Chapter 6

Optimal Traversals: Lec 7

6.1 Eulerian Trails and Tours
6.2 DeBruijn Sequences and Postman Problems
6.3 Hamiltonian Paths and Cycles
6.4 Gray Codes and Traveling Salesman Problems
6.1 EULERIAN TRAILS AND TOURS

REVIEW FROM §1.5:

- An eulerian trail in a graph is a trail that contains every edge of that graph.
- An eulerian tour is a closed eulerian trail.
- An eulerian graph is a graph that has an eulerian tour.

Königsberg Bridges Problem

![Fig 6.1.1 The seven bridges of Königsberg.](image-url)
Euler’s model for the problem was a graph with four vertices, one for each of the land masses \( u, v, w, \) and \( x, \) and seven edges, one for each of the bridges, as shown in Fig 6.1.2.

![Graph representation of Konigsberg.](image)

**Fig 6.1.2** Graph representation of Königsberg.

Since the graph has vertices of odd degree, there is no eulerian tour, by the Eulerian-Graph Characterization (§4.5).

**Theorem from §4.5 [Eulerian-Graph Characterization]**

The following statements are equivalent for a conn graph \( G. \)

1. \( G \) is eulerian.
2. The degree of every vertex in \( G \) is even.
3. \( E_G \) is the union of the edge-sets of a set of edge-disjoint cycles in \( G. \)
A constructive proof of Hierholzer’s Theorem:

**Algo 6.1.1: Eulerian Tour**

*Input:* a conn graph $G$ whose vertices all have even degree.  
*Output:* an eulerian tour $T$.

Start at any vertex $v$, and construct a closed trail $T$.  
While there are edges of $G$ not already in trail $T$  
  Choose a vtx $w$ in $T$ that is endpt of an unused edge.  
  Starting at vtx $w$, construct a closed trail $D$ of unused edges.  
  Enlarge trail $T$ by splicing trail $D$ into $T$ at vtx $w$.  
Return $T$.  

© 2010 by Jonathan L. Gross
Example 6.1.1: The key step in Algo 6.1.1 is enlarging a closed trail by combining it with a second closed trail — the detour. To illustrate, consider the closed trails

\[ T = \langle t_1, \ t_2, \ t_3, \ t_4 \rangle \]

and

\[ D = \langle d_1, \ d_2, \ d_3 \rangle \]

in the graph shown in Figure 6.1.3. The closed trail that results when detour \( D \) is spliced into trail \( T \) at vertex \( w \) is given by

\[ T' = \langle t_1, \ t_2, \ d_1, \ d_2, \ d_3, \ t_3, \ t_4 \rangle \]

At the next iteration, the trail \( \langle e_1, e_2, e_3 \rangle \) is spliced into trail \( T' \), resulting in an eulerian tour of the entire graph.
Open Eulerian Trails

Certain applications require open eulerian trails, which do not end at the starting vertex. The following characterization of graphs having an open eulerian trail is obtained directly from the characterization of eulerian graphs.

Thm 6.1.1. A connected graph $G$ has an open eulerian trail iff it has exactly two vertices of odd degree. Furthermore, the initial and final vertices of an open eulerian trail must be the two vertices of odd degree.

Pf: $(\Rightarrow)$ Suppose that $\langle x, e_1, v_1, e_2, \ldots, e_m, y \rangle$ is an open eulerian trail in $G$. Adding a new edge $e$ joining vertices $x$ and $y$ creates a new graph $G^* = G + e$ with an eulerian tour $\langle x, e_1, v_1, e_2, \ldots, e_m, y, e, x \rangle$. By the eulerian-graph characterization, the degree in the eulerian graph $G^*$ of every vertex must be even, and thus, the degree in graph $G$ of every vertex except $x$ and $y$ is even.

$(\Leftarrow)$ Suppose that $x$ and $y$ are the only two vertices of graph $G$ with odd degree. If $e$ is a new edge joining $x$ and $y$, then all the vertices of the resulting graph $G^*$ have even degree. It follows from the eulerian-graph characterization that graph $G^*$ has an eulerian tour $T$. Hence, the trail $T - e$ obtained by deleting edge $e$ from tour $T$ is an open eulerian trail of $G^* - e = G$. \[\triangleleft\]
Eulerian Trails in Digraphs

The use of the term eulerian is identical for digraphs, except that all trails are directed. The characterizations of graphs that have eulerian tours or open eulerian trails apply to digraphs as well. Their proofs are essentially the same as the ones for undirected graphs and are left as exercises.

Thm 6.1.2 [Eulerian-Digraph Characterization]. The following statements are equivalent for a connected digraph $D$.

1. Digraph $D$ is eulerian.

2. For every vertex $v$ in $D$, $\text{indegree}(v) = \text{outdegree}(v)$.

3. The edge-set $E_D$ is the union of the edge-sets of a set of edge-disjoint directed cycles in digraph $D$. ♦ (Exercises)

Thm 6.1.3. A connected digraph $D$ has an open eulerian trail from vertex $x$ to vertex $y$ iff

(1) $\text{outdegree}(x) = \text{indegree}(x) + 1$

(2) $\text{indegree}(y) = \text{outdegree}(y) + 1$

and

(3) all vertices of $D$ except $x$ and $y$ have equal indegree and outdegree. ♦ (Exercises)
DEF: A bitstring of length $2^n$ is called a \((2, n)\)-DeBruijn sequence if each of the $2^n$ possible bitstrings of length $n$ occurs \textit{exactly once} as a substring, where wraparound is allowed.

Example 6.2.1: These four bitstrings are \((2, n)\)-DeBruijn sequences for the cases $n = 1, 2, 3, 4$, respectively.

$$
\begin{align*}
01 & \quad 0110 & \quad 0110100 & \quad 000100110101111
\end{align*}
$$
DEF: The \((2, n)\)-DeBruijn digraph \(D_{2,n}\) consists of \(2^{n-1}\) vertices, labeled by the bitstrings of length \(n - 1\), and \(2^n\) arcs, labeled by the bitstrings of length \(n\). The arc from vertex \(b_1b_2\cdots b_{n-1}\) to vertex \(b_2\cdots b_{n-1}b_n\) is labeled \(b_1b_2\cdots b_n\).

Example 6.2.2: Each arc in Fig 6.2.1 is labeled with only the leftmost bit of its full label. E.g., the actual label for the arc directed from 000 to 001 is 0001.

Fig 6.2.2 The \((2, 4)\)-DeBruijn digraph \(D_{2,4}\).

KEY PROPERTY: The sequence of bits encountered on a walk of length \(n\) from any vertex \(v\) in a DeBruijn graph \(D_{2,n}\) is precisely the label of vertex \(v\).
Prop 6.2.1. The \((2, n)\)-deBruijn digraph \(D_{2,n}\) is eulerian.

Pf: The deBruijn graph \(D_{2,n}\) is strongly connected, since if \(a_1a_2\cdots a_{n-1}\) and \(b_1b_2\cdots b_{n-1}\) are any two vertices of \(D_{2,n}\), then the vertex sequence:

\[
a_1a_2\cdots a_{n-1}; a_2\cdots a_{n-1}b_1; a_3\cdots a_{n-1}b_1b_2; \ldots; b_1b_2\cdots b_{n-1}
\]
defines a directed trail from \(a_1a_2\cdots a_{n-1}\) to \(b_1b_2\cdots b_{n-1}\).

Moreover, for every vertex \(b_1b_2\cdots b_{n-1}\) of \(D_{2,n}\), the only outgoing arcs from \(b_1b_2\cdots b_{n-1}\) are \(b_1b_2\cdots b_{n-1}0\) and \(b_1b_2\cdots b_{n-1}1\), and the only incoming arcs to \(b_1b_2\cdots b_{n-1}\) are \(0b_1b_2\cdots b_{n-1}\) and \(1b_1b_2\cdots b_{n-1}\). Thus,

\[
\text{indegree}(b_1b_2\cdots b_{n-1}) = \text{outdegree}(b_1b_2\cdots b_{n-1}) = 2
\]

which implies, by Theorem 6.1.2, that \(D_{2,n}\) is eulerian. \(\diamondsuit\)
**Algo 6.2.1:** Constructing a \( (2, n) \)-deBruijn Sequence

*Input:* a positive integer \( n \).
*Output:* a \( (2, n) \)-deBruijn sequence \( S \).

Construct the \( (2, n) \)-deBruijn digraph \( D_{2,n} \).
Choose a vertex \( v \).
Construct a directed eulerian tour \( T \) of \( D_{2,n} \) starting at \( v \).
Initialize sequence \( S \) as the empty sequence \( \langle \rangle \).
For each arc \( e \) in tour \( T \) (taken in order of the tour sequence)
  Append the single-bit label of arc \( e \) to the right of seq \( S \).

---

**Example 6.2.3:** Constructing a \( (2, 4) \)-deBruijn seq can begin
with the \( (2, 4) \)-deBruijn digraph shown in Fig 6.2.2. We see
that the following seq of vertices and arcs is an eulerian tour.

\[
000 \rightarrow 000 \rightarrow 001 \rightarrow 010 \rightarrow 100 \rightarrow 001 \rightarrow 011 \rightarrow 110 \rightarrow 101
\]
\[
\quad \rightarrow 101 \rightarrow 011 \rightarrow 111 \rightarrow 111 \rightarrow 110 \rightarrow 100 \rightarrow 000
\]

The \( (2, 4) \)-deB seq 0000100110101111 is the seq of arc labels.
The following proposition reiterates what we called the KEY PROPERTY, which establishes the correctness of the construction of Algorithm 6.2.1.

Prop 6.2.2. The sequence of single-bit arc labels of any directed trail of length $k$, $1 \leq k \leq n$, in the $(2,n)$-deBruijn digraph, is precisely the string of the leftmost $k$ bits of the full label of the initial arc of that trail.

Pf: The definition of $D_{2,n}$ implies the assertion for $k = 1$. Assume for some $k$, $1 \leq k \leq n - 1$, that the assertion is true for any directed trail of length $k$, and let $\langle v_0, v_1, \ldots, v_{k+1} \rangle$ be any directed trail of length $k+1$. By the ind hyp, the single-bit arc labels associated with the subtrail $\langle v_1, v_2, \ldots, v_{k+1} \rangle$ match the first $k$ bits of the full label of the arc from $v_1$ to $v_2$. By the definition of the deBruijn digraph, these $k$ bits are the first $k$ bits of the label on vertex $v_1$, which means that they are also bits 2 through $k + 1$ of the full label of the arc from $v_0$ to $v_1$. But the single-bit label on that arc is the leftmost bit of its full label, which completes the induction step. ◊
Example 6.2.4: To illustrate Prop 6.2.2 for the digraph $D_{2,4}$ in Fig 6.2.2, consider the directed subtrail given by

$$100 \xrightarrow{1} 001 \xrightarrow{0} 011 \xrightarrow{0} 110 \xrightarrow{1} 101$$

The bitstring 1001 formed from the single-bit arc labels is the full label for the arc from 100 to 001.

By the construction of $D_{2,4}$, each arc is labeled with a different bitstring of length 4. Thus, since the eulerian tour uses each arc exactly once, Proposition 6.2.2 implies that the resulting sequence contains all possible bitstrings of length 4 and is, therefore, a $(2,4)$-deBruijn sequence.
Guan’s Postman Problem

**DEF:** A *postman tour* in a graph $G$ is a closed walk that uses each edge of $G$ at least once.

**DEF:** In a weighted graph, an *optimal postman tour* is a postman tour whose total edge-weight is a minimum.

Unless each vertex of the graph has even degree, some edges must be retraced (or *deadheaded*). Thus, the goal is to find a postman tour whose deadheaded edges have minimum total edge-weight.
DEF FROM §4.7 A matching in a graph $G$ is a subset $M$ of $E_G$ such that no two edges in $M$ have an endpoint in common.

DEF: A perfect matching in a graph is a matching in which every vertex is an endpoint of one of the edges.

The idea behind the Edmonds–Johnson algorithm is to retrace the edges on certain shortest paths between odd-degree vertices, where the edge-weights are regarded as distances. If the edges of a path between two odd-degree vertices are duplicated, then the degrees of those two vertices become even and the parity of the degree of each internal vertex on the path remains the same.

Fig 6.2.3 shows the weighted complete graph on the odd-deg vertices in $G$. E.g., the shortest path between odd-deg vertices $b$ and $d$ has length 8. The min-wt perfect matching is in bold.

Fig 6.2.3 Min-wt perfect matching of complete graph $K_4$. 

© 2010 by Jonathan L. Gross
Suppose that a graph, e.g. $K_{2k}$, has $2k$ odd vertices. Then the number of possible matchings is

\[
\frac{(2k)!}{k!2^k} = (2k - 1)(2k - 3)\cdots 1
\]

This number grows super-exponentially as $k$ increases. Accordingly, Edmonds and Johnson devised what amounts to a hill-climbing method to ascend the space of pairings.

**Example 6.2.5:** Apply Algo 6.2.1 to the weighted graph $G$.

![Graph G](image)

**Fig 6.2.4** A weighted graph for the Postman Problem.

Since there are only 4 odd vertices, there are only 3 possible pairings. Accordingly, we can solve this easily by hand.

- $bd\ f h\ 8 + 9 = 17$
- $bf\ dh\ 7 + 12 = 19$
- $bh\ df\ 8 + 11 = 19$

© 2010 by Jonathan L. Gross
Each edge in the perfect matching obtained in Algo 6.2.2 represents a path in $G$. The edges on this path are the ones that are duplicated to obtain the eulerian graph $G^*$, as in Fig 6.2.5.

\[ \text{Fig 6.2.5 Duplicating edges to obtain an eulerian graph } G^*. \]

Finally, an eulerian tour for graph $G^*$ is an optimal postman tour for $G$. One such tour is given by the vertex sequence

\[ \langle a, b, c, f, e, b, e, d, e, h, i, f, i, h, g, d, a \rangle \]

All but one of the steps of the Postman Algorithm involve algorithms that have already been discussed. The exception is the step that finds an optimal perfect matching of a complete graph. This problem can be transformed into the problem of finding an optimal perfect matching of an associated complete bipartite graph. Methods developed in Chapter 13 lead to a polynomial-time algorithm for this type of problem.
6.3 HAMILTONIAN PATHS AND CYCLES

DEF: A *hamiltonian path (cycle)* of a graph is a path (cycle) that contains all the vertices. For digraphs, the hamiltonian path or cycle is directed.

DEF: A *hamiltonian graph* is a graph that has a hamiltonian cycle.

Remark: Adding or deleting self-loops or extra adjacencies between two vertices does not change a graph from hamiltonian to non-hamiltonian.

One of several aspects of the Icosian Game was to find a hamiltonian cycle on the dodecahedral graph, depicted in Figure 6.3.1 below.

![Dodecahedral graph for the Icosian Game.](image)

© 2010 by Jonathan L. Gross
Showing That a Graph Is Not Hamiltonian

The following rules are based on the simple observation that any hamiltonian cycle must contain exactly two edges incident on each vertex. The strategy for applying these rules is to begin a construction of a hamiltonian cycle and show at some point during the construction that it is impossible to proceed any further. The path that this kind of argument takes depends on the particular graph. There are three rules:

1. If a vertex $v$ has degree 2, then both of its incident edges must be part of any hamiltonian cycle.

2. During the construction of a hamiltonian cycle, no cycle can be formed until all the vertices have been visited.

3. If during the construction of a hamiltonian cycle two of the edges incident on a vertex $v$ are shown to be required, then all other incident edges can be deleted.
Example 6.3.1: The graph in Fig 6.3.2 is not hamiltonian.

![Graph 1](image1)

**Fig 6.3.2** A non-hamiltonian graph.

Example 6.3.2: The graph in Fig 6.3.3 is not hamiltonian.

![Graph 2](image2)

**Fig 6.3.3** Another non-hamiltonian graph.
Sufficient Conditions to be Hamiltonian

The next two results lend precision to the notion that the more edges a simple graph has, the more likely it is hamiltonian. The chronologically earlier of the two results, first proved by G.A. Dirac in 1952 ([Di52]), can now be deduced from the more recent result proved by O. Ore ([Or60]).

**Thm 6.3.1 [Ore, 1960].** Let $G$ be a simple $n$-vertex graph, where $n \geq 3$, such that

$$\deg(x) + \deg(y) \geq n$$

for each pair of non-adjacent vertices $x$ and $y$. Then $G$ is hamiltonian.

**Pf:** By way of contradiction, assume that the theorem is false, and let $G$ be a maximal counterexample. That is,

- $G$ is non-hamiltonian
- $G$ satisfies the conditions of the theorem, and
- the addition of any edge joining two non-adjacent vertices of $G$ results in a hamiltonian graph.
Let $x$ and $y$ be two non-adjacent vertices of $G$ ($G$ is not complete, since $n \geq 3$). To reach a contradiction, it suffices to show that $\deg(x) + \deg(y) \leq n - 1$.

Since graph $G + xy$ contains a hamiltonian cycle, $G$ contains a hamiltonian path whose endpoints are $x$ and $y$. Let $\langle x = v_1, v_2, \ldots, v_n = y \rangle$ be such a path (see Fig 6.3.4).

![Fig 6.3.4 A hamiltonian path in $G$.](image)

For each $i = 2, \ldots, n - 1$, at least one of the pairs $v_1, v_{i+1}$ and $v_i, v_n$ is non-adjacent, since otherwise,

$\langle v_1, v_2, \ldots, v_i, v_n, v_{n-1}, \ldots, v_{i+1}, v_1 \rangle$

would be a hamiltonian cycle in $G$ (see Fig 6.3.5).

![Fig 6.3.5](image)

This means that if $(a_{i,j})$ is the adj matrix for $G$, then $a_{1,i+1} + a_{i,n} \leq 1$, for $i = 2, \ldots, n - 2$.

© 2010 by Jonathan L. Gross
This means that if \((a_{i,j})\) is the adj matrix for \(G\), then
\[
    a_{1,i+1} + a_{i,n} \leq 1, \text{ for } i = 2, \ldots, n - 2
\]

Thus,
\[
    \deg(x) + \deg(y) = \sum_{i=2}^{n-1} a_{1,i} + \sum_{i=2}^{n-1} a_{i,n}
\]
\[
    = a_{1,2} + \sum_{i=3}^{n-1} a_{1,i} + \sum_{i=2}^{n-2} a_{i,n} + a_{n-1,n}
\]
\[
    = 1 + \sum_{i=2}^{n-2} a_{1,i+1} + \sum_{i=2}^{n-2} a_{i,n} + 1
\]
\[
    = 2 + \sum_{i=2}^{n-2} (a_{1,i+1} + a_{i,n})
\]
\[
    \leq 2 + n - 3 = n - 1
\]

which establishes the desired contradiction. \(\diamondsuit\)

**Cor 6.3.2 [Dirac, 1952].** Let \(G\) be a simple \(n\)-vertex graph, where \(n \geq 3\), such that \(\deg(v) \geq \frac{n}{2}\) for each vertex \(v\). Then \(G\) is hamiltonian. \(\diamondsuit\)
6.4 GRAY CODES AND TRAV SALESMAN

DEF: A Gray code of order $n$ is an ordering of the $2^n$ length-$n$ bitstrings such that consecutive bitstrings (and the first and last bitstring) differ in precisely one bit position.

Example 6.4.1: $\langle 000, 100, 110, 010, 011, 111, 101, 001 \rangle$ is a Gray code of order 3.

REVIEW FROM §1.2: The $n$-dimensional hypercube $Q_n$ is the graph whose vertex-set is the set of length-$n$ bitstrings, such that there is an edge between two vertices if and only if they differ in exactly one bit.

A Gray code of order $n$ corresponds to a hamiltonian cycle in the hypercube graph $Q_n$.

Fig 6.4.1 A hamiltonian cycle in the hypercube $Q_3$. 

© 2010 by Jonathan L. Gross
The last result here is that $Q_n$ is hamiltonian for all $n$, thereby proving that Gray codes of all orders exist.

Fig 6.4.2 Gray code of order 3 from a Gray code of order 2.
6.5 SUPPLEMENTARY EXERCISES

6.5.6 Decide whether the join $2K_1 + K_{1,4}$ is hamiltonian.

6.5.16 Give a pair of graphs one of which is hamiltonian, the other non-hamiltonian, with the same number of vertices, such that all the vertex-deleted subgraphs of both graphs are non-hamiltonian. This would establish the non-reconstructibility of the hamiltonian property.