Degree Sequence:

Definition 1.26 The *degree sequence* of a graph is a listing of the degrees of the vertices. It is customary to write these in decreasing order. If a sequence is a degree sequence of a simple graph then we call it *graphical*.

Example 1.18 Explain why neither $4,4,2,1,0$ nor $4,4,3,1,0$ can be graphical.

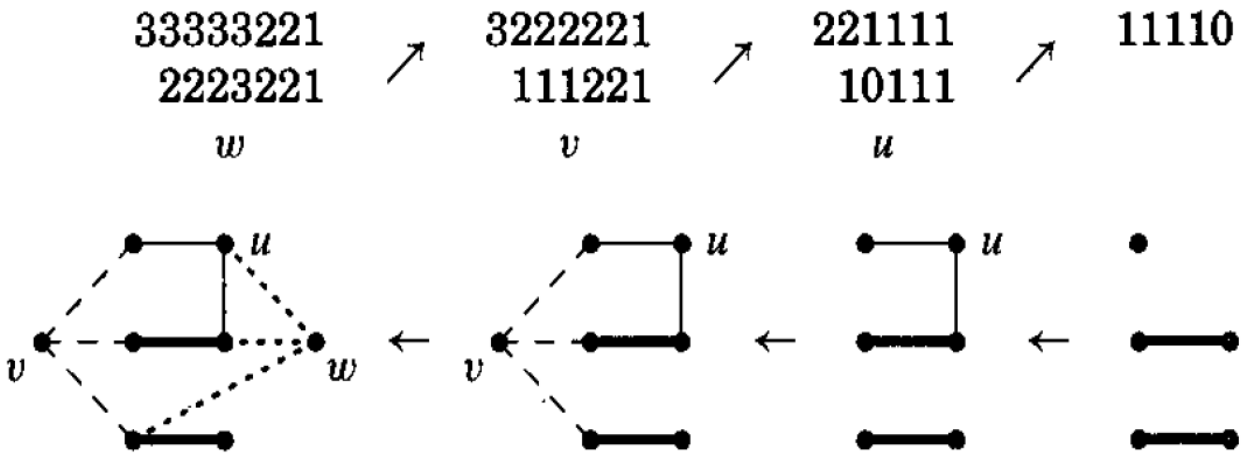
Solution: The first sequence sums to 11, but we know the sum of the degrees of a graph must be even by the Handshaking Lemma. Thus it cannot be a degree sequence.

The second sequence sums to 12, so it is at least even. However, in a simple graph with 5 vertices, a vertex with degree 4 must be adjacent to all the other vertices, which would mean no vertex could have degree 0. Thus the second sequence cannot be a degree sequence.

1.3.28. Proposition. The nonnegative integers $d_1, \ldots, d_n$ are the vertex degrees of some graph if and only if $\sum d_i$ is even.

1.3.30. Example. *A recursive condition.* The lists 2, 2, 1, 1 and 1, 0, 1 are graphit. The graph $K_2 + K_1$ realizes 1, 0, 1. Adding a new vertex adjacent to vertices of degrees 1 and 0 yields a graph with degree sequence 2, 2, 1, 1, as shown below. Conversely, if a graph realizing 2, 2, 1, 1 has a vertex w with neighbors of degrees 2 and 1, then deleting w yields a graph with degrees 1, 0, 1.

The next theorem implies that this recursive test works.



- It is easy to confirm certain sequences are not degree sequences using graph properties (such as the Handshaking Lemma or isolated vertices), but we would like to have a procedure that will always answer this question.
- The theorem below, attributed to the Czech mathematician V´aclav Havel and the Iranian-American mathematician Seifollah Louis Hakimi, provides an iterative approach to answering the graph realization problem.

Theorem 1.27 (Havel-Hakimi Theorem) An increasing sequence $S : s_1, s_2, \ldots, s_n$ (for $n \geq 2$) of nonnegative integers is graphical if and only if the sequence

$$S' : s_2 - 1, s_3 - 1, \ldots, s_{s_1} - 1, s_{s_{n+1}}, \ldots, s_n$$

is graphical.

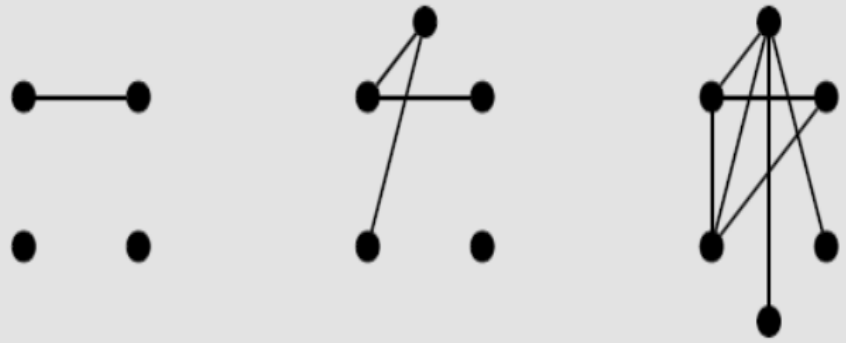
- The intention behind this procedure is to peel of the first vertex, removing one from the degrees of its neighbors.
- We continue this process until we arrive at a sequence that either can easily be turned into a graph or contains negative integers and therefore cannot be a degree sequence.
- Since the theorem is a biconditional statement, we know that any sequence that appeared throughout this iterative process has the same graphical answer as the last sequence considered.
- Note, we may need to reorder the vertices after the peeling procedure to ensure the sequence remains decreasing.

Example 1.19 Determine if either of $S : 4, 4, 2, 1, 1, 0$ or $T : 4, 3, 3, 2, 1, 1$ is graphical.

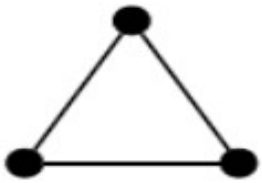
Solution: Applying the Havel-Hakimi Theorem to S, we note the first term of the sequence is 4, and so we eliminate the first term and subtract 1 from

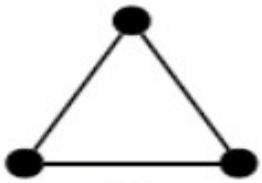
the next 4 terms and leaving the last one alone. This gives $S_1 : 3, 1, 0, 0, 0$. In this new sequence the first term is 3, so we eliminate it and subtract 1 from the next 3 terms of S_1, producing $S_2 : 0, -1, -1, 0$. This last sequence cannot be graphical since degrees cannot be negative. Thus S_1 and S cannot be graphical either.

Using the same procedure on T, after the first iteration we get $2, 2, 1, 0, 1$. We reorder this to make it decreasing as $T_1 : 2, 2, 1, 1, 0$. After the second iteration we have $1, 0, 1, 0$, which is again reordered to $T_2 : 1, 1, 0, 0$. At this point we can stop since it is not too difficult to see this sequence is graphical: the two vertices of degree 1 are adjacent and the other two vertices are isolated (see below). This means that T_2, T_1, and the original sequence T are all graphical.

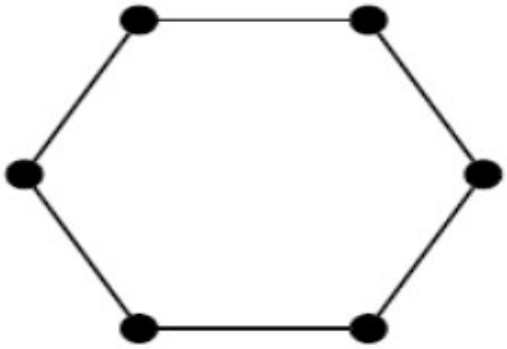


T_2 T_1 T





G_5



G_6

- Graph G_5 contains two pieces, each of which is the graph K_3 (these pieces are called components, which we formally define in the next chapter). Its degree sequence is 2, 2, 2, 2, 2, 2.
- Note G_6 has the same degree sequence but is just one big circle (which we will define as a cycle in the next chapter).

✓ It should be fairly obvious that these two graphs are not isomorphic. Thus degree sequences cannot uniquely determine a graph.

Algorithm 2.5.7. Let $d = (d_1, d_2, ..., d_n)$ be a sequence of n integers.

Step 1: If $d_i \geq n$ for some $i = 1, 2, ..., n$, then $\boldsymbol{d}$ is not graphic.

Step 2: If each term in the current sequence is '0', then $\boldsymbol{d}$ is graphic.

Step 3: If there is a 'negative' term in the current sequence, then $\boldsymbol{d}$ is not graphic.

Step 4: Arrange the current sequence in non-increasing order.

Step 5: Let 'r' be the first term of the current sequence. Form a new sequence by deleting 'r' and subtracting '1' from each of the next r terms. Go to Step 2.

Example 2.5.8. Determine if the sequence $(5, 4, 4, 3, 1, 1)$ is graphic.
By applying Algorithm 2.5.7, we have:

$$\begin{array}{cccccc} 5 & 4 & 4 & 3 & 1 & 1 \\ & 3 & 3 & 2 & 0 & 0 \\ & & 2 & 1 & -1 & 0 \end{array}$$

As the last sequence contains a negative term (namely, -1), the given sequence is not graphic.

Example 2.5.9. Determine if the sequence $(4, 4, 3, 2, 2, 1)$ is graphic.

By applying Algorithm 2.5.7, we have:

$$\begin{array}{cccccc} 4 & 4 & 3 & 2 & 2 & 1 \\ & 3 & 2 & 1 & 1 & 1 \\ & & 1 & 0 & 0 & 1 \\ & & 1 & 1 & 0 & 0 \\ & & & 0 & 0 & 0 \end{array}$$

As the last sequence consists of '0' only, the given sequence is graphic.

Example 2.5.11. By Example 2.5.9, the sequence $(4, 4, 3, 2, 2, 1)$ is graphic. Construct a graph representing it.

The following is what we have obtained by applying Algorithm 2.5.7:

$$\begin{array}{cccccc} 4 & 4 & 3 & 2 & 2 & 1 \\ & 3 & 2 & 1 & 1 & 1 \\ & & 1 & 0 & 0 & 1 \\ & & 1 & 1 & 0 & 0 \end{array}$$

It is clear that $(1, 1, 0, 0)$ is representable by the graph G_1 shown in Fig. 2.5.3.

Fig. 2.5.3: G_1

A graph representing $(3, 2, 1, 1, 1)$, shown in Fig. 2.5.4, can then be obtained using the method mentioned above.

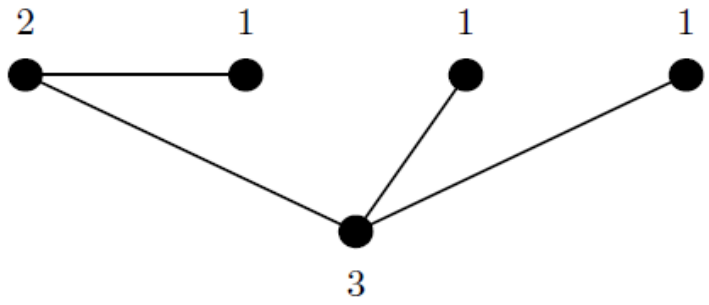


Fig. 2.5.4: G_2

Repeating this procedure, we finally obtain the following graph G representing the sequence $(4, 4, 3, 2, 2, 1)$

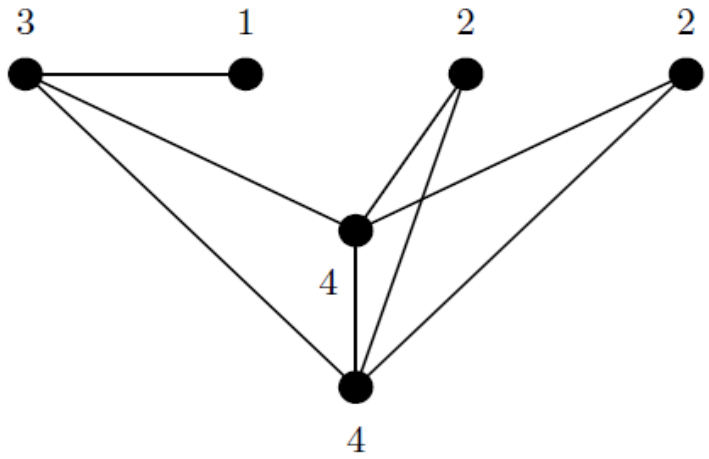


Fig. 2.5.5: G

Questions:

1.3.8. (–) Which of the following are graphic sequences? Provide a construction or a proof of impossibility for each.

a) (5,5,4,3,2,2,2,1), c) (5,5,5,3,2,2,1,1),
b) (5,5,4,4,2,2,1,1), d) (5,5,5,4,2,1,1,1).

1. Determine if each of the following sequences is graphic, and if it is so, construct a graph representing it.

   (i) $(5, 4, 3, 3, 2, 2, 1)$
   (ii) $(5, 4, 4, 4, 3, 2, 2)$
   (iii) $(5, 5, 4, 4, 3, 2, 1)$
   (iv) $(6, 5, 4, 3, 3, 3, 3, 1)$
   (v) $(4, 4, 3, 3, 3, 3, 2, 2, 2)$
   (vi) $(9, 7, 6, 4, 3, 3, 3, 1, 1, 1)$
   (vii) $(8, 6, 6, 5, 5, 5, 4, 3, 3, 3, 2)$

Self-Complementary:

A graph G is said to be **self-complementary** if $\overline{G} \cong G$.

11. For each of the graphs in Fig. 2.4.5,

(i) construct its complement and

(ii) determine if it is self-complementary.

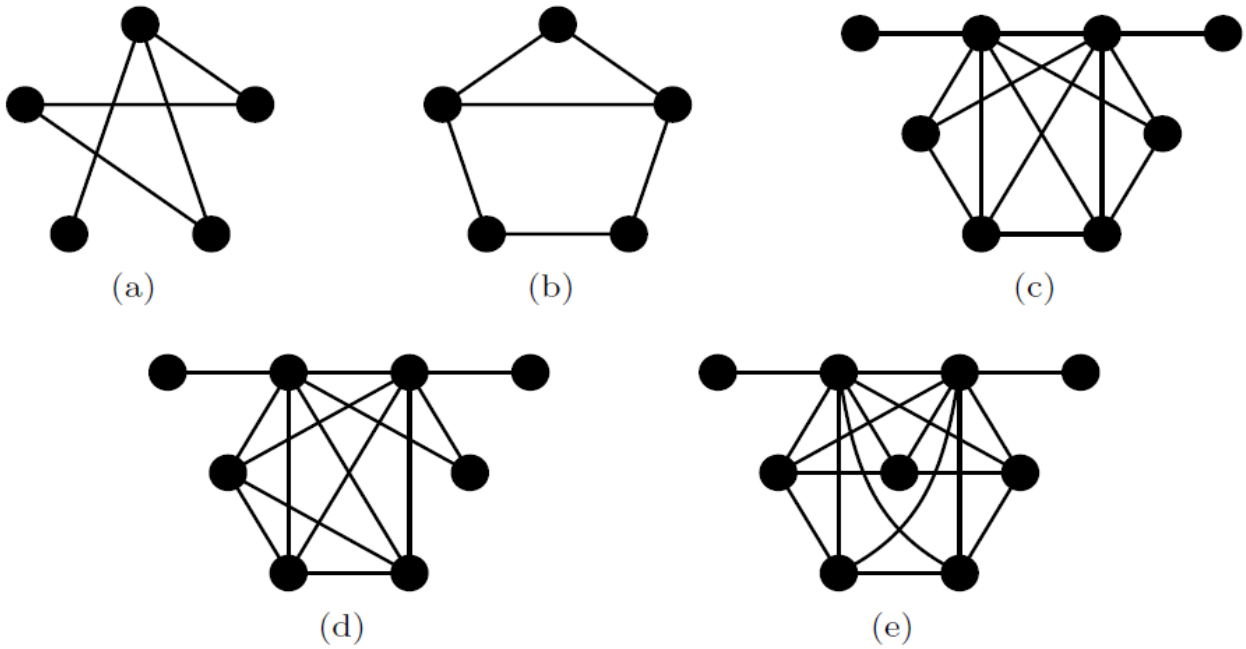


Fig. 2.4.5

TOURING A GRAPH:

- Part of Euler's brilliance is how he figured out how to model a real-world question with a mathematical object that we now call a graph.
- A part of the modeling process is determining what the answer we are searching for looks like in graph form. What type of structure or operation are we looking for?
- In our original search for a solution to the K¨onigsberg Bridge Problem, we discuss traveling through the city.
- What would traveling through a graph mean?

Definition 2.1 Let G be a graph.

- A ***walk*** is a sequence of vertices so that there is an edge between consecutive vertices. A walk can repeat vertices and edges.

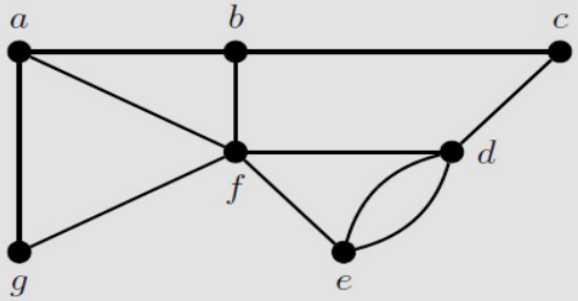
- A ***trail*** is a walk with no repeated edges. A trail can repeat vertices but not edges.
- A ***path*** is a trail with no repeated vertex (or edges). A path on n vertices is denoted P_n.
- A ***closed walk*** is a walk that starts and ends at the same vertex.
- A ***circuit*** is a closed trail; that is, a trail that starts and ends at the same vertex with no repeated edges though vertices may be repeated.
- A ***cycle*** is a closed path; that is, a path that starts and ends at the same vertex. Thus cycles cannot repeat edges or vertices. Note: we do not consider the starting and ending vertex as being repeated since each vertex is entered and exited exactly once. A cycle on n vertices is denoted C_n.

The ***length*** of any of these tours is defined in terms of the number of edges. For example, P_n has length $n-1$ and C_n has length n.

REMARKS:

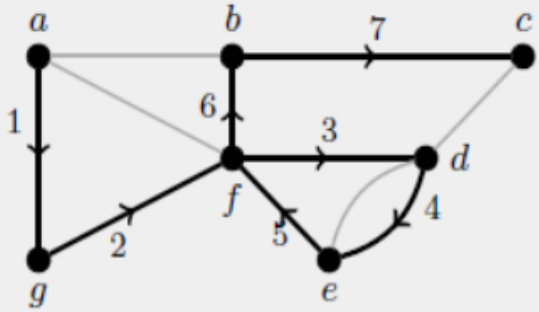
- A path is a more restrictive version of a trail.
- A trail is a more restrictive form of a walk.
- Any path can also be viewed as a trail and as a walk.
- A walk might not be a trail or a path (for example, if it repeats vertices or edges).
- A cycle is a circuit and a closed walk.

Example 2.1 Given the graph below, find a trail (that is not a path) from a to c, a path from a to c, a circuit (that is not a cycle) starting at b, and a cycle starting at b.

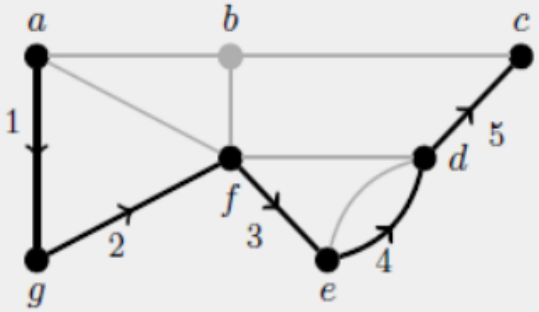


Solution: The order in which the edges will be traveled are noted in the following routes.

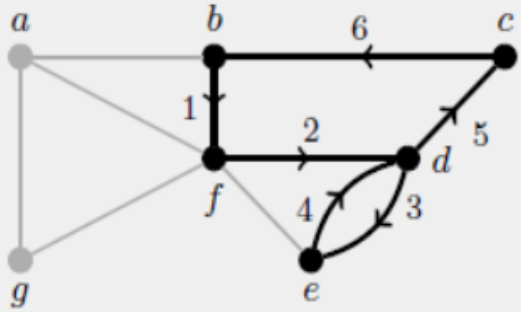
Trail from a to c



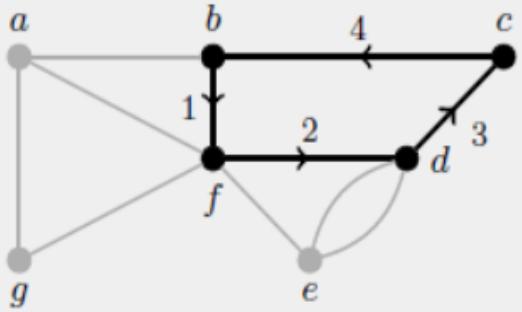
Path from a to c



Circuit starting at b



Cycle starting at b



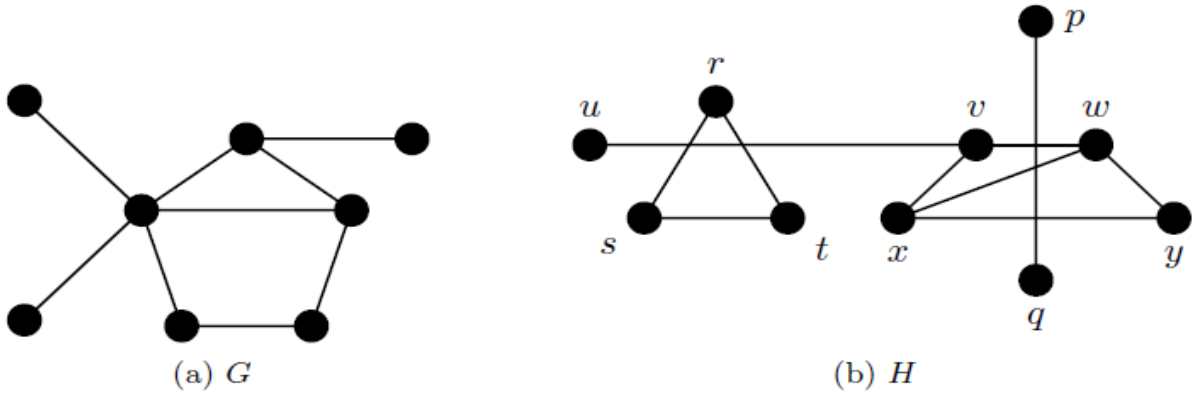
Note that the trail from a to c is not a path since the vertex f is repeated and the circuit starting at b is not a cycle since the vertex d is repeated. Moreover, the examples given above are not the only solutions but rather an option among many possible solutions.

CONNECTED GRAPHS:

Definition 2.2 Let G be a graph. Two vertices x and y are ***connected*** if there exists a path from x to y in G. The graph G is ***connected*** if every pair of distinct vertices is connected.

- Let G be a multigraph and u; v be two vertices in G. We say that u and v are connected if they are joined by a path.
- If every two vertices in G are connected, we say that G is connected.
- A multigraph is said to be disconnected if it is not connected.

Example 1.4.11. There are two graphs G and H in Fig. 1.4.4. It can easily be checked that every two vertices in G are connected. Thus G is a connected graph. However, the graph H is not connected since, for instance, the vertices r and u are not connected.



(a) G (b) H

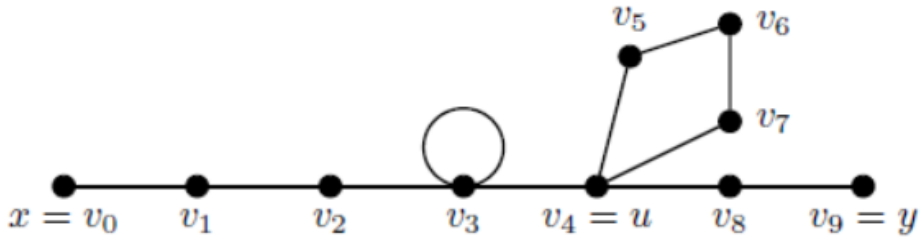
Fig. 1.4.4

✓ Thus, the graph H is Fig. 1.4.4(b) is disconnected.

Theorem 2.3 Every $x - y$ walk contains an $x - y$ path.

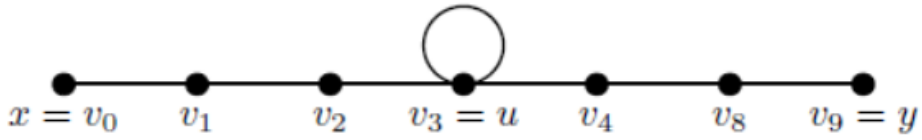
Consider a walk as shown below. Using the technique from the proof above, we begin by identifying a repeated vertex, namely v_4.

$$W : v_0\ v_1\ v_2\ v_3\ v_3\ \boxed{v_4\ v_5\ v_6\ v_7\ v_4}\ v_8\ v_9$$



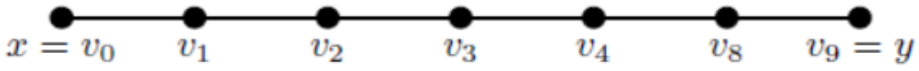
Notice that these selected edges and vertices can be removed without destroying the walk from v_0 to v_9, as shown below. Also, we find another repeated vertex, namely v_3, and can repeat the process.

$$W' : v_0\ v_1\ v_2\ \boxed{v_3\ v_3}\ v_4\ v_8\ v_9$$



The final graph below does not contain any repeated vertices and so can be considered a path from v_0 to v_9, all of whose vertices and edges were contained in the original walk W.

$$W'' : v_0\ v_1\ v_2\ v_3\ v_4\ v_8\ v_9$$



The theorem above shows that the underlying nature of graphs lend themselves to induction proofs. This is in part because we often want to remove a portion of the graph, whether it is a vertex, edge, or something more complex, and consider how that effects the graph.

2. Consider the following multigraph G.

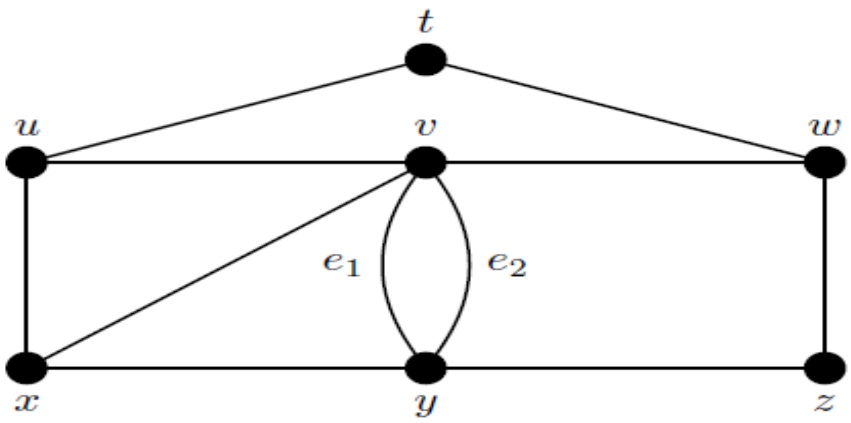


Fig. 1.4.9

(a) For $k = 2, 3, 4, 5, 6, 7$, find a cycle of length k in G.
(b) Find a circuit of length 6 in G that is not a cycle.
(c) Find a circuit of length 8 in G that does not contain t.
(d) Find a circuit of length 9 in G that contains t and v.

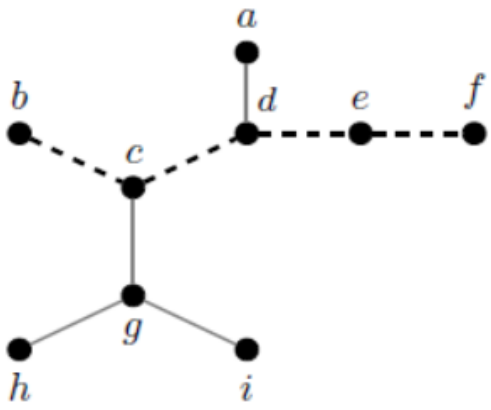
Definition 2.4 An object X is *maximum* if it is the largest among all objects under consideration; that is, $|X| \geq |A|$ for all $A \in \mathcal{U}$.

An object X is *maximal* if it cannot be made larger.

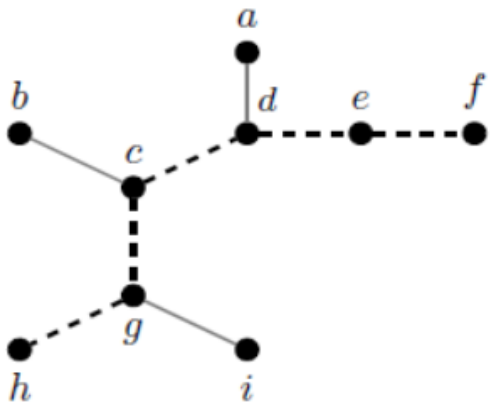
- In certain scenarios, the adjectives maximum and maximal may be applied to the same object; however, this need not be true.
- Consider the graphs shown below. The path b c d e f (highlighted on the left) is maximal because we cannot add any additional vertices and keep it a path.

However, it is not maximum since a longer path can be found, namely h g c d e f shown on the right.

- Note that this second path is a maximum path within the graph shown.



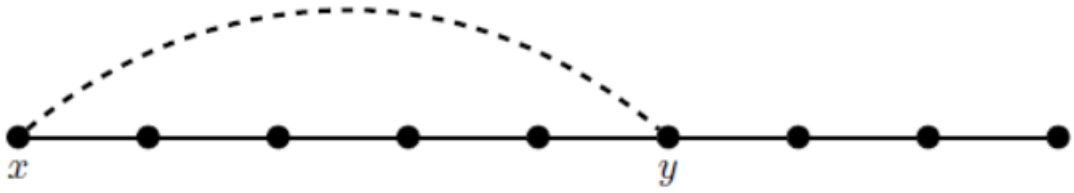
Maximal $b-f$ path



Maximum path in G

Theorem 2.5 If every vertex of a graph has degree at least 2 then G contains a cycle.

Proof: Let P be a maximal path in G and let x be an endpoint of P. Since P is maximal, it cannot be extended and so every neighbor of x must already be a vertex of P. Since x has degree at least 2, we know there must be another neighbor y of x in $V(P)$ via an edge not a part of the path.



Then the edge xy completes a cycle with the portion of P from x to y.

Remark:

- Paths and cycles serve as the building blocks to many structures we wish to find within a larger graph.
- As seen above, the presence of a path between any two vertices indicates the graph is connected.
- What about circuits or cycles? Can a cycle contain every vertex? every edge?

The remainder of this section will focus on the edges (through Eulerian circuits) whereas the next section will explore vertices (through Hamiltonian cycles).

EULERIAN GRAPHS:

Looking back at the K¨onigsberg Bridge Problem, we now have the necessary pieces to describe the question in graph theoretic terminology, namely an exhaustive circuit that includes all vertices and edges of the graph.

- In honor of Euler's solution, these special types of circuits bear his name.

Definition 2.6 Let G be a graph. An *eulerian circuit* (or *trail*) is a circuit (or trail) that contains every edge and every vertex of G.

If G contains an eulerian circuit it is called *eulerian* and if G contains an eulerian trail but not an eulerian circuit it is called *semi-eulerian*.

EULER CIRCUIT: A circuit W in G is called an Euler circuit if W contains all the edges in G.

EULERIAN MULTIGRAPH: The multigraph G is called an Eulerian multigraph if G possesses an Euler circuit.

Example 4.1.1. The multigraph of Fig. 4.1.1(a) is Eulerian as it has an Euler circuit.

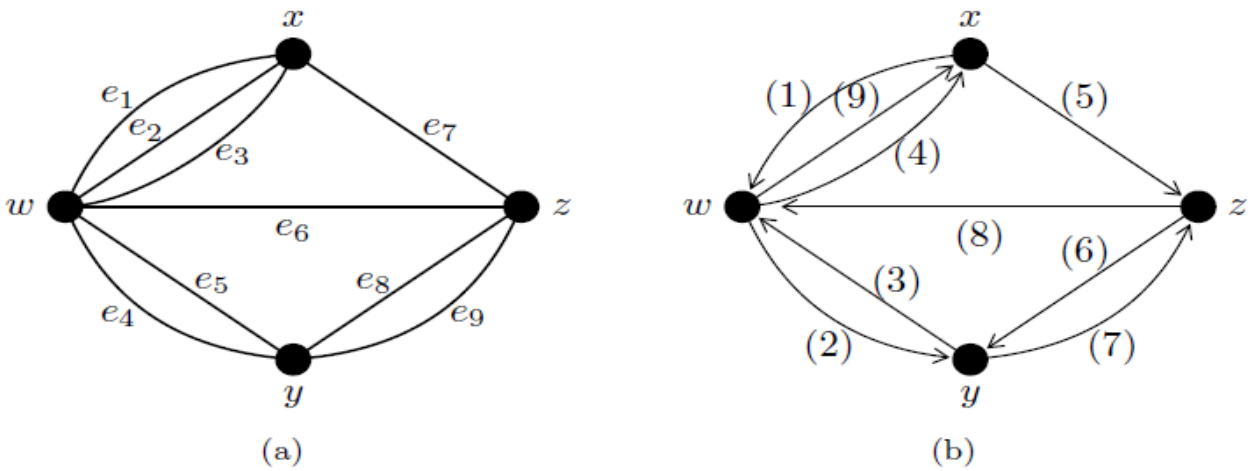


Fig. 4.1.1

QUESTIONS:

Question 4.1.2. *Show that each of the following multigraphs is Eulerian by exhibiting an Euler circuit. Is there an odd vertex in any of the multigraphs?*

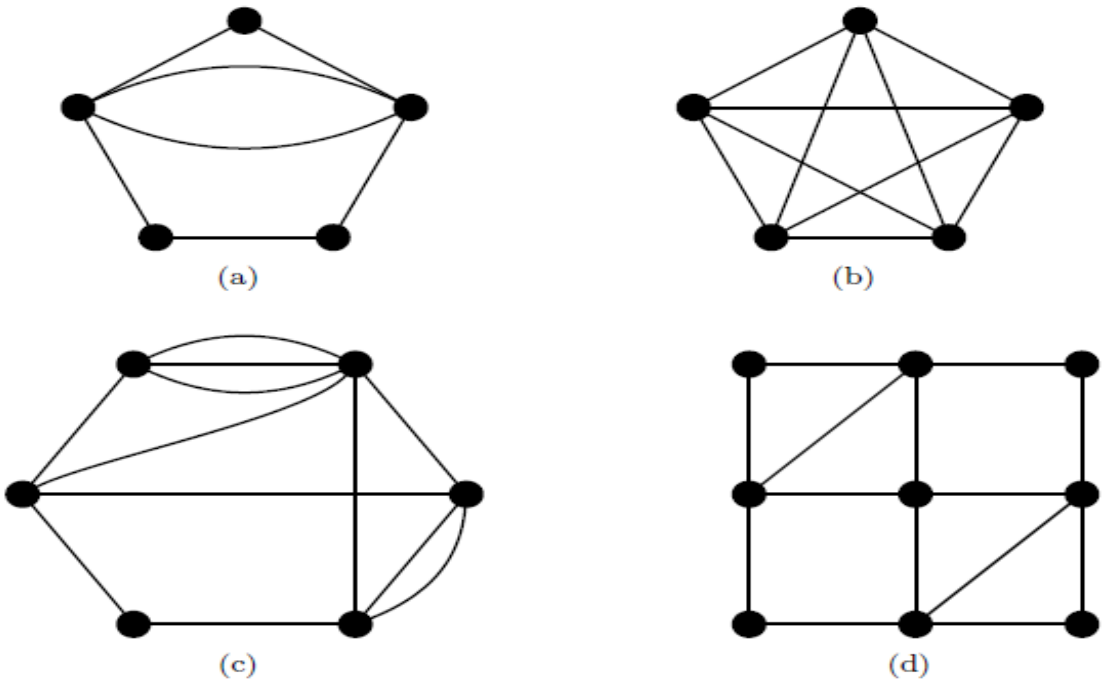


Fig. 4.1.2

Question 4.1.3. *Are the following multigraphs Eulerian? Are there any odd vertices in each multigraph?*

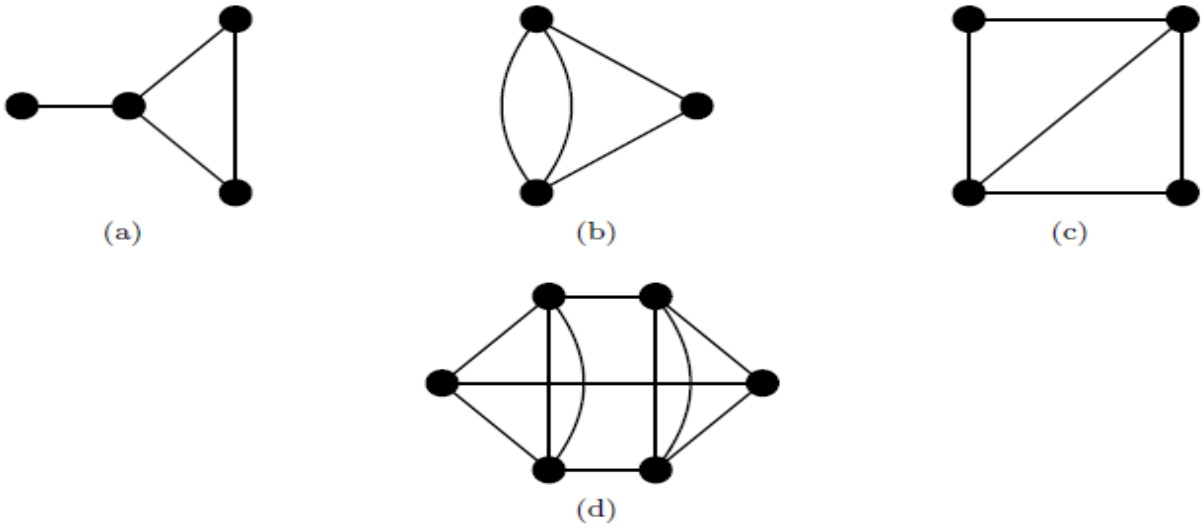


Fig. 4.1.3

Characterization of Eulerian Graphs:

Theorem 2.7 A graph G is eulerian if and only if

(i) G is connected and

(ii) every vertex has even degree.

Corollary 2.8 A graph G is semi-eulerian if and only if

(i) G is connected and

(ii) exactly two vertices have odd degree.

Example 2.3 Consider the graphs appearing in the examples from this and the previous chapter. Which ones are eulerian? semi-eulerian? neither?

Solution:

- Even though the graph in Example 2.1 is connected, it is neither eulerian nor semi-eulerian since it has more than two odd vertices (namely, $a, b, e,$ and f).
- The graph representing Island City in Example 2.2 is not connected, so it is neither eulerian nor semi-eulerian.
- The graph representing Königsberg is neither eulerian nor semi-eulerian since all four vertices are odd.
- The graph in Example 1.1 is neither eulerian nor semi-eulerian since it is not connected.
- The graph in Example 1.7 is eulerian since it is connected and all the vertices have degree 4.
- The graph in Example 1.11 is semi-eulerian since it is connected and exactly two vertices are odd (namely, 372 and 271).

PROPERTIES OF EULERIAN GRAPHS:

- Let G be a connected multigraph, and u and v be the only two odd vertices in G. Then the number of u - v paths in G is odd.
- If G is a connected Eulerian multigraph, then any edge in G is contained in an odd number of cycles.
- Let G be a connected multigraph. Then G is Eulerian if and only if every edge in G is contained in an odd number of cycles.

ALGORITHMS:

Fleury's Algorithm

Input: Connected graph G where zero or two vertices are odd.

Steps:

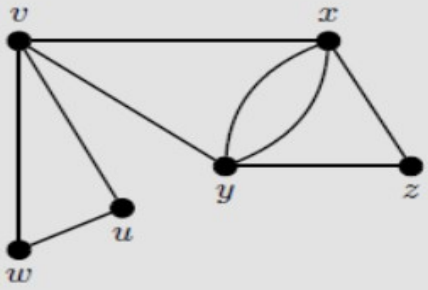
1. Choose a starting vertex, call it v. If G has no odd vertices, then any vertex can be the starting point. If G has exactly two odd vertices, then v must be one of the odd vertices.
2. Choose an edge incident to v that is unlabeled and label it with the number in which it was chosen, ensuring that the graph consisting of unlabeled edges remains connected.
3. Travel along the edge to its other endpoint.
4. Repeat Steps (2) and (3) until all edges have been labeled.

Output: Labeled eulerian circuit or trail.

✓ The intention behind Fleury's Algorithm is that you are prevented from getting stuck at a vertex with no edges left to travel.

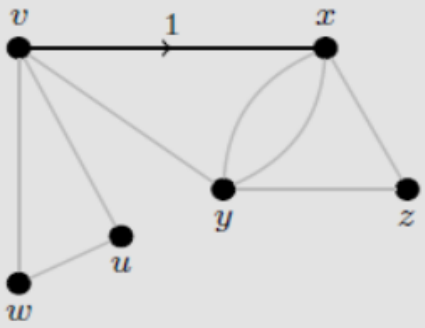
Example:

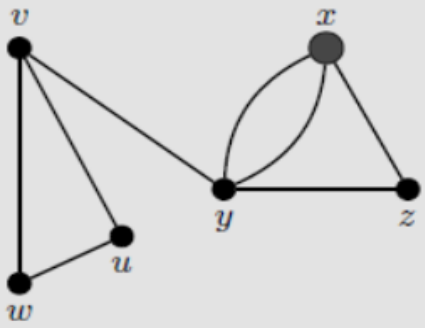
Example 2.4 Input: A connected graph (shown below) where every vertex has even degree. We are looking for an eulerian circuit.



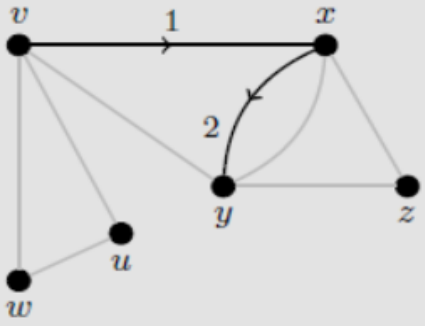
Step 1: Since no starting vertex is explicitly stated, we choose vertex v to be the starting vertex.

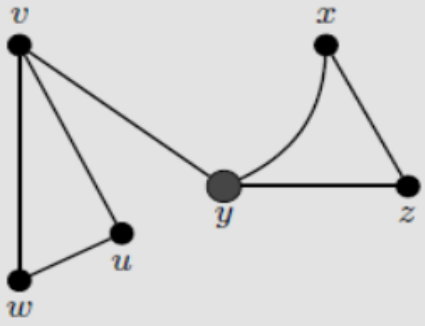
Step 2: We can choose any edge incident to v. Here we chose vx. The labeled graph is on the left and the unlabeled portions are shown on the right with edges removed that have already been chosen.



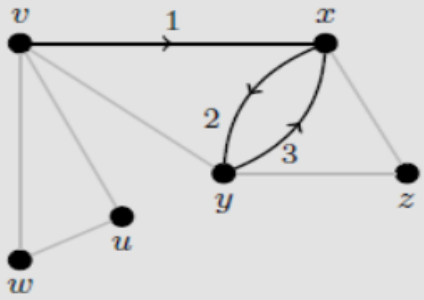


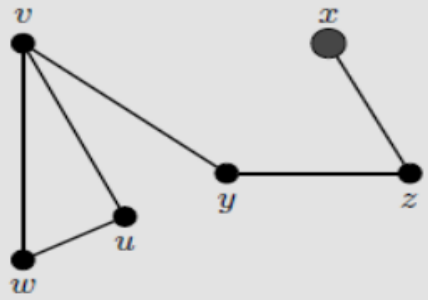
Step 3: Looking at the graph to the right, we can choose any edge out of x. Here we chose xy. The labeled and unlabeled graphs have been updated below.



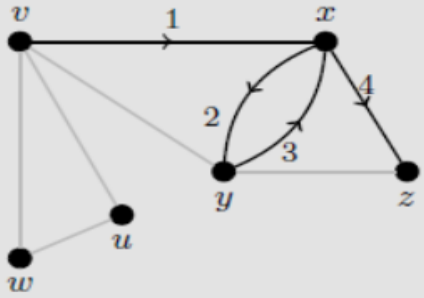


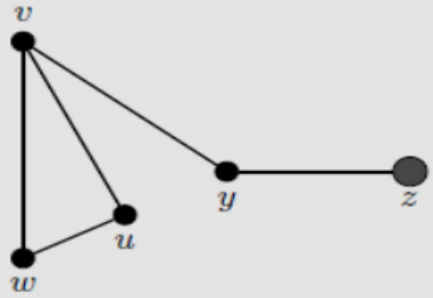
Step 4: At this point we cannot choose yv, as its removal would disconnect the unlabeled graph shown on the right in Step 3. However, yx and yz are both valid choices. Here we chose yx.



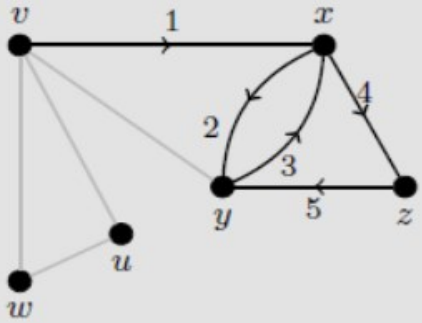


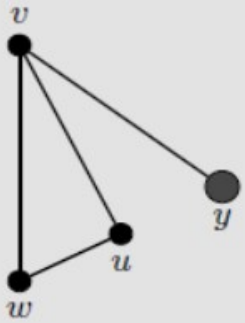
Step 5: There is only one available edge xz.



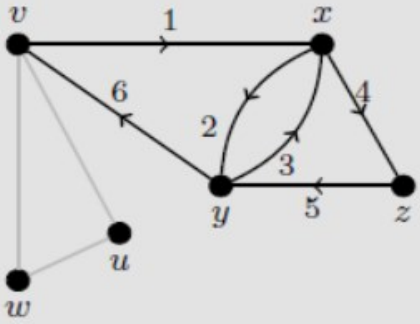


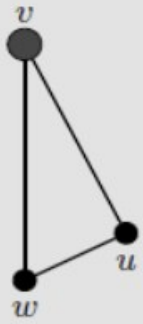
Step 6: There is only one available edge zy.



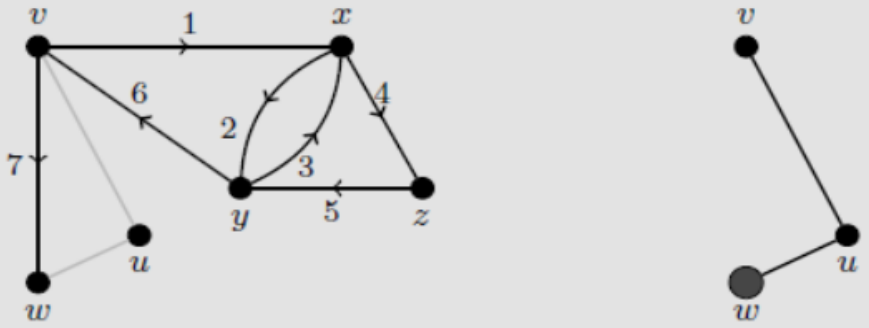


Step 7: There is only one available edge yv.

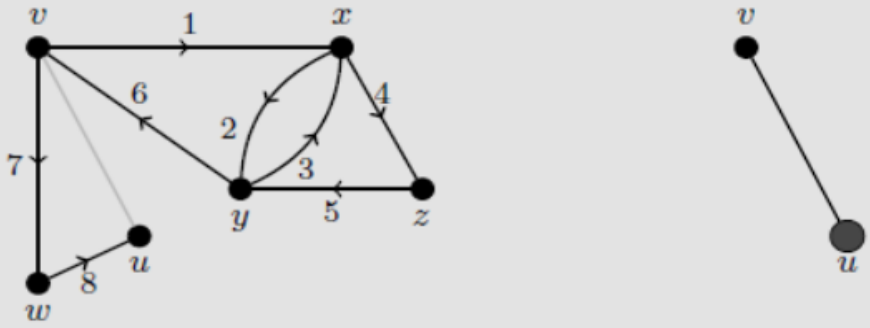




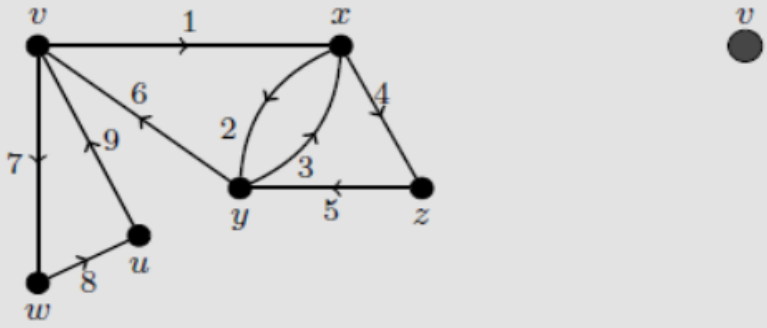
Step 8: Both vw and vu are valid choices for the next edge. Here we chose vw.



Step 9: There is only one available edge wu.



Step 10: There is only one available edge uv.



Output: The graph above on the left has an eulerian circuit labeled, starting and ending at vertex v.

- The second algorithm we study, Hierholzer's Algorithm, is named for the German mathematician mentioned earlier whose paper inspired the procedure to follow.
- This efficient algorithm begins by finding an arbitrary circuit originating from the starting vertex. If this circuit contains all the edges of the graph, then an Eulerian circuit has been found. If not, then we join another circuit to the existing one.

Hierholzer's Algorithm

Input: Connected graph G where all vertices are even.

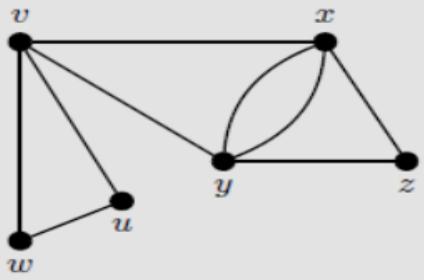
Steps:

1. Choose a starting vertex, call it v. Find a circuit C originating at v.
2. If any vertex x on C has edges not appearing in C, find a circuit C' originating at x that uses two of these edges.
3. Combine C and C' into a single circuit C^*.
4. Repeat Steps (2) and (3) until all edges of G are used.

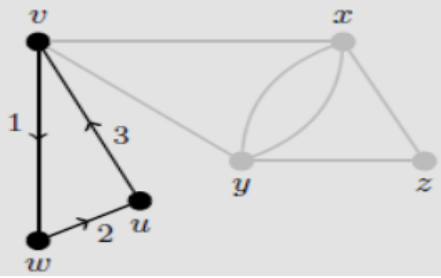
Output: Labeled eulerian circuit.

- Note that Hierholzer's Algorithm requires the graph to be Eulerian, whereas Fleury's Algorithm allows for the graph to be Eulerian or semi-Eulerian.

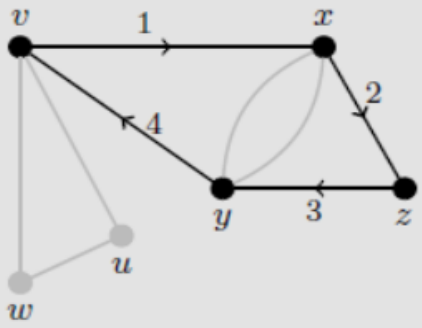
Example 2.5 Input: A connected graph where every vertex has even degree. We are looking for an eulerian circuit.



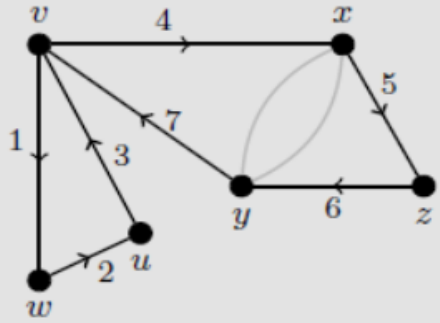
Step 1: Since no starting vertex is explicitly stated, we choose v and find a circuit originating at v. One such option is highlighted below.



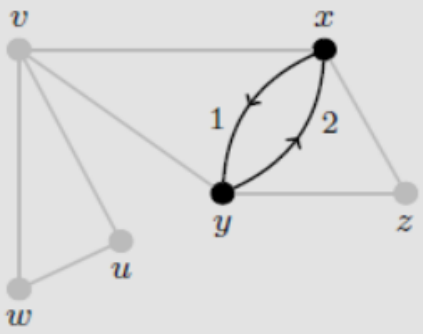
Step 2: As $\deg(v) = 4$ and two edges remain for v (shown in gray in the previous figure), a second circuit starting at v is needed. One option is shown below.



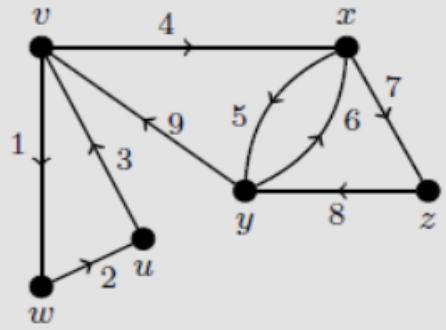
Step 3: Combine the two circuits from Step 1 and Step 2. There are multiple ways to combine two circuits, but it is customary to travel the first circuit created and then travel the second.



Step 4: As $\deg(x) = 4$ and two edges remain for x (shown in gray above), a circuit starting at x is needed. It is shown below.



Step 5: Combine the two circuits from Step 3 and Step 4.

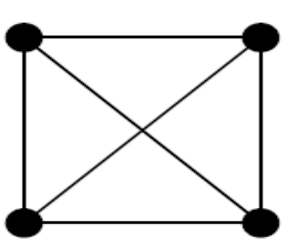


Output: The graph above gives a labeled eulerian circuit originating at v.

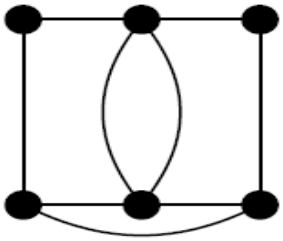
- There are advantages and disadvantages for these two methods; in particular, both Fleury's and Hierholzer's Algorithms will find an Eulerian circuit when one exists, whereas only Fleury's can be used to find an Eulerian trail (see Exercise 2.10 for a modification of Hierholzer's that will find an eulerian trail).

Questions:

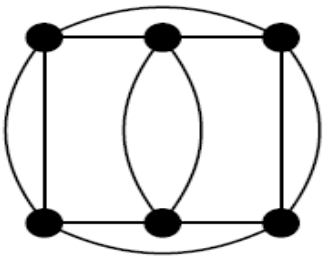
1. Determine whether the following multigraphs are Eulerian, semi-Eulerian or neither:



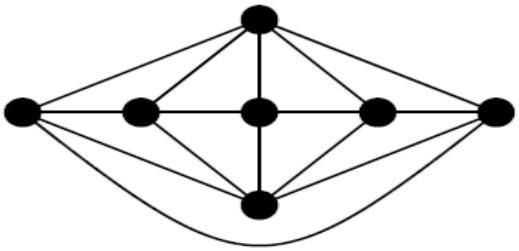
(a)



(b)

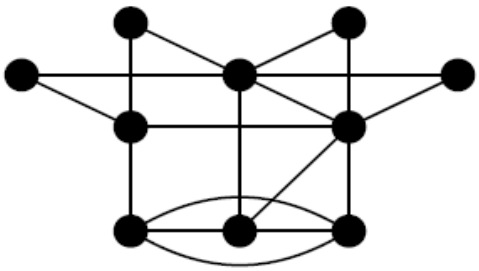


(c)

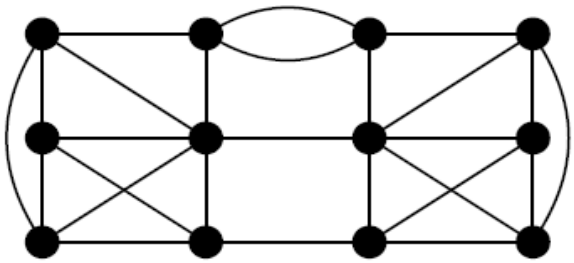


(d)

1. Apply both Algorithm 4.4.1 (Hierholzer's algorithm) and Algorithm 4.4.3 (Fleury's algorithm) to find an Euler circuit in each of the following Eulerian multigraphs:



(a)



(b)

Fig. 4.4.5

2. Identify two odd vertices in the following semi-Eulerian multigraph. Then apply Algorithm 4.4.3 (Fleury's algorithm) to find an Euler trail in the multigraph.