cycle, C . If $B \neq C$, then there is a path P in B such that $P \cup C$ is a theta graph. Therefore $B = K_2$ or B is a cycle. We know by the last result that G is then a tree of cycles and K_2 s. ■

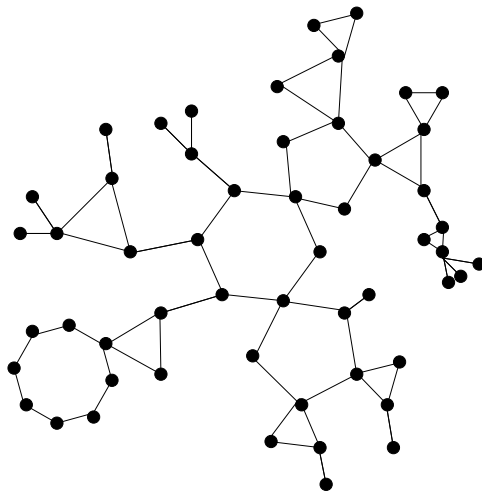


Figure 8 : Tree of cycles and K_2 s

4.4 k -connected graphs

A **cut** of a connected graph G is a set of vertices or edges whose removal from G gives a disconnected graph. A graph G is **k -connected** if every cut of G has size at least k , and **k -edge-connected** if every edge cut of G has size at least k . So far we have characterized trees and connected graphs using block decompositions. The situation for k -connected graphs is more complicated when $k \geq 2$. Fortunately, **Menger's Theorems** give a necessary and sufficient condition for a graph to be k -connected or k -edge-connected. Let u and v be vertices in a graph G , and let P and Q be uv -paths. Then P and Q are **internally disjoint** if the only vertices they have in common are u and v . A **uv -separator** is a set $W \subset V(G) \setminus \{u, v\}$ such that u and v are in distinct components of $G - W$. Finally, let $\kappa(u, v)$ denote the minimum size of a uv -separator in G .

Theorem 7 (Menger's Theorem – Vertex Form)

Let u and v be non-adjacent vertices in a graph G . Then the maximum number of pairwise vertex disjoint uv -paths in G is $\kappa(u, v)$. In particular, a graph is k -connected if and only if each pair of its vertices is connected by at least k pairwise internally disjoint paths.

It is important to note that condition that u and v are not adjacent in this theorem. Clearly, there is no uv -separator if u and v are adjacent. The vertex-form of Menger's Theorem has an edge-form. A set $L \subset E(G)$ is a uv -edge-separator if u and v are in different components of $G - L$. Let $\lambda(u, v)$ denote the minimum size of a uv -edge-separator in G .

Theorem 8 (Menger's Theorem – Edge Form)

Let u and v be any vertices in a graph G . Then $\lambda(u, v)$ equals the maximum number of pairwise edge-disjoint uv -paths in G . In particular, a graph is ℓ -edge-connected if and only if each pair of its vertices is connected by at least ℓ pairwise edge-disjoint paths.

We will see how these theorems follow readily from the max-flow min-cut theorem. Note that it is possible to prove these theorems directly by structural arguments, but this is beyond the scope of this course. It is an interesting consequence of Menger's Theorem that if G is a graph with minimum degree $\delta(G)$, and $\kappa(G)$ and $\lambda(G)$ are the sizes of the smallest vertex cut and edge cut of G respectively, then

$$\kappa(G) \leq \lambda(G) \leq \delta(G).$$

This is intuitively true in the sense that we can do more damage to a graph by removing vertices than edges, so we expect $\kappa(G) \leq \lambda(G)$. We prove this result as follows

Corollary 9 Let G be any graph which is not a complete graph. Then $\kappa(G) \leq \lambda(G) \leq \delta(G)$.

Proof \triangleright To see that $\lambda(G) \leq \delta(G)$, note that if we remove all the edges around a vertex of minimum degree, we disconnected G and therefore that set of $\delta(G)$ edges is an edge cut of G . Since $\lambda(G)$ is the smallest possible size of an edge cut of G , we get $\lambda(G) \leq \delta(G)$. To prove $\kappa(G) \leq \lambda(G)$, we use the two versions of Menger's Theorem given above. By definition,

$$\kappa(G) = \min\{\kappa(u, v) : u, v \in V(G) \text{ are not adjacent}\}.$$

Similarly

$$\lambda(G) = \min\{\lambda(u, v) : u, v \in V(G)\}.$$

Now by Menger's Theorems, if u and v are not adjacent, then $\kappa(u, v) \leq \lambda(u, v)$ since a set of internally disjoint uv -paths is also a set of edge-disjoint uv -paths. If u and v are adjacent – $e = \{u, v\} \in E(G)$ – then we remove the edge between them. Clearly that reduces $\lambda(u, v)$ by exactly one, and also reduces the maximum number of edge-disjoint uv -paths by one. So we get

$$\kappa(G - e) \leq \lambda(G) - 1.$$

However, $\kappa(G - e) \geq \kappa(G) - 1$ so we get

$$\kappa(G) - 1 \leq \lambda(G) - 1$$

and the proof is complete: $\kappa(G) \leq \lambda(G)$. ■