



# 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

<b>Chapter 1 Basic Concepts of Graphs</b>	<b>1</b>
1.1 Isomorphism . . . . .	1
1.2 Vertex Degrees . . . . .	2
1.3 Paths and Connection . . . . .	2
1.4 Cycles . . . . .	5
1.5 Shortest Paths and Dijkstra's Algorithm . . . . .	8
<b>Chapter 2 Trees</b>	<b>13</b>
2.1 Definition of Trees . . . . .	13
<b>References</b>	<b>15</b>
<b>Index</b>	<b>16</b>

# 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  $\theta : V(G) \rightarrow V(H)$  and  $\phi : E(G) \rightarrow E(H)$  such that

$$\psi_G(e) = uv \implies \psi_H(\phi(e)) = \theta(u)\theta(v) \quad (1.1)$$

The ordered pair  $(\theta, \phi)$  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  $\theta$  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  $\theta$  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  $\theta : V(G) \rightarrow V(H)$  such that

$$uv \in E(G) \implies \theta(u)\theta(v) \in E(H) \quad (1.2) \quad \img alt="red heart icon" data-bbox="882 684 901 699"/>$$

**Proof** (Necessity) Suppose that there exist  $\theta$  and  $\phi$  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  $\phi : E(G) \rightarrow E(H)$  by

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

We need to show  $\phi$  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  $\phi$  is injective. On the other hand, for any edge  $f \in H$ . Write  $f = ij$  (i.e.,  $\psi_H(f) = ij$ ). Then because  $\theta$  is bijective, there exist  $u, v \in V(G)$  such that  $\theta(u) = i$  and  $\theta(v) = j$ . Hence,  $\phi(uv) = ij$ , which implies  $\phi$  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.

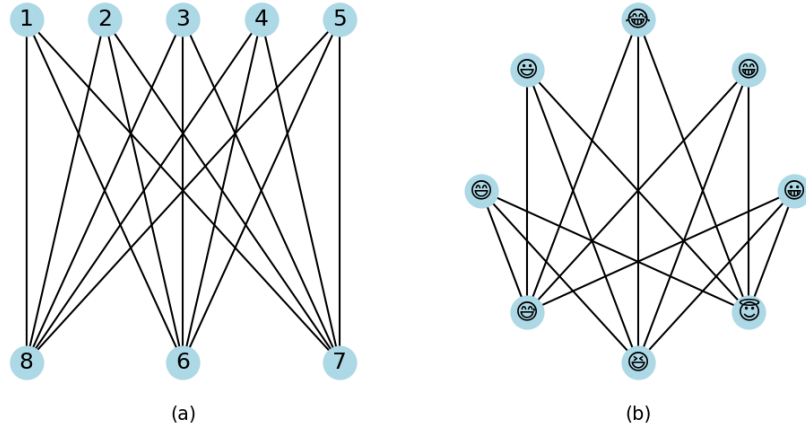


Figure 1.1: Both (a) and (b) are  $K_{5,3}$ .

### 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  $\theta : V(G) \rightarrow V(H)$  in such a way that  $\theta$  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


## 1.3 Paths and Connection

A **walk** in a graph is a sequence of edges  $e_1 \cdots e_k$  joining a *nonempty* sequence of vertices  $v_0 v_1 \cdots v_k$ , which is denoted by

$$v_0 e_1 v_1 \cdots e_k v_k \quad (1.3)$$

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 1.3.1

*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  $P$

```

1 initialize  $P$  as a sequence containing just one vertex  $v_0$  ;
2 for  $i = 1, \dots, k$  do
3   if  $v_i$  is not in  $P$  then
4     append  $e_i$  and  $v_i$  to  $P$  ;
5   else
6     remove all the vertices and edges after the vertex  $v_i$  from  $P$  ;
7   end
8 end
```

---

#### Proposition 1.3.2

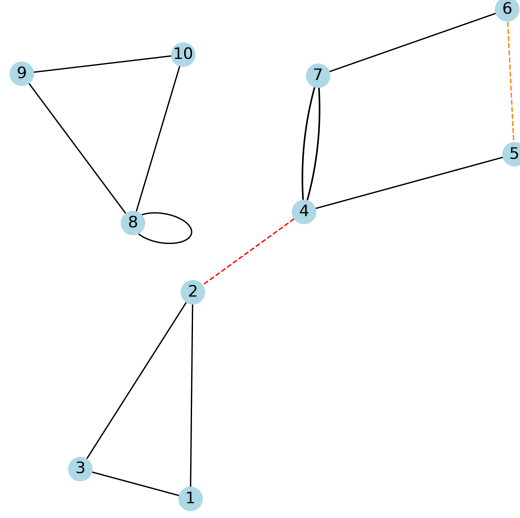
*The number  $(v_i, v_j)$ -walks of length  $k$  in  $G$  is the  $(i, j)$ -th entry of the  $k$ -th power of the adjacency matrix  $A$ , i.e.,  $A^k$ .*



**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 1.2.



**Figure 1.2:** The graph  $G$  has 2 components. If we remove edge 56, then  $G - 56$  still has 2 components. But if we remove edge 24, then the remaining graph  $G - 24$  has 3 components.

### Proposition 1.3.3

If  $e \in E$ , then we have

$$\omega(G) \leq \omega(G - e) \leq \omega(G) + 1 \quad (1.4) \quad \spadesuit$$

Inequalities (1.4) describe this idea 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  $e = uv$  is not a loop, i.e.,  $u$  and  $v$  are distinct, in the rest of the proof. Suppose that the components of  $G$  are  $G[V_1], \dots, G[V_\omega]$ . Without loss of generality, we may also assume that  $u, v \in V_1$ .

(Case 1) Suppose that there exists a  $(u, v)$ -path in  $G - e$ , then  $u$  is still connected to  $v$  in  $G - e$ , i.e.,  $u \sim v$  in  $G - e$ . Pick an arbitrary vertex  $x \in V_1$ . Suppose that  $x$  is originally connected to  $u$  by a path  $P$  containing  $e$ , then  $P$  is of the form

$$P = x \cdots v e u$$

Hence,  $x$  must be connected to  $v$  in  $G - e$ , i.e.,  $x \sim v$ , since the  $(x, v)$ -section in  $P$  does not involve  $e$ . But  $x \sim v$ . By the transitivity, we have  $x \sim u$ . Therefore,

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

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  $u$  is disconnected from  $v$  in  $G - e$ . For any vertex  $x \in V_1$ , if  $x$  is disconnected from  $u$  in  $G - e$ , then  $x$  must be originally connected to  $u$  in  $G$  by a path containing edge  $e$ . Applying a similar argument as before, we conclude that  $x \sim v$  in  $G - e$ .

Therefore, for every  $x \in V_1$ , in graph  $G - e$ , we have either

1.  $x \sim u$ , or
2.  $x \sim v$

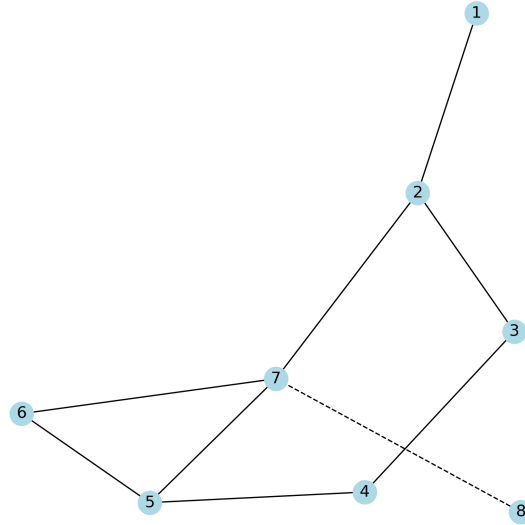
But  $x$  cannot be connected to both  $u$  and  $v$  since  $u$  and  $v$  are assumed disconnected from each other. It then follows that  $V_1$  can be partitioned into two equivalent classes,  $[u]$  and  $[v]$ , that is,

$$V_1 = [u] \uplus [v]$$

Therefore,  $G - e$  has one more component than that of  $G$ , and hence  $\omega(G - e) = \omega(G) + 1$ . ■

## 1.4 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  $P = v_0 e_1 v_1 \cdots e_k v_k$  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  $\psi(e_{k+1}) = v_k v_{k+1}$ .) Then the path  $P + e_{k+1} v_{k+1}$  is clearly longer than  $P$ , which leads to a contradiction. Figure 1.3 depicts such an example. Note that if 8 were a neighbor of 7, then path 12345678 would be longer.



**Figure 1.3:** Path 1234567 is one of the longest paths.

### Proposition 1.4.1

*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  $v_1, \dots, v_{\delta(G)}$  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  $G$ .

---

**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    $C \leftarrow vev$  ;
3   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.,  $\psi(e) = v_0v$  ;
8 while  $v \notin P$  do
9    $P \leftarrow Pev$  ;
10  pick  $u \in N(v)$  such that there exists an edge  $f$  satisfying  $\psi(f) = vu$  and  $f \neq e$  ;
11   $v \leftarrow u$  ;
12   $e \leftarrow f$  ;
13 end
14 remove from  $P$  all vertices and edges before  $v$  ;
15  $C \leftarrow Pev$  ;
```

---

**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. 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 1.4.1**

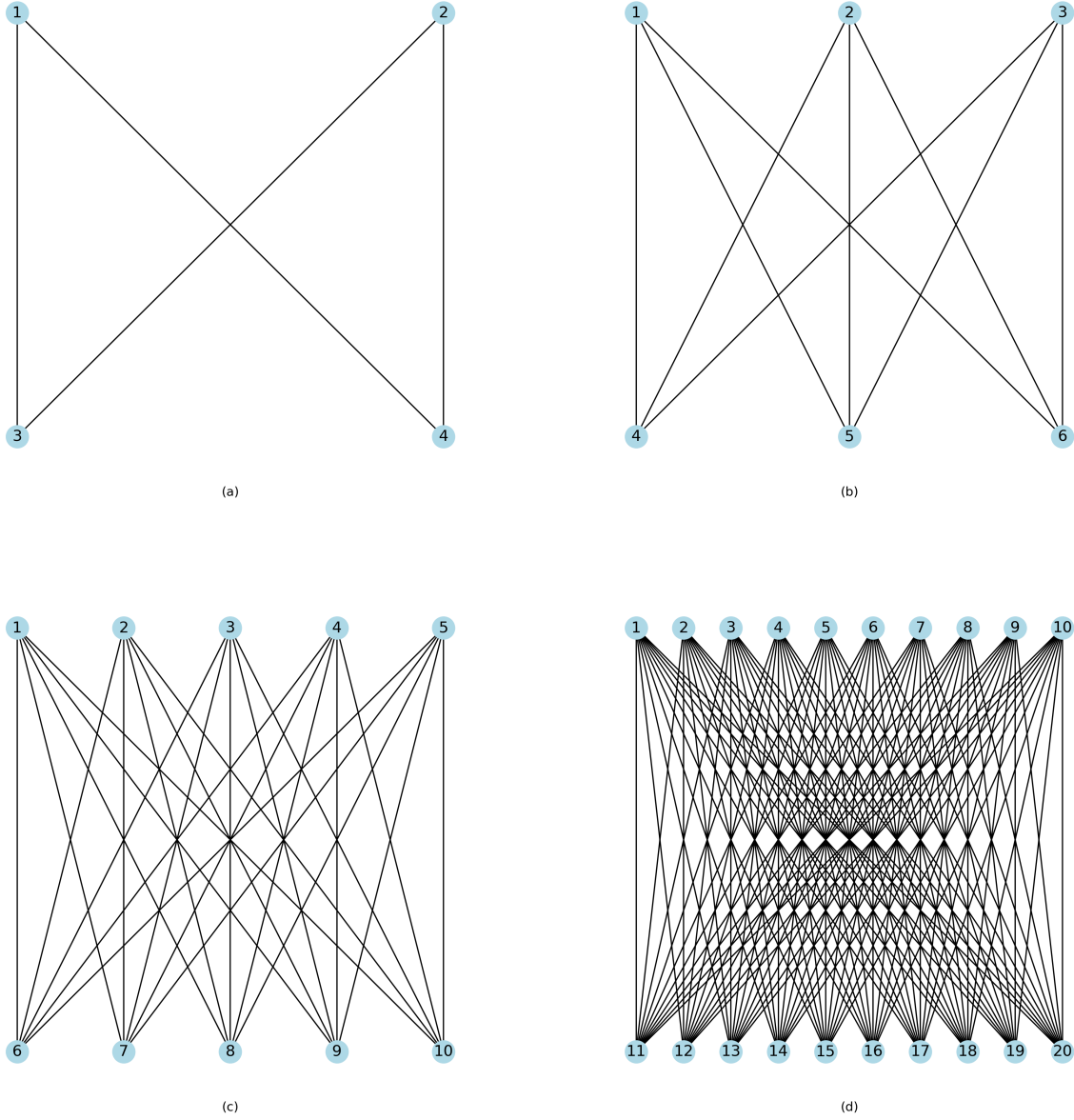

The **girth** of a graph is defined as the length of its shortest cycle. We say the girth of  $G$  is infinity if it contains no cycles. Apparently, if  $G$  has girth 1, then it contains a loop. If it has girth 2, then there must be a parallel edge. But it has no loops. And if the girth of  $G$  is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

**Proposition 1.4.2**

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 1.4 depicts several  $K_{k,k}$  graphs.



**Figure 1.4:** (a)  $K_{2,2}$ . (b)  $K_{3,3}$ . (c)  $K_{5,5}$ . (d)  $K_{10,10}$ .

**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 have at least  $2k$  vertices having these two disjoint sets  $N(u)$  and  $N(v)$ .

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 1.4.1. 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  $X = \{x_1, \dots, x_k\}$  and  $Y = \{y_1, \dots, y_k\}$  form a bipartition of  $K_{k,k}$ , then  $x_1y_1x_2y_2x_1$  is a 4-cycle. ■

## 1.5 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  $\infty$ .

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  $S^c$  as the minimum distance from  $u_0$  to every  $v \in S^c$ , that is,

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

Of course, if  $d(u_0, S^c) = \infty$ , then there are no paths from  $u_0$  to any vertex in  $S^c$ , in other words,  $u_0$  is disconnected from  $S^c$ . Suppose  $d(u_0, S^c)$  is a finite number, which means there exists a path  $P = u_0 \cdots uv$  from  $u_0$  to some vertex  $v \in S^c$ . 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

$$d(u_0, S^c) = d(u_0, u) + w(uv) \tag{1.5}$$

Equation (1.5) motivates us to state the following proposition, which is the central idea of Dijkstra's

algorithm.

### Proposition 1.5.1

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

$$d(u_0, S^c) = \min_{u \in S, v \in S^c} \{d(u_0, u) + w(uv)\} \quad (1.6)$$

**Proof** First, note that the set

$$\{d(u_0, u) + w(uv) \mid u \in S, v \in S^c\}$$

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

Assume there exist  $u^* \in S$  and  $v^* \in S^c$  such that

$$d(u_0, S^c) > d(u_0, u^*) + w(u^*v^*) \quad (1.7)$$



**Note** Note that (1.7) implicitly implies that  $u_0$  and  $u^*$  are connected and  $u^*$  and  $v^*$  are connected since the right-hand side of (1.7) 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  $S^c$ . But  $Pv^*$  is a path from  $u_0$  to  $S^c$  with length  $d(u_0, u^*) + w(u^*v^*)$ , which is less than  $d(u_0, S^c)$  by assumption. Therefore, this leads to a contradiction. We have

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

And (1.6) is proved. ■

## 1.5.1 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 (1.6) 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).

## 1.5.2 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  $\Pi$* .

**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

**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  $\Pi$  maintaining the lengths of shortest paths from  $u_0$  and predecessors of all vertices connected to  $u_0$ , respectively

```

1  $S^{(0)} \leftarrow \{u_0\}$ ;
2  $D[u_0] \leftarrow 0$ ;
3 for  $j = 1, \dots, |V| - 1$  do
4    $d^* \leftarrow \infty$ ;
5    $u^* \leftarrow v^* \leftarrow \text{none}$ ;
6   for  $u \in S^{(j-1)}$  do
7     for  $v \in V \setminus S^{(j-1)}$  do
8        $d \leftarrow D[u] + w(uv)$ ;
9       if  $d < d^*$  then
10         $u^* \leftarrow u$ ;
11         $v^* \leftarrow v$ ;
12         $d^* \leftarrow d$ ;
13     end
14   end
15 end
16 if  $d^* = \infty$  then
17   return  $D$  and  $\Pi$ ;
18 end
19  $D[v^*] \leftarrow d^*$ ;
20  $\Pi[v^*] \leftarrow u^*$ ;
21  $S^{(j)} \leftarrow S^{(j-1)} \cup \{v^*\}$ ;
22 end

```

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  $\Pi$  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

$$D[v] = \min \left\{ \min_{u \in S^{(j-2)}} \{d(u_0, u) + w(uv)\}, D[u^{(j-1)}] + w(u^{(j-1)}v) \right\} \quad \forall v \in V \setminus S^{(j-1)} \quad (1.8)$$

after this inner loop is completed. But invariant 1 implies that  $D[u^{(j-1)}] = d(u_0, u^{(j-1)})$ . Moreover, we have  $S^{(j-1)} = S^{(j-2)} \uplus u^{(j-1)}$  by invariant 4. Hence, (1.8) reduces to

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

Note that invariant 2 for iteration  $j$  follows directly from (1.9) 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



**Algorithm 4:** Dijkstra's Algorithm**Input:** A simple weighted graph  $G = (V, E)$  and a source  $u_0 \in V$ **Output:** Dictionaries  $D$  and  $\Pi$  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    $D[v] \leftarrow \infty$ ;
4 end
5  $D[u_0] \leftarrow 0$ ;
6  $u^{(0)} \leftarrow u_0$ ;
7 for  $j = 1, \dots, |V| - 1$  do
8    $d^* \leftarrow \infty$ ;
9    $v^* \leftarrow \text{none}$ ;
10  for  $v \in V \setminus S^{(j-1)}$  do
11     $d \leftarrow D[u^{(j-1)}] + w(u^{(j-1)}v)$ ;
12    if  $D[v] > d$  then
13       $D[v] \leftarrow d$ ; // update distance
14       $\Pi[v] \leftarrow u^{(j-1)}$ ; // update predecessor
15    end
16    if  $D[v] < d^*$  then
17       $v^* \leftarrow v$ ;
18       $d^* \leftarrow D[v^*]$ ;
19    end
20  end
21  if  $d^* = \infty$  then
22    return  $D$  and  $\Pi$ ;
23  end
24   $u^{(j)} \leftarrow v^*$ ;
25   $S^{(j)} \leftarrow S^{(j-1)} \cup \{u^{(j)}\}$ ;
26 end

```

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 1.5.1, 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 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  $\Pi$  (invariant 3 for  $j - 1$ ). Note that  $Pu^{(j)}$  is a  $(u_0, u^{(j)})$ -path, and it can be recovered using  $\Pi$  since  $\Pi[u^{(j)}] = u^{(j-1)}$  and  $P$  itself is recovered using  $\Pi$ . 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

$$D[u^{(j)}] = d(u_0, u^{(j)}) \tag{1.10}$$

This equation (1.10) 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  $\Pi$ . This completes the proof. ■

# Chapter 2 Trees

## 2.1 Definition of Trees

We call a graph **acyclic** if it contains no cycles. 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 be 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 1.4.1, 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.

### 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  $P_2$  (regarded 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 1.3.1, 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. ■

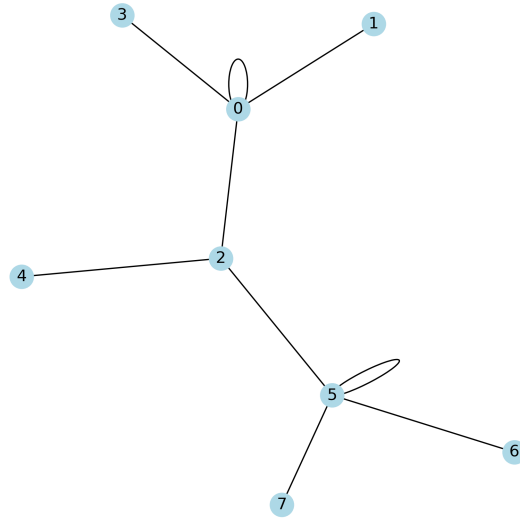
The graph in Figure 2.1 is certainly not a tree. But any two distinct vertices are indeed connected by a unique path. Therefore, for the converse of Theorem 2.1.1 to hold, we need to exclude the cases where graphs contain loops.

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



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

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*

$$|E| = |V| - 1 \quad (2.1) \quad \heartsuit$$

**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. ■

## 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.



# Index

<b>A</b>		<b>O</b>	
acyclic graphs	13	origin of a walk	3
<b>C</b>		<b>P</b>	
complete bipartite graph	2	path in a graph	3
connected and disconnected vertices	3	<b>R</b>	
<b>F</b>		root of a tree	13
free tree	13	rooted tree	13
<b>G</b>		<b>T</b>	
girth	6	terminus of a walk	3
<b>I</b>		trail in a graph	3
isomorphism	1	tree	13
<b>L</b>		trivial walk	3
leaf	13	<b>W</b>	
		walk in a graph	2
		weight of an edge	8