Graph Theory

Author: Isaac FEI

Preface

When writing this book, I mainly refer to (Bondy and Murty 1976), which covers both theoretical results and crucial applications in graph theory.

Contents

Chapter 1 Basic Concepts of Graphs **1**

1.1 Isomorphism 1

1.2 Vertex Degrees 2

1.3 Graph Realization Problem 3

1.4 Paths and Connection 4

1.5 Cycles 9

1.6 Shortest Paths and Dijkstra's Algorithm 12

Chapter 2 Trees **18**

2.1 Definition of Trees 18

References **21**

Index **22**

Chapter 1 Basic Concepts of Graphs

1.1 Isomorphism

Two graphs G and H are identical, written as $G = H$, if all their components are the same, that is, $V(G) = V(H)$, $E(G) = E(H)$ and $\psi_G = \psi_H$. Identical graphs of course share the same properties. However, a graph H does not necessarily have to be exactly G to preserve all its properties. The labels of the vertices and edges are immaterial.

Definition 1.1.1

Two graphs G and H are said to be isomorphic, written as $G \cong H$, if there exist bijections $\theta : V(G) \to V(H)$ and $\phi : E(G) \to E(H)$ such that

$$\psi_G(e) = uv \implies \psi_H(\phi(e)) = \theta(u)\theta(v) \tag{1.1}$$

*The ordered pair (θ, ϕ) is called an **isomorphism** between G and H.* ♣

(Bondy and Murty 1976) includes the reverse direction of (1.1) in the definition, that is,

$$\psi_G(e) = uv \iff \psi_H(\phi(e)) = \theta(u)\theta(v)$$

But the reverse direction is redundant. To see this, we suppose that $\psi_H(\phi(e)) = \theta(u)\theta(v)$ and $\psi_G(e) = xy$. By (1.1), we have $\psi_H(e) = \theta(x)\theta(y)$. It then follows that $\theta(u)\theta(v) = \theta(x)\theta(y)$. We have either $\theta(u) = \theta(x)$, $\theta(v) = \theta(y)$, or $\theta(u)\theta(y)$, $\theta(v) = \theta(x)$. Because θ is a bijection, either $u = x$, $v = y$, or $u = y$, $v = x$. Either way, we have $uv = xy$. Therefore, $\psi_G(e) = xy = uv$, which proves the reverse direction $\Leftarrow$.

For simple graphs, there is no need to find a bijection between edges once the bijection θ between vertices is established.

Proposition 1.1.1

Let G and H be simple graphs. Then $G \cong H$ if and only if there exists a bijection $\theta : V(G) \to V(H)$ such that

$$uv \in E(G) \implies \theta(u)\theta(v) \in E(H) \tag{1.2}$$

♠

Proof (Necessity) Suppose that there exist θ and ϕ satisfying (1.1). If $e = uv \in E(G)$, then by (1.1), $\psi_H(\phi(e)) = \theta(u)\theta(v)$, which implies $\theta(u)\theta(v) \in E(H)$.

(Sufficiency) Define $\phi : E(G) \to E(H)$ by

$$\phi(uv) = \theta(u)\theta(v)$$

We need to show ϕ is bijective. Suppose $\phi(uv) = \phi(xy)$. We have $\theta(u)\theta(v) = \theta(x)\theta(y)$. Applying a similar argument we used in the previous comments, we will finally obtain $uv = xy$, which means ϕ is injective. On the other hand, for any edge $f \in H$. Write $f = ij$ (i.e., $\psi_H(f) = ij$). Then because θ is bijective, there exist $u, v \in V(G)$ such that $\theta(u) = i$ and $\theta(v) = j$. Hence, $\phi(uv) = ij$, which implies ϕ is surjective.

If $\psi(e) = uv$, i.e., $e = uv \in E(G)$, then we have $\theta(u)\theta(v) \in E(H)$ by (1.2). Equivalently, $\psi_H(\phi(e)) = \theta(u)\theta(v)$. ■

A **complete bipartite graph** is a *simple* bipartite graph with bipartition (X, Y) in which each vertex in X is incident with each vertex in Y. That is, if $x \in X$ and $y \in Y$, then $xy \in E$. If $|X| = m$ and $|Y| = n$, we often use the symbol $K_{m,n}$ to denote this complete bipartite graph. (See Figure 1.1.) Note that this implicitly implies that the complete bipartite graph is unique in some way since we can represent it with a common symbol. Indeed, it is unique up to isomorphism, as we will show in the next proposition.

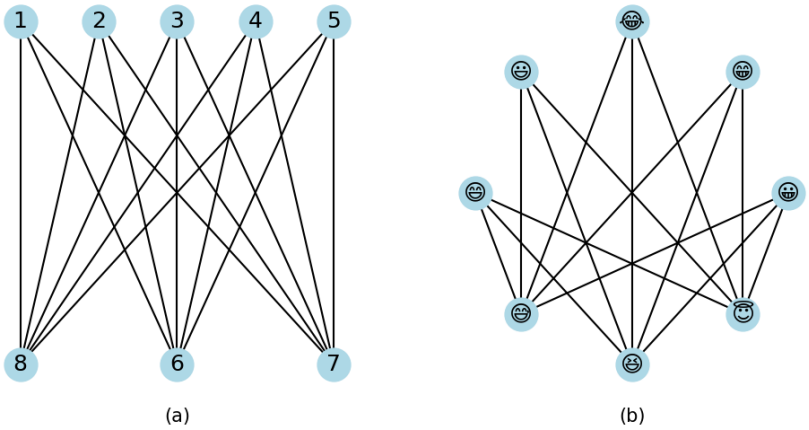

Figure 1.1: Both (a) and (b) are $K_{5,3}$.

Proposition 1.1.2

Let $G[X, Y]$ and $H[U, V]$ be two complete bipartite graphs with $|X| = |U|$ and $|Y| = |V|$. Then $G \cong H$. In other words, a complete bipartite graph is unique up to isomorphism if the sizes of its two vertex sets in bipartition are determined.

♠

Proof Since $|X| = |U|$ and $|Y| = |V|$, we can find a bijection $\theta : V(G) \to V(H)$ in such a way that θ maps each point in X onto U, and each point in Y onto V. Then for an edge $xy \in E(G)$, we have $\theta(x)\theta(y) \in E(H)$ since there has to be an edge connecting $\theta(x) \in U$ and $\theta(y) \in V$ by the definition of complete bipartite graphs. This proves $G \cong H$ by Proposition 1.1.1. ■

1.2 Vertex Degrees

Theorem 1.2.1

The sum of degrees of all vertices is twice the number of edges in any graph. That is,

$$\sum_{v \in V} \deg(v) = 2\,|E|$$

♡

Proof ■

Corollary 1.2.1

The number of vertices of odd degree is even in any graph. ♡

1.3 Graph Realization Problem

Let G be a graph with vertices $v_1, \ldots, v_n$. The **degree sequence** of G, as the name suggests, is a list of integers consists of vertex degrees. It is denoted by

$$\mathbf{d} = (\deg(v_1), \ldots, \deg(v_n))$$

And it is often written in non-increasing order.

Given an arbitrary list of non-negative integers $\mathbf{d}' = (d'_1, \ldots, d'_n)$, it is possible that we cannot find any graphs whose degree sequence is $\mathbf{d}'$. For example, no graphs have the following degree sequence:

$$(2, 2, 1, 1, 1)$$

Why? Note that $2 + 2 + 2 + 1 + 1 + 1 = 9$. Hence, by Theorem 1.2.1, we see that is is impossible to find such a graph since the sum of degrees should be an even number. Alternatively, one may also argue by Corollary 1.2.1 that the number of odd vertices (the last three vertices) is 3, which is an odd number.

So, what conditions must a list of non-negative integers must satisfy so that it is a degree sequence? The answer is simple, as the next proposition will show, the provided list of integers only need to satisfy condition given in Theorem 1.2.1, that is, all the integers need to sum up to an even number.

Proposition 1.3.1

A sequence $(d_1, \ldots, d_n)$ of non-negative integers is a degree sequence if and only if $\sum_{i=1}^{n} d_i$ is even. ♠

Proof ■

From the above proof, we see that the graph we construct is a multigraph since we mainly increase the vertex degrees by adding loops. This invites us to think can we construct a *simple graph* out of the given sequence of integers? This question is much more complecated. And we say that a sequence of integers is **graphical** if it is a degree sequence of some simple graph. The problem of finding such a simple graph is referred to as the **graph realization problem**.

1.3.1 Havel-Hakimi Theorem

The algorithm (Hakimi 1962) constructs a simple graph through a recursive procedure. The theorem it based on is referred to as Havel-Hakimi theorem. Before proving this theorem, we first introduce a lemma, which will be helpful in the later proof.

Lemma 1.3.1

Let G be a simple graph. If

1. $u_1v_1, u_2v_2 \in E$ *and*
2. $u_1v_2, u_2v_1 \notin E$,

then the graph

$$G' = G - \{u_1v_1, u_2v_2\} + \{u_1v_2, u_2v_1\}$$

has the same degree sequence as G. ♡

This lemma mainly says we can move two non-incident edges in G in a way that the degree sequence remains unchanged. Refer to Figure 1.2 to get more insight.

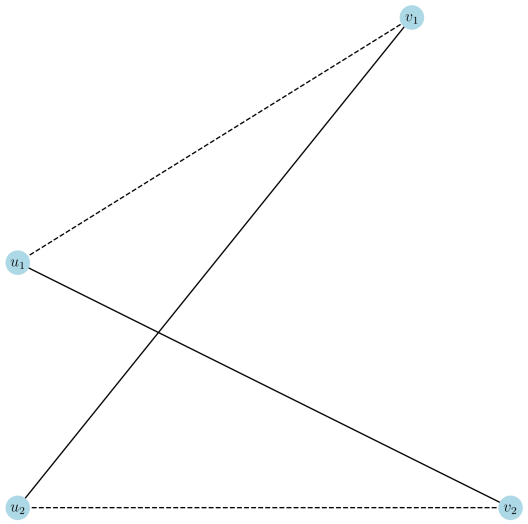

Figure 1.2: Dash lines are removed edges while solid lines are newly added.

Proof This result is evident. We note that the degrees of vertices u_1, v_1, u_2 and v_2 remain unchanged in G' since from the perspective of each of these four vertices, one incident edge is removed but a new one is added. And the degrees of all the other vertices in G' are also the same as in G. Hence, the degree sequence of G' is the same as G. ■

1.4 Paths and Connection

A **walk** in a graph is a sequence of edges $e_1 \cdots e_k$ joining a *nonempty* sequence of vertices $v_0v_1 \cdots v_k$, which is denoted by

$$v_0e_1v_1 \cdots e_kv_k \tag{1.3}$$

with each edge written after one of its end and followed by its other end. Note that though the sequence of vertices in a walk is required to be nonempty, the sequence of edges may be empty. And in that case, the walk contains only one vertex, say v_0, and it is called the **trivial walk**.

Note *The term **sequence** in mathematics often means an infinite sequence, which is essentially a*

*function defined on $\mathbb{N}^*$. However, in graph theory, we usually refer to sequence as a **finite** list of ordered elements.*

We call a walk W from v_0 to v_k a (v_0, v_k)-walk. The vertices v_0 and v_k are referred to as the **origin** and the **terminus** of that walk, respectively.

It should be emphasized that neither the edges nor the vertices in a walk are necessarily distinct. However, if all edges of walk W are distinct, we call W a **trail**. And if all vertices in W are distinct, it is then called a **path**. Of course, all edges in a path are also distinct since the vertices are.

Two vertices are said to be **connected** if there exists a path joining them. Otherwise, they are **disconnected**.

The length of a path P, written as $|P|$, is defined as the number of edges along it. Note that the length of a trivial path is zero since there are no edges.

If G is a simple graph, then we may write a walk simply as a sequence of vertices since there is one and only one edge joining each pair of consecutive vertices in the walk. For example, we write the (v_0, v_k)-walk in (1.3) as

$$v_0 v_1 \cdots v_k$$

with edges dropped.

Proposition 1.4.1

If there is a (u, v)-walk in G, then there is also a (u, v)-path in G. ♠

This can be proved easily using the following algorithm (Algorithm 1).

Algorithm 1: Extracting a Path From a Walk

Input: A walk $W = v_0 e_1 v_1 \cdots e_k v_k$
Output: A path P

```
initialize P as a sequence containing just one vertex v_0 ;
for i = 1, ..., k do
    if v_i is not in P then
        append e_i and v_i to P ;
    else
        remove all the vertices and edges after the vertex v_i from P ;
    end
end
```

Proposition 1.4.2

The number (v_i, v_j)-walks of length k in G is the (i, j)-th entry of the k-th power of the adjacency matrix A, i.e., A^k. ♠

Proof ■

Imagine removing one edge from the original graph G. Then we can obtain at most one more component by cutting in half one of G's components. See Figure 1.3.

To prove this, we first consider the following useful lemma.

Lemma 1.4.1

Let G be a connected graph, and e be one of its edges. Suppose that the two ends of e are x and y. Then every vertex v is either connected to x or y in $G - e$. ♡

Proof First, suppose that e is a loop. Since G is connected, every two vertices are connected by a path. And they remain connected in $G - e$ since a path cannot contain any loops. Hence, in this case, every vertex v is connected to both x and y in $G - e$.

Assume e is not a loop. If vertex v is no longer connected to x in $G - e$, then v must be originally connected to x in G by a path P containing the edge e. Note that P is of the form

$$P = v \cdots yex$$

Because the (v, y)-section of P does not traverse e, v is connected to y in $G - e$ by this (v, y)-section. Hence, we have shown that, in subgraph $G - e$, v must be connected to y if it is disconnected from x. This completes the proof. ■

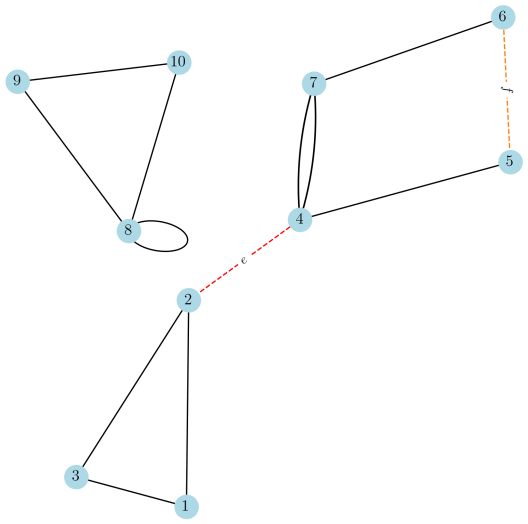

Figure 1.3: The graph G has 2 components. If we remove edge f, then $G - f$ still has 2 components. But if we remove edge e, then the remaining graph $G - e$ has 3 components.

Proposition 1.4.3

If $e \in E$, then we have

$$\omega(G) \leq \omega(G - e) \leq \omega(G) + 1 \tag{1.4}$$

♠

Inequalities (1.4) describe the result of cutting an edge in a compact way. The first inequality $\omega(G) \leq \omega(G - e)$ says the number of components may increase by cutting an edge. While the second inequality says this number will be increased by at most one.

Proof If e is a loop, then the conclusion is trivial. We assume $e = xy$ is not a loop, i.e., x and y are distinct, in the rest of the proof. Suppose that the components of G are $G[V_1], \ldots, G[V_\omega]$. Without loss of generality, we may also assume that $x, y \in V_1$.

(Case 1) Suppose that there exists a (x, y)-path in $G - e$, then x is still connected to y in $G - e$, i.e., $x \sim y$ in $G - e$. Pick an arbitrary vertex $v \in V_1$. By Lemma 1.4.1, we know either $v \sim x$ or $v \sim y$. Either way, by the transitivity, we have $v \sim x$ since $x \sim y$. Therefore,

$$v \sim x \quad \forall v \in V_1$$

This means V_1 remains a equivalent class in $G - e$. In this case, $\omega(G - e) = \omega(G)$.

(Case 2) We now consider the case where x is disconnected from y in $G - e$. For any vertex $v \in V_1$, by Lemma 1.4.1, we have either $v \sim x$ or $v \sim y$. But v cannot be connected to both x and y since x and y are assumed disconnected from each other. It then follows that V_1 can be partitioned into two equivalent classes, $[x]$ and $[y]$. In other words, V_1 is split into two components in $G - e$. Therefore, $G - e$ has one more component than that of G, and hence $\omega(G - e) = \omega(G) + 1$. ■

The idea of cutting an edge is quite helpful in many proofs since by doing so, we will probably divide the original graph into two smaller ones.

The edge if removed will indeed results in one more component deserves a name, as we shall define in the following.

Definition 1.4.1

*The edge e of G is called a **cut edge** if*

$$\omega(G - e) = \omega(G) + 1 \tag{1.5}$$

In Figure 1.3, e is a cut edge of G but f is not.

Note *Equation* (1.5) *is replaced by* $\omega(G - e) > \omega(G)$ *in (Bondy and Murty 1976). But since we have already proved Proposition 1.4.3, we can be more specific.*

Consider a path graph P_n on n vertices. It is connected and it has altogether $n - 1$ edges. Apparently, if we remove any edge from it, then we will end up with a disconnected graph. This somehow tells us that for a graph to be connected, it cannot have too few edges, which leads to the question that what is the minimum number of edges of a connected graph of order n?

Proposition 1.4.4

The minimum number of edges of a connected graph on n vertices is $n - 1$.

Proof We shall prove by induction on the order. The induction hypothesis is that if G is a connected graph G with minimum number of edges and its order is less than or equal to n, then it has exactly $n - 1$ edges.

Base Case: If G is a trivial graph, then clearly it has no edges.

Inductive Step: Assume the hypothesis holds for $n = k$. Note that we only need to show G has k edges under the case where G is of order $k + 1$. Because G is a connected graph with minimum number of edges. By removing any edge, say e, from G will result in a disconnected graph $G - e$. And by Proposition 1.4.3, we know $G - e$ has two components, say G_1 and G_2. Note that both G_1 and G_2 are of orders less than or equal to k, say n_1 and n_2, respectively. Moreover, both G_1 and G_2 are connected graphs with minimum number of edges. Hence, applying the induction hypothesis to

both G_1 and G_2, we conclude that

$$|E(G_1)| = n_1 - 1 \quad \text{and} \quad |E(G_2)| = n_2 - 1$$

Therefore, the total number of edges of G is

$$|E(G)| = |E(G_1)| + |E(G_2)| + 1 = n_1 - 1 + n_2 - 1 + 1 = n_1 + n_2 - 1 = |V(G)| - 1 = k$$

The second last equality follows from the fact that the vertices of G is partitioned into vertices of G_1 and G_2, respectively, since G_1 and G_2 are complements. This completes the proof. ■

Apart from the path graph P_n, Figure 1.4 also depicts several other connected graphs of order 6 with minimum number of edges, in this case, 5 edges. Such graphs are called trees, as we will formally introduce in Chapter 2.

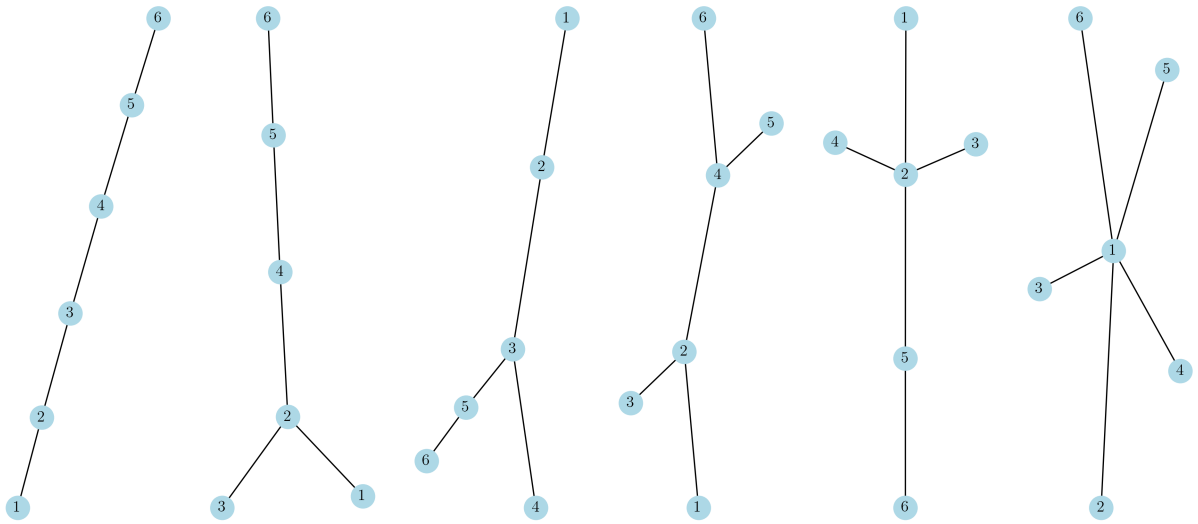

Figure 1.4: Connected graphs on 6 vertices with minimum number of edges, i.e., 5 edges.

Proposition 1.4.5

Let G be a simple and connected graph with order greater than or equal to 3. If G is not complete, then there exist three vertices u, v and w such that $uv, vw \in E$ but $uw \notin E$. ♠

Proof Because G is not complete, there exist two vertices u and w that are not incident. Let P be a shortest path from u to w.

If P is of length 2, then we can write $P = uvw$. Edges uv and vw exist in G. But the edge uw does not, which is as desired.

If the length of P is greater than or equal to 3, then P is of the form

$$P = uxv \cdots w$$

Note that $uv \notin E$. Otherwise, $uv \cdots w$ would be a shorter (u, w)-path than P. In this case the vertices u, x and v are as desired. ■

1.5 Cycles

One simple yet useful observation of a particular longest path in a graph is that all the neighbors of the terminus must occur along the path. To be specific, if $P = v_0e_1v_1 \cdots e_kv_k$ is one of the longest paths in G then P must contain all vertices in $N(v_k)$. To prove this, we assume P does not contain $v_{k+1} \in N(v_k)$. (Suppose $\psi(e_{k+1}) = v_kv_{k+1}$.) Then the path $P + e_{k+1}v_{k+1}$ is clearly longer than P, which leads to a contradiction. Figure 1.5 depicts such an example. Note that if 8 were a neighbor of 7, then path 12345678 would be longer.

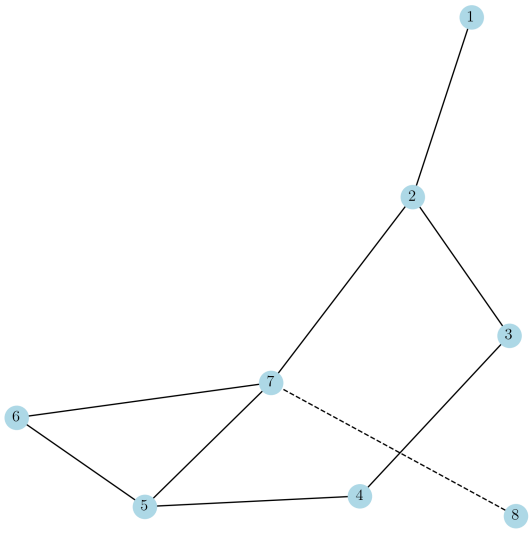

Figure 1.5: Path 1234567 is one of the longest paths.

Proposition 1.5.1

If G is a simple graph with $\delta(G) \geq 2$, then there exists a cycle of length at least $\delta(G) + 1$.

Proof Let $P = u \cdots v$ be a longest path in G. As noted before, all the neighbors of the terminus v, denoted by $v_1, \ldots, v_{\delta(G)}$ arranged by their original order in P, must occur along the path P. Note that the (v_1, v)-section, denoted by Q, is of length $\delta(G)$. Because $\deg(v) \geq 2$, v has at least two neighbors. In other words, the section Q is different from the edge connecting v and v_1. Hence, Qv_1 forms a cycle. And it is of length $\delta(G) + 1$. This completes the proof. ■

In fact, we have an algorithm to find a cycle without knowing the longest path in G.

Proof We need to show that Algorithm 2 works correctly.

Initialization: Firstly, note that line 7 is possible since v_0 has no loops and $\deg(v_0) \geq 2$.

We claim the loop invariants are

1. P has j vertices assuming that we are to execute the j-th iteration,
2. P has no duplicated vertices, i.e., P is a path, and
3. edge e is incident with v.

Maintenance: Suppose we are in the j-th iteration. After line 9, P remains a path. Because $\deg(v) \geq 2$, there exists an edge f other than e that is incident with v. Hence, line 10 works correctly.

Algorithm 2: Finding a Cycle in G With $\delta(G) \geq 2$

Input: G with $\delta(G) \geq 2$
Output: A cycle C

1 **if** G *has a loop* e *from* v *to* v **then**
2 $\quad C \leftarrow vev$;
3 $\quad$ **return** C ;
4 **end**
5 pick a vertex v_0 ;
6 $P \leftarrow v_0$;
7 pick $v \in N(v_0)$ and let e be the corresponding edge, i.e., $\psi(e) = v_0 v$;
8 **while** $v \notin P$ **do**
9 $\quad P \leftarrow Pev$;
10 $\quad$ pick $u \in N(v)$ such that there exists an edge f satisfying $\psi(f) = vu$ and $f \neq e$;
11 $\quad v \leftarrow u$;
12 $\quad e \leftarrow f$;
13 **end**
14 remove from P all vertices and edges before v ;
15 $C \leftarrow Pev$;

After executing line 12, we find that the number of vertices in P is increased by one, i.e., $j + 1$, P is still a path and e is incident with v.

Termination: We can complete at most $n - 1$ iterations since P can hold at most as many vertices as there are in G. Upon termination, we find v is in P and e is incident with v. By removing from P all vertices and edges before v and then append to it edge e and vertex v, we will obtain a cycle from v to itself. ■

Theorem 1.5.1

Graph G is a bipartite graph if and only if it contains no odd cycles. ♡

Proof ■

The **girth** of a graph is defined as the length of its shortest cycle. We say the girth of G is infinity if it contains no cycles. Apparently, if G has girth 1, then it contains a loop. If it has girth 2, then there must be a parallel edge. But it has no loops. And if the girth of G is greater than or equal to 3, then it must be a simple graph.

Proposition 1.5.2

A k-regular graph with girth 4 has at least $2k$ vertices. Moreover, if it has exactly $2k$ vertices, then it must be $K_{k,k}$, i.e., a complete bipartite graph with both sides of the bipartition having the same size k. Conversely, $K_{k,k}$ is a k-regular graph with girth 4. ♠

Figure 1.6 depicts several $K_{k,k}$ graphs.

Proof Take a pair of incident vertices u and v in G. We have $N(u) \cap N(v) = \emptyset$. Otherwise, a triangle would appear. Because the degrees of u and v are both k, equivalently, the sizes of their neighbors are k, G must has at least $2k$ vertices having these two disjoint sets $N(u)$ and $N(v)$.

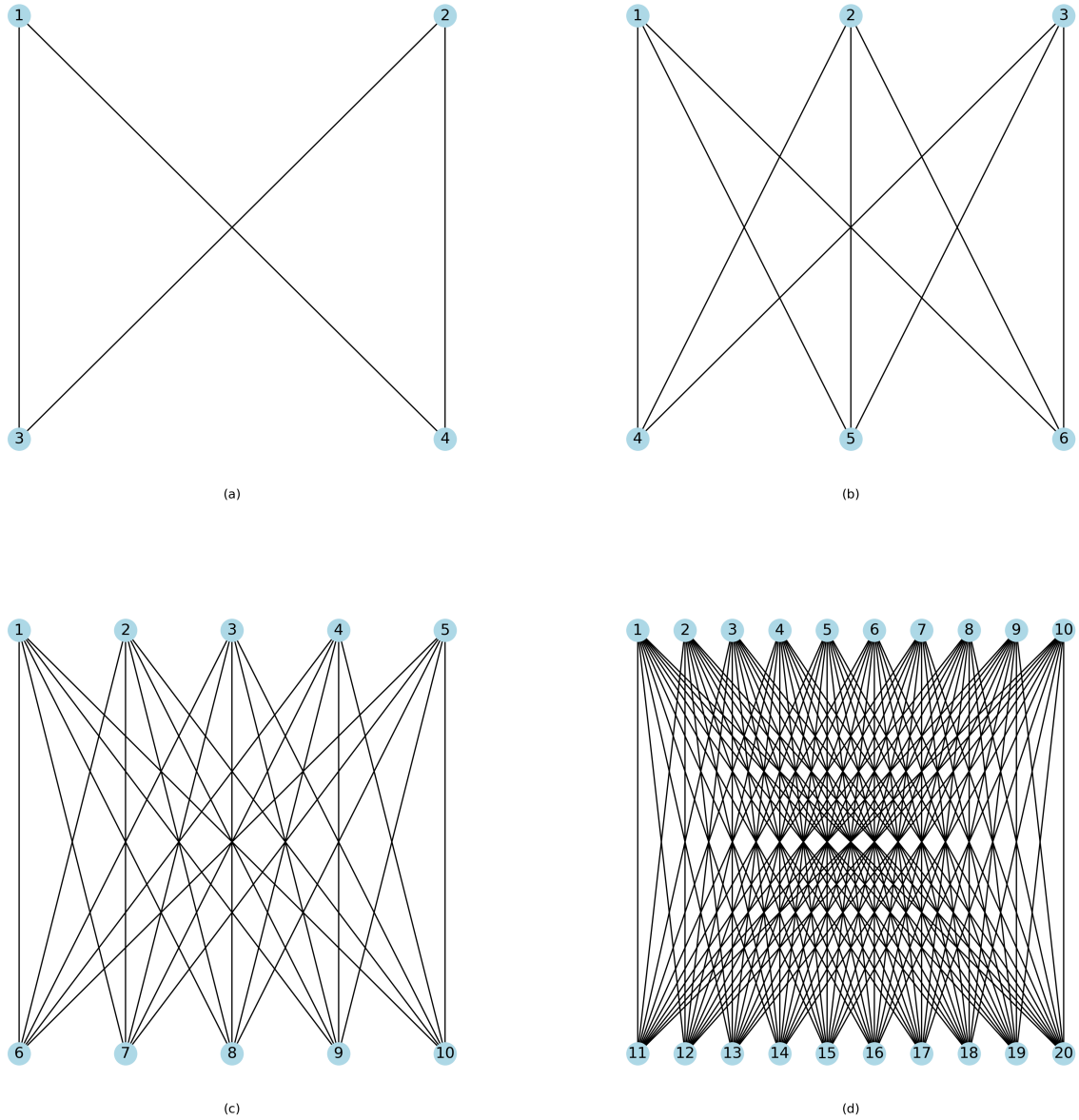

Figure 1.6: (a) $K_{2,2}$. (b) $K_{3,3}$. (c) $K_{5,5}$. (d) $K_{10,10}$.

Suppose $|V| = 2k$. Pick a pair of incident vertices u and v as before. From the previous proof, we know $|N(u)| = |N(v)| = k$ and they are disjoint. Since G now only has $2k$ vertices, V is composed of these two neighbor sets, i.e., $V = N(u) \uplus N(v)$. Observe that each pair of vertices in $N(u)$ cannot be joined with each other for there are no triangles. The same conclusion also holds for $N(v)$. Therefore, every vertex in $N(u)$ is joined with every vertex in $N(v)$ because the degree of every vertex is k. This shows that G is $K_{k,k}$.

We also need to show $K_{k,k}$ is a k-regular graph with girth 4. Clearly, it is k-regular. It contains no cycles with length 1 or 3 by Theorem 1.5.1. Of course, it does have any parallel edges, hence no 2-cycles. And we can easily find a 4-cycle in it. For example, if we suppose $X = \{x_1, \ldots, x_k\}$ and $Y = \{y_1, \ldots, y_k\}$ form a bipartition of $K_{k,k}$, then $x_1y_1x_2y_2x_1$ is a 4-cycle. ■

We call a graph **acyclic** if it contains no cycles.

The next theorem is useful to determine whether an edge is a cut edge by checking that if it is

contained in any cycles.

Theorem 1.5.2

An edge e is an cut edge of G if and only if it is contained in no cycles of G. ♡

Proof In the following proof, we suppose that the two ends of e are x and y and that they are in the path equivalent class V_1. Then, by Lemma 1.4.1, we know every vertex v in V_1 is either connected to x or y in subgraph $G - e$.

($\Longrightarrow$) We first exclude the case where e is a loop. For the rest scenarios, we shall prove the contrapositive. Suppose e is contained in some cycle C. Then $C - e$ is a path joining x and y. This implies that x and y remain connected in $G - e$. Therefore, for each $v \in V_1$, v is connected to both x and y in $G - e$. In other words, $(G - e)[V_1]$ is connected. Since no additional components will appear in $G - e$, e is not an cut edge.

($\Longleftarrow$) If e is contained in no cycles of G, then x and y are disconnected in $G - e$. Hence, $(G - e)[V_1]$ is then disconnected. Therefore, $\omega(G - e) > \omega(G)$, which further implies that e is a cut edge. ■

As we can imagine, an acyclic graph cannot have too many edges. Interestingly, an acyclic graph with maximum number of edges is exactly a connected graph with minimum number of edges. Compare the following proposition with Proposition 1.4.4.

Proposition 1.5.3

The maximum number of edges of an acyclic graph G on n vertices is $n - 1$. Furthermore, in this case, G is also connected. ♠

Proof Suppose G is an acyclic graph with maximum number of edges. We are going to show that G is in fact a connected graph with minimum number of edges.

We first show that G is connected. Assume G is not connected. Then there exist two distinct vertices u and v such that there are no paths between them. Of course, $uv \notin E$. By adding the edge uv, $G + uv$ remains acyclic. But then $G + uv$ has more edge than that of G, which contradicts the assumption that G is an acyclic graph with maximum number of edges. Therefore, we see that G must be connected.

On the other hand, choose an edge e and we will find that e is contained in no cycles since G is acyclic. By Theorem 1.5.2, e is a cut edge. Hence, $G - e$ is disconnected. This implies that G is a connected graph with minimum number of edges. Therefore, G has exactly $n - 1$ edges by Proposition 1.4.4. ■

1.6 Shortest Paths and Dijkstra's Algorithm

With every edge e in graph G, we can associate a real number $w(e)$, usually positive, which we shall call the **weight** of that edge. As we will see, to unify our notations for some special cases, it is convenient to define the weight of a nonexistent edge between two non-incident vertices as infinity ∞.

We can now extend our former definition of the length of a path by

$$|P| := \sum_{e \in P} w(e)$$

Note that if all the edges are assigned to unit weights, then $|P|$ is just the number of edges in it, which reduces to the former definition. Notice also that the length of a trivial path is also zero since the sum of nothing in the above equation, by convention, is zero.

Given a map, the cities can be regarded as vertices, and for each pair of adjacent cities, we can draw an edge in between them. In this example, it is reasonable to define the weight of edge uv as the distance between cities u and v. Then the problem of finding a shortest path from u_0 to v arises quite naturally if we want to travel from city u_0 (where we live) to every other city v at the minimum cost.

Formally, for all $u, v \in V$, the length of the shortest path from u to v is defined as

$$d(u,v) := \begin{cases} \min\{|P| \mid P \text{ is a } (u,v)\text{-path}\}, & \text{if } u \text{ and } v \text{ are connected} \\ \infty, & \text{if } u \text{ and } v \text{ are disconnected} \end{cases}$$

Given a proper subset S of V and a source $u_0 \in S$ in it, it is natural to define the distance from u_0 to the complement $S^{\complement}$ as the minimum distance from u_0 to every $v \in S^{\complement}$, that is,

$$d(u_0, S^{\complement}) := \min_{v \in S^{\complement}} d(u_0, v)$$

Of course, if $d(u_0, S^{\complement}) = \infty$, then there are no paths from u_0 to any vertex in $S^{\complement}$, in other words, u_0 is disconnected from $S^{\complement}$. Suppose $d(u_0, S^{\complement})$ is a finite number, which means there exists a path $P = u_0 \cdots uv$ from u_0 to some vertex $v \in S^{\complement}$. The following are some simple observations about this path P:

1. P is a shortest (u_0, v)-path,
2. $u \in S$ and the (u_0, u)-section, $u_0 \cdots u$, is a shortest (u_0, u)-path, and
3. we have the equation

$$d(u_0, S^{\complement}) = d(u_0, u) + w(uv) \tag{1.6}$$

Equation (1.6) motivates us to state the following proposition, which is the central idea of Dijkstra's algorithm.

Proposition 1.6.1

Let $G = (V, E)$ be a simple graph. Let $S \subseteq V$ be a subset of vertices and $u_0 \in S$ a point in it. We have

$$d(u_0, S^{\complement}) = \min_{u \in S, v \in S^{\complement}} \{d(u_0, u) + w(uv)\} \tag{1.7}$$

♠

Proof First, note that the set

$$\left\{ d(u_0, u) + w(uv) \;\middle|\; u \in S, v \in S^{\complement} \right\}$$

is a finite set (possibly containing infinite numbers). Hence, it does have a minimum.

Assume there exist $u^* \in S$ and $v^* \in S^{\complement}$ such that

$$d(u_0, S^{\complement}) > d(u_0, u^*) + w(u^* v^*) \tag{1.8}$$

Note *Note that* (1.8) *implicitly implies that* u_0 *and* u^* *are connected and* u^* *and* v^* *are connected since the right-hand side of* (1.8) *is a finite number.*

Then there exists a (u_0, u^*)-path P such that $|P| = d(u_0, u^*)$. Note that Pv^* forms a path since all vertices in P are in S while v^* is in the complement S^*. But Pv^* is a path from u_0 to $S^\complement$ with length $d(u_0, u^*) + w(u^*v^*)$, which is less than $d(u_0, S^\complement)$ by assumption. Therefore, this leads to a contradiction. We have

$$d(u_0, S^\complement) \leq d(u_0, u) + w(uv) \quad \forall u \in S \, \forall v \in S^\complement$$

And (1.7) is proved. ■

1.6.1 A First Attempt

Starting from an initial set $S^{(0)} = \{u_0\}$ containing just the source, we want to enlarge this set in a way that a shortest path from u_0 to any vertex within the set is known. Given $S^{(j-1)}$, in each iteration j, we find optimal vertices $u^* \in S^{(j-1)}$ and $v^* \in V \setminus S^{(j-1)}$ in the sense that equation (1.7) is satisfied. In doing so, we are guaranteed to find a shortest path from u_0 to $V \setminus S^{(j-1)}$ with the terminus v^*. Then we extend the set $S^{(j-1)}$ to $S^{(j)}$ by adding v^*.

Based on this idea, we propose our first attempt to find shortest paths in the following algorithm (Algorithm 3).

1.6.2 Dijkstra's Algorithm

Given a vertex v, its predecessor is given by $\Pi[v]$. And the predecessor of $\Pi[v]$ is $\Pi[\Pi[v]]$, and so forth. We can keep accessing the predecessors until hopefully stops at the source u_0. In this case, we have recovered a path from u_0 to v, $u_0 \cdots \Pi[v]v$. To make our proof concise, we call this procedure *recovering a* (u_0, v)*-path using* Π.

Proof Note that there is an early return in line 22. Hence, we may not complete all $n - 1$ iterations of the for loop (lines 7-23). Suppose we can complete k iterations.

Loop Invariants: Upon completion of the j-th iteration, we claim that

1. $D[u] = d(u_0, u) \quad \forall u \in S^{(j)}$,
2. $D[v] = \min_{u \in S^{(j-1)}} \{d(u_0, u) + w(uv)\} \quad \forall v \in V \setminus S^{(j)}$,
3. we can recover a (u_0, u)-path using Π for every $u \in S^{(j)}$, and
4. $S^{(j)} = S^{(j-1)} \uplus u^{(j)}$.

Maintenance: Assuming all loop invariants hold for $j - 1$. We now consider the j-th iteration. In the inner loop (lines 10-20), by referring to loop invariant 2, we note that the procedure from line 11 to line 15 ensures that

$$D[v] = \min \left\{ \min_{u \in S^{(j-2)}} \{d(u_0, u) + w(uv)\}, D[u^{(j-1)}] + w(u^{(j-1)}v) \right\} \quad \forall v \in V \setminus S^{(j-1)} \tag{1.9}$$

after this inner loop is completed. But invariant 1 implies that $D[u^{(j-1)}] = d(u_0, u^{(j-1)})$. Moreover,

Algorithm 3: A First Attempt to Find the Shortest Paths

Input: A simple weighted graph $G = (V, E)$ and a source $u_0 \in V$
Output: Dictionaries D and Π maintaining the lengths of shortest paths from u_0 and predecessors of all vertices connected to u_0, respectively

```
1  S^(0) ← {u_0} ;
2  D[u_0] ← 0 ;
3  for j = 1, ..., |V| − 1 do
4      d* ← ∞ ;
5      u* ← v* ← none ;
6      for u ∈ S^(j−1) do
7          for v ∈ V \ S^(j−1) do
8              d ← D[u] + w(uv) ;
9              if d < d* then
10                 u* ← u ;
11                 v* ← v ;
12                 d* ← d ;
13             end
14         end
15     end
16     if d* = ∞ then
17         return D and Π ;
18     end
19     D[v*] ← d* ;
20     Π[v*] ← u* ;
21     S^(j) ← S^(j−1) ∪ {v*} ;
22 end
```

we have $S^{(j-1)} = S^{(j-2)} \uplus u^{(j-1)}$ by invariant 4. Hence, (1.9) reduces to

$$D[v] = \min\left\{\min_{u \in S^{(j-2)}} \{d(u_0, u) + w(uv)\}, d(u_0, u^{(j-1)}) + w(u^{(j-1)}v)\right\}$$
$$= \min_{u \in S^{(j-1)}} \{d(u_0, u) + w(uv)\} \quad \forall v \in V \setminus S^{(j-1)} \tag{1.10}$$

Note that invariant 2 for iteration j follows directly from (1.10) because $V \setminus S^{(j)} \subseteq V \setminus S^{(j-1)}$ by line 25 if line 25 is reachable in the current iteration. If line 25 is not reachable, which means the algorithm terminates at line 22, then there is nothing to prove for this iteration.

The purpose of lines 16-19 is that upon completion of lines 10-20, we have

$$d^* = \min_{v \in V \setminus S^{(j-1)}} \min_{u \in S^{(j-1)}} \{d(u_0, u) + w(uv)\}$$

By Proposition 1.6.1, we know d^* is the length of a shortest path from u_0 to $V \setminus S^{(j-1)}$, that is,

$$d^* = d(u_0, V \setminus S^{(j-1)})$$

We are now at the end of line 20. There are two cases.

(Case 1: Early Return) If $d^* = \infty$ or equivalently $d(u_0, V \setminus S^{(j-1)}) = \infty$, then it means that u_0 is disconnected from $V \setminus S^{(j-1)}$. Since no shortest paths are to be discovered, the algorithm needs to

Algorithm 4: Dijkstra's Algorithm

Input: A simple weighted graph $G = (V, E)$ and a source $u_0 \in V$
Output: Dictionaries D and Π maintaining the lengths of shortest paths from u_0 and predecessors of all vertices connected to u_0, respectively

```
// initialization
```
1 $S^{(0)} \leftarrow \{u_0\}$;
2 **for** $v \in V$ **do**
3 $\quad D[v] \leftarrow \infty$;
4 **end**
5 $D[u_0] \leftarrow 0$;
6 $u^{(0)} \leftarrow u_0$;
7 **for** $j = 1, \ldots, |V| - 1$ **do**
8 $\quad d^* \leftarrow \infty$;
9 $\quad v^* \leftarrow$ none ;
10 $\quad$ **for** $v \in V \setminus S^{(j-1)}$ **do**
11 $\quad\quad d \leftarrow D[u^{(j-1)}] + w(u^{(j-1)}v)$;
12 $\quad\quad$ **if** $D[v] > d$ **then**
13 $\quad\quad\quad D[v] \leftarrow d$; `// update distance`
14 $\quad\quad\quad \Pi[v] \leftarrow u^{(j-1)}$; `// update predecessor`
15 $\quad\quad$ **end**
16 $\quad\quad$ **if** $D[v] < d^*$ **then**
17 $\quad\quad\quad v^* \leftarrow v$;
18 $\quad\quad\quad d^* \leftarrow D[v^*]$;
19 $\quad\quad$ **end**
20 $\quad$ **end**
21 $\quad$ **if** $d^* = \infty$ **then**
22 $\quad\quad$ **return** D and Π ;
23 $\quad$ **end**
24 $\quad u^{(j)} \leftarrow v^*$;
25 $\quad S^{(j)} \leftarrow S^{(j-1)} \cup \{u^{(j)}\}$;
26 **end**

terminate. Recall we assume we can only complete k iterations. Therefore, the current iteration must be $k + 1$ for we are exiting the algorithm. No loop invariants need to be proved since this iteration is not completed.

(Case 2) On the other hand, we now suppose v^* is some vertex at the end of line 20. Let $u^{(j)} = v^*$ (line 24). Then invariant 4 for iteration j is proved immediately by line 25.

We now prove invariant 3 for j. Note that the predecessor of $u^{(j)}$ is $u^{(j-1)}$, i.e., $\Pi[u^{(j)}] = u^{(j-1)}$ by line 14. We then recover a $(u_0, u^{(j-1)})$-path, say P, using Π (invariant 3 for $j - 1$). Note that $Pu^{(j)}$ is a $(u_0, u^{(j)})$-path, and it can be recovered using Π since $\Pi[u^{(j)}] = u^{(j-1)}$ and P itself is recovered using Π. This proves invariant 3 of iteration j.

Furthermore, by recalling $u^{(j)} = v^*$ and $d^* = d(u_0, V \setminus S^{(j-1)})$, we note that $Pu^{(j)}$ is a shortest path from u_0 to $u^{(j)}$ since its length is $d(u_0, V \setminus S^{(j-1)})$. In other words, it is also a shortest path from

u_0 to $V \setminus S^{(j-1)}$. Therefore, we can write

$$d(u_0, u^{(j)}) = d^* = D[v^*]$$

where the last equality follows from line 18. Replacing v^* with $u^{(j)}$, equivalently we have

$$D[u^{(j)}] = d(u_0, u^{(j)}) \tag{1.11}$$

This equation (1.11) along with invariant 1 for $j-1$ implies that invariant 1 also holds for j. Note that we have shown that all loop invariants are preserved when the current iteration (iteration j) is completed.

Termination: As the algorithm terminates, there are $k+1$ vertices in $S^{(k)}$. And the vertices outside $S^{(k)}$ are not reachable from the source u_0. For each $u \in S^{(k)}$, we have $D[u] = d(u_0, u)$ (invariant 1), that is, $D[u]$ stores the length of a shortest path from u_0 to u, as desired. And by invariant 3, we can recover a shortest (u_0, u)-path using Π. This completes the proof. ■

Chapter 2 Trees

2.1 Definition of Trees

A **tree** is a connected acyclic graph. A tree is clearly a simple graph for there cannot be any loops (1-cycles) or parallel edges (2-cycles).

If a tree T is nontrivial, then the degree of every vertex must greater than or equal to 1 since a tree is connected. Moreover, there always exists at least one vertex with degree 1, that is,

$$\delta(T) = 1$$

If this is not the case, then we end up with a graph with $\delta \geq 2$. But this further implies that there exists a cycle in the graph by Proposition 1.5.1, contradicting the definition of a tree.

If the reader has a background of computer science, then the tree data structure one might be familiar with is actually what we call a **rooted tree** with one vertex called the **root** with special treatment. However, the tree we defined here is an unrooted tree, or **free tree**, since we often do not specify the root vertex.

A tree **leaf** is a vertex of degree 1. Note that it is possible that the degree of the tree root is 1 if it is specified. But we often do not call it a leaf. Hence, in a rooted tree, a leaf is a non-root vertex of degree 1. Clearly, a nontrivial tree always has a leaf. In fact, it always has at least two leafs, as we shall see in Corollary 2.1.1.

Theorem 2.1.1

*In a tree T, any two distinct vertices are connected by a **unique** path.* ♡

Proof Assume there are two distinct (u, v)-paths, P_1 and P_2 (regraded as path graphs). Then there exists at least one distinct edge in these two, say $e = xy$. Without loss of generality, we may assume $e \in V(P_1)$.

Consider the graph $P_1 \cup P_2 - e$. We claim that it contains a (x, y)-path. To see this, we note that there exist a (x, u)-path and a (v, y)-path in P_1, and also a (u, v)-path in P_2. By concatenating all these three paths, we will obtain a (x, y)-walk (not necessarily a path). But by Proposition 1.4.5, we can always extract a (x, y)-path from it, say P_3.

Now, we have two distinct paths from x to y in T. One is xy and the other is P_3, which contradicts the hypothesis. ■

The graph in Figure 2.1 is certainly not a tree. But any two distinct vertices are indeed connected by a unique path. Therefore, for the converse of Theorem 2.1.1 to hold, we need to exclude the cases where graphs contain loops.

Theorem 2.1.2

If graph T is loopless and there exists one and only one path connecting each pair of distinct vertices, then T is a tree. ♡

Proof First, note that T is connected. Assume, on the contrary, there exists a cycle C in G. Let e be

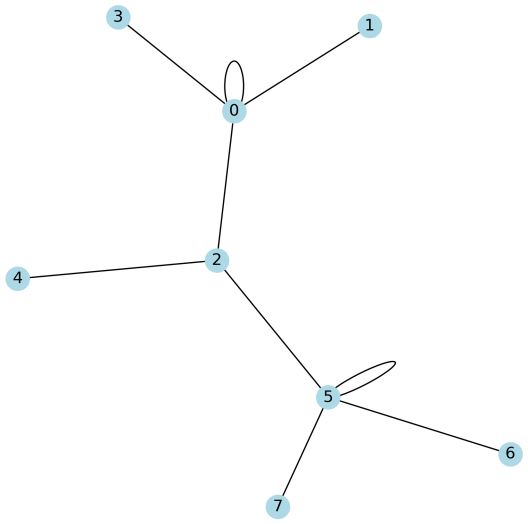

Figure 2.1: It will become a tree if the loops are removed.

an edge in C. Since T is assumed loopless, the two ends of e must be distinct, say x and y.

Note that there exists a (x, y)-path in $C - e$, which is different from the path xy. (In fact, this path is $C - e$ itself.) This leads to a contradiction. ■

Theorem 2.1.3

Let T be a tree. Then we have

$$|E| = |V| - 1 \tag{2.1}$$

♡

Proof We shall prove by induction on the order $|V|$.

Base Case: If $|V| = 1$, then T is a trivial graph without any edges, and hence (2.1) holds.

Inductive Step: Assume this theorem holds for any trees with order k. Suppose now $|V(T)| = k+1$. Pick a leaf v, and then remove it from T. This is always possible as we have noted. Observe that $T - v$ remains a tree. And by removing v, we only remove a single edge from T since $\deg(v) = 1$. Therefore, $|E(T - v)| = |E(T)| - 1$. But by the induction hypothesis, we know $|E(T - v)| = |V(T - v)| - 1 = k - 1$. It then follows that

$$|E(T)| = |E(T - v)| + 1 = k - 1 + 1 = (k + 1) - 1 = |V(T)| - 1$$

This completes the proof. ■

Corollary 2.1.1

Every nontrivial tree has at least two leafs.

♡

Proof As we have noted, a nontrivial tree T has at least one leaf. Assume T only has one leaf. Then we have

$$\sum_{v \in V} \deg(v) \geq 1 + 2(|V| - 1) = 2\,|V| - 1$$

since all vertices, except one, are of degree at least 2. Combined with Theorem 1.2.1, we know

$$2|E| = \sum_{v \in V} \deg(v) = 2|V| - 1 \tag{2.2}$$

On the other hand, it follows from Theorem 2.1.3 that

$$2|E| = 2|V| - 2 \tag{2.3}$$

Note that equations (2.2) and (2.3) contradict each other. Therefore, T has at least two leafs. ■

Theorem 2.1.4

Let T be a graph with $|T| - 1$ edges, then the following statements are equivalent:

1. *T is a tree.*
2. *T is connected.*
3. *T is acyclic.*

♡

Proof Note that we only need to show 2 $\implies$ 3 and 3 $\implies$ 2.

(2 $\implies$ 3) By Proposition 1.4.4, we know T is a connected graph with minimum number of edges. We need to show T is acyclic. Assume that, on the contrary, there exists a cycle C in T. Pick an edge e in C. Note that e cannot be a cut edge by Theorem 1.5.2. Therefore, $G - e$ remains connected. But clearly $G - e$ has one less edge than that of G, which leads to a contradiction. Therefore, T is indeed acyclic.

(3 $\implies$ 2) This direction follows directly from Proposition 1.5.3. ■

References

[1] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. New York: North Holland, 1976. ISBN: 978-0-444-19451-0.

[2] S. L. Hakimi. "On Realizability of a Set of Integers as Degrees of the Vertices of a Linear Graph. I". In: *Journal of the Society for Industrial and Applied Mathematics* 10.3 (Sept. 1962), pp. 496–506. ISSN: 0368-4245, 2168-3484. DOI: 10.1137/0110037.

Index

A

acyclic graphs 11

C

complete bipartite graph 2
connected and disconnected vertices 5
cut edge 7

D

degree sequence 3

F

free tree 18

G

girth 10
graph realization problem 3
graphical degree sequence 3

I

isomorphism 1

L

leaf 18

O

origin of a walk 5

P

path in a graph 5

R

root of a tree 18
rooted tree 18

T

terminus of a walk 5
trail in a graph 5
tree 18
trivial walk 4

W

walk in a graph 4
weight of an edge 12