Seminararbeit

Oriented Trees

von
Florian Guthmann
Matrikel-Nr.: 23013034
Betreuung:
Prof. Oliver Keszöcze

11. März 2023

This paper aims at examining undirected graphs and directed graphs and defining for both the notion of a graph that behaves in a "tree-like" manner. We will define several properties of undirected and directed graphs and examine the definitions for trees as they are presented in Donald Knuth's The Art of Computer Programming [3]. We will then see how these notions help us by looking at eulerian circuits in directed graphs. First, by proving their existence in directed graphs that have the right properties, and then by showing how an eulerian circuit can be constructed in such a graph.

1 Fundamentals in Graph Theory

We will first examine some rudimentary properties in graph theory, looking both at undirected and directed graphs.

1.1 Undirected Graphs

Definition. A finite *undirected graph* $G = \langle V, E \rangle$ consists of

- a finite set of *vertices* V
- a finite set of *edges* E

This definition is best illustrated by the example undirected graph in Figure1.1:

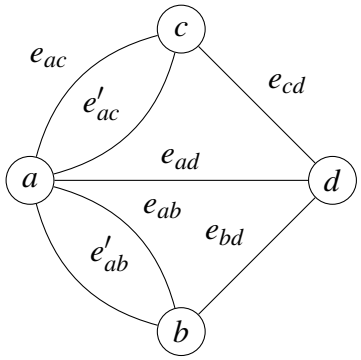


Figure 1.1: An undirected graph

We can see that the set V of vertices is $\{a,b,c,d\}$. Furthermore, we can specify the set E of edges to be $\{e_{ab}, e'_{ab}, e_{ac}, e'_{ac}, e_{ad}, e_{bd}, e_{cd},\}$. It is natural to talk about edges as being *between* two vertices, so for example the edge e_{ab} is between vertices a and b. It might be worth mentioning also that, unlike many other sources, we do not restrict our definition of an edge such that there exists at most one edge between any two vertices. This is illustrated by edges e_{ac} and e'_{ac} which are both between a and c. For this very reason what we talk about as undirected graphs may be called undirected multigraphs in other sources such as Rahman [4].

Definition. Two vertices v and w in an undirected graph G are *adjacent*, if there is an edge between them.

If we take Figure 1.1 as an example, we can see that vertices a and c are adjacent, since there is an edge between them. Vertices b and c are not adjacent. It is obvious however, that they are, in a way, "connected" if we can string sequences of edges together to create such a connection. We will now specify this notion as paths in undirected graphs.

1.1.1 Paths

Definition. Let v_0 and v_n be vertices in an undirected graph G. Then $(v_0, v_1, \ldots, v_n)$, with $0 \leq n$ is a path of length n in G if for $0 \leq k < n$ it holds that v_k is adjacent to v_{k+1}.

For vertices v and w we will refer to a path *from v to w* as a path with $v_0 = v$ and $v_n = w$.

See how with this definition, again taking Figure 1.1 as an example, we can form the path (b,d,c) which gives us our desired "connection" between vertices b and c which we previously missed since they are not in fact adjacent. Notice also how the path from b to c does not have to be unique. In this case we can also form the path (b,a,c). We do not, however, distinguish between multiple edges between the same vertices, which the path (a,c) illustrates.

Notice that in all our example paths so far, so $(b,d,c),(b,a,c),(a,c)$, each vertex occurs at most once. We can specify this notion, which will later prove useful.

Definition. A path is *simple*, if $v_0, v_1, \ldots, v_n$ are distinct.

To see an example of a path in Figure 1.1 that is *not* simple, consider (a,c,d,a,b) where a occurs twice.

With our definition of a simple path, we can determine another property paths can exhibit. We want to talk about paths that loop, that is paths that start at the same vertex that they end in.

Definition. A *cycle* is a path from a vertex to itself, i.e. where $v_0 = v_n$, and $v_1, \ldots, v_n$ are distinct.

Take, for instance, the path (a,b,c,d,a) in Figure 1.1. We can see that it a cycle since it starts and ends in a and all other vertices are distinct.

For the most part, we will be interested not in cycles themselves, but rather graphs that do not contain them. We will later see why, as graphs without cycles exhibit properties that we could not otherwise guarantee.

1.1.2 Properties of Undirected Graphs

Now let us define some additional properties that undirected graphs may satisfy.

Definition. An undirected graph G is *connected*, if there is a path between any two vertices in G.

This generalises our notion of two vertices being "connected" for all vertices in a graph. It is easy to visualise a connected graph as one that contains only one component where all vertices are connected and no vertices that are outside or isolated. So if G is a connected graph, we may also refer to it as a graph without isolated vertices.

Our working example, Figure 1.1, is a connected graph which we can verify visually or by giving paths from v to w for every unordered pair of vertices (v, w).

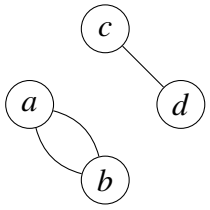


Figure 1.2: A graph that is not connected

In Figure 1.2 we observe a graph that is not connected. Notice how it does not suffice to define connected graphs as those that do not contain vertices that have no other adjacent vertices. Our graph would satisfy this property, which is not what we want. It is in fact not connected as there is, for instance, no path from a to c

Now we can formalise our notion of an undirected graph that has a tree-like structure.

Definition. A *free tree* is a connected undirected graph that contains no cycles.

This is perhaps a more general definition than one might expect, considering that we do not even specify the "root" of such a tree. In fact, Knuth also calls them *unrooted trees* [3]. It is, however, a concept that is often useful in computer science, as we will observe later.

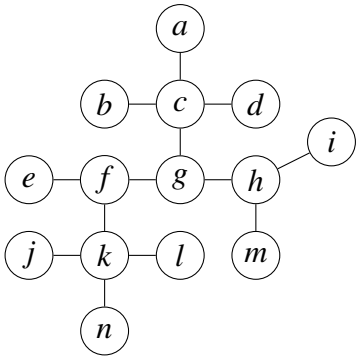


Figure 1.3: A free tree

1.2 Directed Graphs

If we take our definition of undirected graphs and extend edges by introducing the notion of direction. We will thus cease to call them edges, rather calling them *arcs*. In illustrations, this direction will be visualised by arrows on arcs.

Definition. A finite *directed graph* $G = \langle V, A \rangle$ consists of

- a finite set of *vertices* V
- a finite set of *arcs* A

For each arc a we have

- an initial vertex, denoted by $\text{init}(a)$
- a final vertex, denoted by $\text{fin}(a)$

Note that in contrast to the definition of an edge in an undirected graph, we do not exclude the case of an arc where initial and final are one and the same vertex, i.e. $V = \text{init}(a) = \text{fin}(a)$. Additionally, nothing prevents us from having several arcs with the same initial and final vertex, e.g. arcs a, b with $\text{init}(a) = \text{init}(b)$ and $\text{fin}(a) = \text{fin}(b)$. Our notion of a directed graph thus corresponds to a finite directed multigraph or *quiver*. [2] Given a directed graph, we can define its corresponding undirected graph by ignoring the direction of arcs. That is, instead of an arc a we take an edge e as the unordered pair of $\text{init}(a)$ and $\text{fin}(a)$.

With that, we can take a look at the relationship between arcs and vertices.

Definition. Given a vertex v in a directed graph G, we can define:

$\text{in-degree}(v)$ as the number of arcs a in G with $\text{init}(a) = v$

$\text{out-degree}(v)$ as the number of arcs a in G with $\text{fin}(a) = v$

Definition. A directed graph G is *balanced* if every vertex v in G has the same in-degree and out-degree

As an example, consider the directed graph in Figure 1.4, where for each vertex the in-degree is labeled as I and the out-degree is labeled as O. As both values are equal for each vertex, the graph is balanced. The same cannot be said about the graph in Figure 1.5, where for both vertices b and c the in-degree does not equal the out-degree. The graph is therefore not balanced.

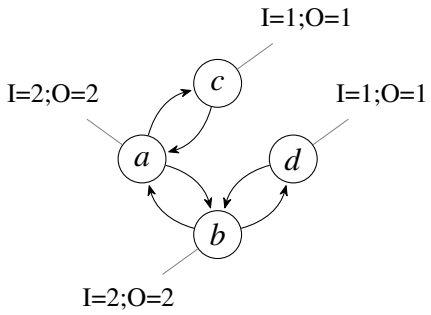


Figure 1.4: A balanced graph

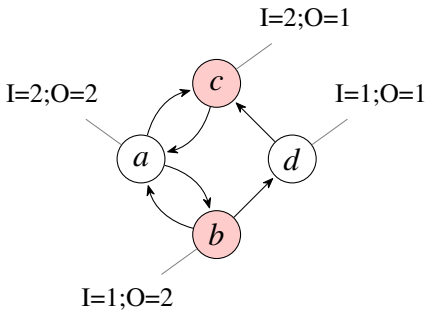


Figure 1.5: A graph that is not balanced

1.2.1 Oriented paths

Like with undirected graphs we will introduce the notion of a path, but now for directed graphs. To distinguish paths in undirected and directed graphs we will call them *oriented paths*.

Definition. Let $a_1, a_2, \ldots, a_n$, $n \geq 1$ be arcs in a directed graph. $(a_1, a_2, \ldots, a_n)$ is an *oriented path* of length n from vertices v_1 to v_2 if

1. $\text{init}(a_1) = v_1$
2. $\text{fin}(a_n) = v_2$
3. $\text{fin}(e_i) = \text{init}(e_{i+1})$ for $1 \leq i \leq n$

As a convenience, we can define a path's *length* as the number of arcs in it. So if $P = (a_1, \ldots, a_n)$ then $\text{length}(P) = n$.

It is important to notice that we switched from defining a path on the vertices it visits, like we did for undirected graphs, to listing the arcs we take. Unlike with undirected graphs, in the case where multiple arcs are between the same two vertices going in the same direction, we do in fact care about which of these arcs we use in our path. This may seem an arbitrary distinction we will, however, see later why we take this route.

With this distinction made clear, we can see that we can transfer our definitions of simple paths and cycles from undirected graphs, only changing what relates to vertices to what relates to arcs.

Definition. An oriented path $P = (a_1, a_2, \ldots, a_n)$ is *simple* if

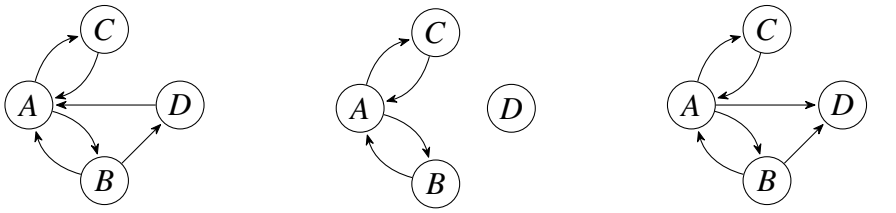
1. $\text{init}(a_1), \ldots, \text{init}(a_n)$ are distinct
2. $\text{fin}(a_1), \ldots, \text{fin}(a_n)$ are distinct

Definition. An oriented path $P = (a_1, a_2, \ldots, a_n)$ is a *cycle* or *circuit*, if it is simple and $\text{init}(a_1) = \text{fin}(a_n)$.

1.2.2 Properties of Directed Graphs

Definition. A directed graph $G = \langle V, A \rangle$ is *strongly connected*, if for any two vertices $a, b \in V, a \neq b$ there exists an oriented path from a to b.

It also makes sense to talk about directed graphs that are *weakly connected*, i.e. where the corresponding undirected graph is connected. Though in the context of directed graphs, we will refer to connected directed graphs as those that are strongly connected.



A strongly connected directed graph

A graph that is not connected

A weakly connected graph

Figure 1.6: Examples of connected and unconnected digraphs

Definition. A *root* is a vertex r in a directed graph $G = \langle V, A \rangle$ such that for any vertex $v \in V, v \neq r$ there exists an oriented path from v to r

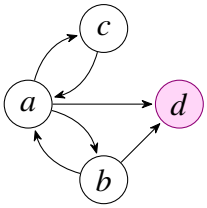


Figure 1.7: A graph with root d

2 Oriented Trees

We have examined undirected graphs that exhibit tree-like structure and formalized the notion to be free trees. For directed graphs a similar formalization can be done. As we can no longer ignore the direction of edges however, we will have to restrict our definition to make sure that our graphs still behave in a tree-like manner.

Definition. An *oriented tree* is a directed graph G with a vertex r such that

(a) each vertex v in G, $v \neq r$ is the initial vertex of exactly one arc, denoted by $\mathrm{e}[v]$

(b) no arc in G has r as its initial vertex

(c) r is a root in G

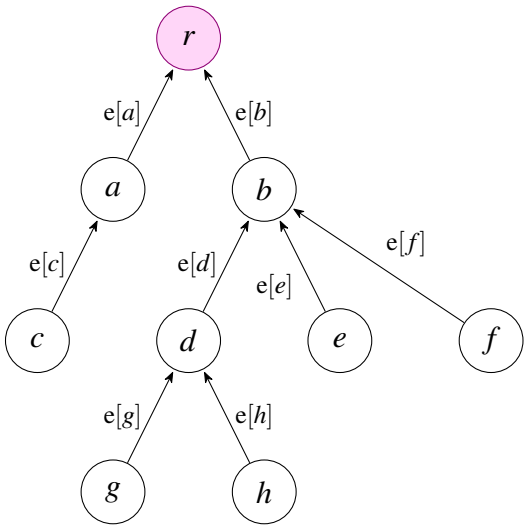


Figure 2.1: An oriented tree with root r

If we now a directed graph to by free of cycles we need only show properties (a) and (b) to prove it is an oriented tree, as shown below.

Lemma C. A directed graph that possesses no oriented cycles and satisfies properties (a) and (b) of the definition of an oriented tree is an oriented tree.

Proof. Fix a directed graph G and vertex r with no cycles that satisfies properties (a) and (b) of an oriented tree. We will only have to show that G also satisfies property (c), that is that r is a root in G.

Consider choosing a vertex in G that is not r and traversing G by forming an oriented path. Since every vertex is the initial vertex of exactly one arc, we can always continue our path, except when we encounter r. As r is not the initial vertex of any arc, we cannot continue. Notice that since G possesses no cycle, every path we traverse must be finite, so there must a last vertex visited in our path. This shows that for every vertex v in G we can start at, the path starting from v that is of longest possible length must end in r. If it did, in fact, end in another vertex w, $w \neq r$ then we could extend our path by the arc that w is the initial vertex of.

Thus, we have shown that for every vertex in G there must exist an oriented path from that vertex to r. This means that G satisfies property (c) along with properties (a) and (b), making it an oriented tree. □

This property can be useful in showing graphs to be oriented trees if we can assume them to be free of cycles, as we will see later.

2.1 Oriented Trees and Free Trees

We can now make a comparison between oriented trees and free trees. As oriented trees are just directed graphs with additional properties, we can likewise get the corresponding undirected graph by ignoring the direction of arcs. By Lemma C 2 we can see that an oriented tree is connected and possesses no cycles. As such, the corresponding undirected graph is also cycle-free and connected, thus making it a free tree.

For example, consider the oriented tree given in Figure 2.2 and its corresponding free tree. Though the representation of oriented trees starting with the root vertex and growing down "makes sense", free trees do not define the concept of a root vertex, so it is only represented that way to emphasise the correlation to the oriented tree.

It is even possible to define an inverse mapping from a free tree to an oriented tree. The intuition for this is to "pick up" the tree by a chosen root vertex and assigning an "upward" direction to all edges. Notice that this means that, unlike with our mapping from oriented trees to free trees, one free tree corresponds to many oriented trees. Depending on which vertex of the free tree we pick as the root, we get different oriented trees. For example, both oriented trees in Figure2.3 correspond to the same free tree given in Figure2.2. They are however not equal as the root vertices were picked differently (a and d) which results in the assignment of different direction for the edges.

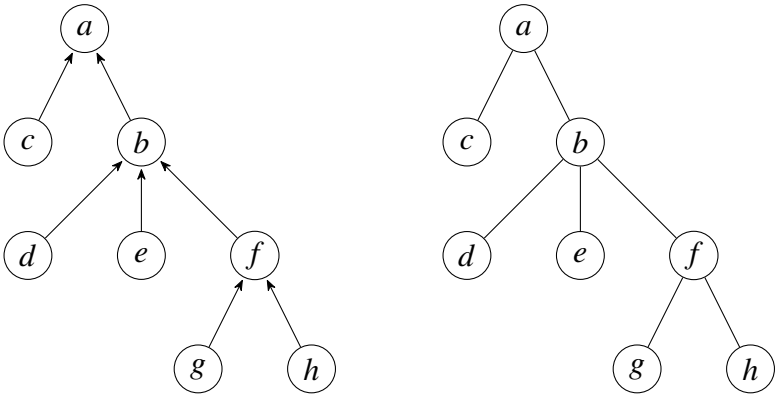


Figure 2.2: An oriented tree and corresponding free tree

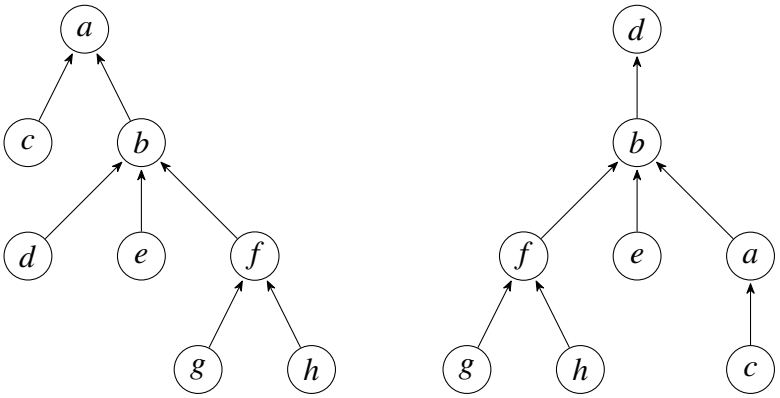


Figure 2.3: Two oriented trees

2.2 Eulerian Circuits

We can take a look at a special case of path, first observed by Leonhard Euler in 1736 [3], though in his case only for undirected graphs. In a directed graph, we want to find a path that traverses over all arcs and starts at the same vertex it ends in. In the case of Euler, he was interested in finding a path that traverses over all bridges spanning the river Pregel in Königsberg and ends up on the same side of the river or island it started at. Such a path would be called an eulerian circuit. Consequently, we can define eulerian circuits on directed graphs. We will later see an interesting connection between eulerian circuits in directed graphs and oriented trees.

Definition. An *eulerian circuit* in a directed graph G is an oriented path $P = (a_1, a_2, \ldots, a_n)$ such that

- each arc in G occurs exactly once in P
- $\text{init}(a_1) = \text{fin}(a_n)$

2.3 Existence of Eulerian Circuits

Theorem G. A finite directed graph possesses an eulerian circuit iff it is connected and balanced.

Proof. We fix a balanced and connected directed graph G and an oriented path $P = (a_1, \ldots, a_n)$ in G that is longest possible length and uses no arc twice. Note, that we make no provisions as to how such a path may be obtained, it is however reasonable to assume that such a path must exist. Let us now show that P is an eulerian circuit.

First, let us examine that P is a cycle. If vertex $v = \text{fin}(a_n)$ and $k = \text{out-degree}(v)$ we can see that all k arcs starting at v must be used in P. If an arc a with $\text{init}(a) = v$ that is not in P existed then we could append a to P, resulting in $P' = (a_1, \ldots, a_n, a)$ with $\text{length}(P') = \text{length}(P) + 1$, where no arc in P' would be used twice. But this contradicts our assumption that P is of longest possible length, so a can not exist.

Furthermore, we observe that for any arc a_i, $1 < i < n$ in P with $\text{init}(a_i) = v$ it must hold that $\text{fin}(a_{i-1}) = v$. We can thus form pairings of arcs (a_i, a_{i-1}) as such:

$\text{init} = v$	$\text{fin} = v$
	a_n
a_i	a_{i-1}
$\vdots$	$\vdots$

Notice that the left column for the arc a_n is left empty. What if we wanted to find a pairing (a, a_n) such that $\text{init}(a) = v$? In fact, such an arc must exist! As G is balanced, it must hold that $\text{in-degree}(v) = \text{out-degree}(v)$. We can see that if we consider all pairings (a_i, a_{i-1}) this must hold. But if we could not find a such a pairing for a_n then $\text{in-degree}(v) = \text{out-degree}(v) + 1$ which would contradict our assumption that G is balanced. As our only choice is to pick arc a_1 it follows that $\text{init}(a_1) = v$.

$\text{init} = v$	$\text{fin} = v$
a_1	a_n
a_i	a_{i-1}
$\vdots$	$\vdots$

So as $\text{init}(a_1) = \text{fin}(a_n)$, we can conclude that P is a cycle. Also let us observe that since P is a cycle, we can form cyclic permutations of P. For instance, the path $(a_{i+1 \bmod n}, \ldots, a_n, a_1, a_i)$ can be formed for any a_i in P.

Now let us show that all arcs in G are used in P by contradiction. Assume that there exists an arc a in G that is not used in P. If $\text{init}(a)$ occurred in P, that is $\text{init}(a) = \text{fin}(a_i)$ for some a_i in P, then the cyclic permutation $P' = (a_{i+1 \bmod n}, \ldots, a_i)$ must exist. Since $\text{fin}(a_i) = \text{init}(a)$ the path $(a_{i+1 \bmod n}, \ldots, a_i, a)$ is also valid and of

length $1 + \text{length}(P)$. This contradicts our assumption that P is of longest possible length, so $\text{init}(a)$ cannot occur in P. We can make an analogous argument for the final vertex of a. It follows a has neither initial nor final vertex in common with any arc in P. Evidently, this contradicts our assumption that G is strongly connected, so a cannot exist i.e. every arc in G occurs in P.

Since we assumed that every arc in P occurs only once and have shown every arc in G occurs in P and that P is a cycle, we can conclude that P is an eulerian circuit. □

2.4 Eulerian Circuits and Oriented Trees

Now we can finally take a look at the promised connection between oriented trees and eulerian circuits. If we have an arc a_1 in a directed Graph G and some arbitrary path in G that is a circuit. Traversing the circuit, we can see that for every vertex v we visit, though we may visit it multiple times, there will always be a last visit. We call the arc with which we exit a vertex after the last visit to it the *last exit* of v.

Lemma E. Let $G = \langle V, A \rangle$ be a directed graph that is strongly connected. Let $P = (a_1, \ldots, a_n)$ be an eulerian circuit in G where vertex $r = \text{init}(a_1) = \text{fin}(a_n)$. For every vertex v, $v \neq r$ let $\text{e}[v]$ be the last arc a in P with $\text{init}(a) = v$. That is,

$$\text{e}[v] = a_j \quad \text{if } \text{init}(a_j) = v \quad \text{and } \text{init}(a_k) \neq v \quad \text{for } j < k \leq n$$

Then V along with the arcs $\text{e}[V]$ form an oriented tree with root r.

Proof. Let T be the directed graph with vertices V and arcs $\text{e}[V]$. Let r be the last vertex in P, that is $r = \text{fin}(a_n)$. It is easy to show that T satisfies properties (a) and (b) of an oriented tree. For once, we can see that for every vertex v in T, $v \neq r$ there exists an arc with v as its initial vertex, namely $\text{e}[v]$. Furthermore, we can see that r is not the initial vertex of any arc in T, since we explicitly exclude a_1.

Now let us show that T also satisfies property (c). From Lemma C we know that we need only show that T contains no oriented cycles. Consider vertices v and w in T with $\text{fin}(\text{e}[v]) = w = \text{init}(\text{e}[w])$. If $\text{e}[v] = a_i$ and $\text{e}[w] = a_j$ are arcs in P, so i and j are the indices in P, then it must be that $i < j$. If T contained a cycle, then the indices of arcs would increase indefinitely. But since n is the maximal value of indices of all arcs, this cannot be so T does not contain any cycles.

We can now apply Lemma C to obtain a proof that T is an oriented tree. □

This shows us that there is a "striking relationship", as Aardenne-Ehrenfest and de Bruijn [1] put it, between oriented trees and eulerian circuits. We can use this relationship to find a construction method for eulerian circuits, as we will now see.

2.5 Construction of Eulerian Circuits

In Theorem G we examined that connected and balanced directed graphs possess an eulerian circuit. Our proof for this is sadly inherently non-constructive. Notice how we assume to be given an oriented path of longest possible length that uses no arc twice. It is safe to assume that such a path must exist, however, we make no provisions as to how to obtain such a path.

We will now look at a construction method for eulerian circuits that does give us an instance of that path and verifies that it is in fact an eulerian circuit. We will still assume the graph which we reason about to be connected and balanced to guarantee the existence of an eulerian circuit. Then we can use the relationship between eulerian circuits and oriented trees from Lemma E to our advantage.

Theorem D. Let G be a connected and balanced directed graph and T be an oriented tree of all vertices and some arcs of G with a root vertex R. Let a_1 be an arc of G with $\text{init}(a_1) = R$. Then $P = (a_1, a_2, \ldots, a_m)$ is an eulerian circuit if

(a) no arc is used twice in P

(b) if for some vertex V in G, $a_j = \text{e}[V]$ and there is an arc a with $\text{init}(a) = V$ then $a = a_k$ with $1 \leq k \leq j$

(c) if for some arc a in G, $\text{init}(a) = \text{fin}(a_m)$ then $a = a_k$ with $1 \leq k \leq m$

Proof. First, we can employ a similar argument to the one used in the proof of Theorem G to show that P must be a cycle. Let v be the final vertex of a_m, $v = \text{fin}(a_m)$. Assume there exists an arc a with $\text{init}(a) = v$ that does not occur in P. This contradicts our assumption from (c), since if a existed, then P could still be extended with a so in our case we would have terminated prematurely.

Now for every arc a_i in P, so $1 < i < m$ we have $v = \text{init}(a_i)$. Consequently, we also have $\text{fin}(a_{i+1}) = v$. As we have already shown in Theorem G, G being balanced means that we now know that $\text{init}(a_1) = \text{fin}(a_m)$, so P is a cycle.

Now, let us show that all arcs in G are used in P. Assume a to be an arc in G that is not used in P and $v = \text{fin}(a)$. Then we also have an arc b not in P with $v = \text{init}(b)$ since G is balanced. If $v \neq r$ then we can infer from (b) that $\text{e}[v]$ is not in P because if b is not in P then $\text{e}[v]$ cannot be in P as (b) prohibits us from choosing an arc T before any other choice is exhausted.

Now if we apply this argument to $a = \text{e}[v]$ we can extend it to the oriented path from v to r, which we know to exist. For every arc from c to d in this path we can reason that there must exist an arc with initial vertex d. Eventually we will find an arc initial vertex v. But we know this contradicts (c) as is shown above, so all arcs in G must occur in P.

Since P contains all arcs in G and is a cycle, it is an eulerian circuit. □

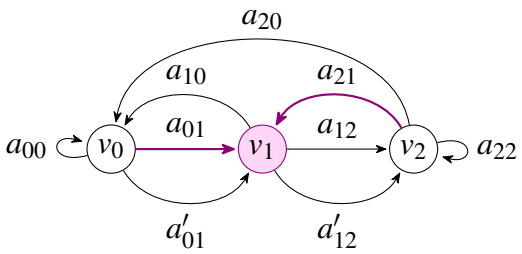


Figure 2.4: A directed graph with oriented sub-tree

As an example, consider the directed graph G in Figure 2.4 that possesses a subgraph T that is an oriented tree with root v_1 and arcs $a_{01} = \mathrm{e}[v_0], a_{21} = \mathrm{e}[v_2]$. We can easily convince ourselves that G is balanced, as in-degree(v_i) = out-degree(v_i) for $i \in \{0,1,2\}$. We can also see that all vertices in G are also part of T. Therefore, the construction method in Theorem D can be applied to find an eulerian circuit P in G starting from T's root vertex v_1.

Table 2.1 shows in detail how the construction method in Theorem G works. For each row P represents the current state of our traversal. C_G and C_T show choices for the next arc to append to P, that is if P ends with arch a with $\mathrm{fin}(a) = v$ then all arcs b in G with $\mathrm{init}(b) = v = \mathrm{fin}(a)$ are valid choices. We do however make the distinction between C_T the arcs that are part of the oriented tree T, and C_G, the arcs that are in G but *not* part of T. If in a row the set C_G contains more than one arc, we have a choice to select one of the valid arcs in C_G. If C_G is empty, we have to choose the singleton element of C_T. Theorem D shows how, in this case, this arc is guaranteed to exist. If both C_G and C_T are empty, we terminate.

P	C_G	C_T	Choice
(a_{12})	$\{a_{20}, a_{22}\}$	$\{a_{21}\}$	C_G
(a_{12}, a_{20})	$\{a_{00}, a'_{01}\}$	$\{a_{01}\}$	C_G
(a_{12}, a_{20}, a_{00})	$\{a'_{01}\}$	$\{a_{01}\}$	C_G
$(a_{12}, a_{20}, a_{00}, a'_{01})$	$\{a_{10}, a'_{12}\}$	$\emptyset$	C_G
$(a_{12}, a_{20}, a_{00}, a'_{01}, a'_{12})$	$\{a_{22}\}$	$\{a_{21}\}$	C_G
$(a_{12}, a_{20}, a_{00}, a'_{01}, a'_{12}, a_{22})$	$\emptyset$	$\{a_{21}\}$	C_T
$(a_{12}, a_{20}, a_{00}, a'_{01}, a'_{12}, a_{22}, a_{21})$	$\{a_{10}\}$	$\emptyset$	C_G
$(a_{12}, a_{20}, a_{00}, a'_{01}, a'_{12}, a_{22}, a_{21}, a_{10})$	$\emptyset$	$\{a_{01}\}$	C_T
$(a_{12}, a_{20}, a_{00}, a'_{01}, a'_{12}, a_{22}, a_{21}, a_{10}, a_{01})$	$\emptyset$	$\emptyset$	$\lrcorner$

Table 2.1: Construction of an eulerian circuit

Bibliography

[1] T. Aardenne-Ehrenfest, van and N. G. Bruijn, de. Circuits and trees in oriented linear graphs. *Simon Stevin: Wis- en Natuurkundig Tijdschrift*, 28:203–217, 1951.

[2] Béla Bollobás. *Modern Graph Theory*. Springer New York, New York, NY, 1998.

[3] Donald E. Knuth. *The Art of Computer Programming, Volume I: Fundamental Algorithms*. Addison-Wesley, Reading, Massachusetts, 1997.

[4] Md. Saidur Rahman. *Basic Graph Theory*. Springer Cham, 2017.