Elemente der Graphentheorie

Definition 1:

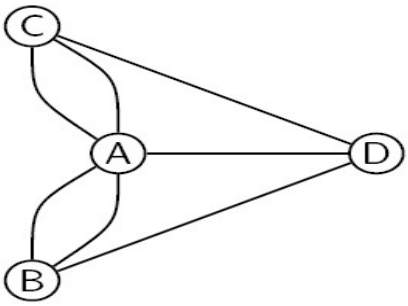
Ein **Graph** besteht aus **Knoten** (engl. vertex, vertices) und **Kanten** (engl. edge, edges). Die Menge der Knoten bezeichnen wir mit **V**, diejenige der Kanten mit **E**.

Der **Graph** heißt **zusammenhängend**, wenn er keine isolierte Knoten hat, d. h. wenn es zwischen je zwei Knoten stets eine Kante gibt.

Unter dem **Grad eines Knotens** verstehen wir bei einem ungerichteten Graph die Anzahl der Kanten, die von diesem Knoten ausgehen. (Hinweis: Bei gerichteten Graphen unterscheidet man zwischen Eingangsgrad und Ausgangsgrad.)

Zwei Knoten eines ungerichteten Graphen heißen **adjazent**, wenn sie durch eine Kante verbunden sind.

Beispiel für einen ungerichteten Graphen mit Mehrfachkanten:



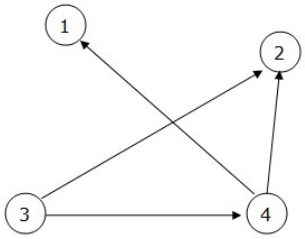
V = {A, B, C, D}

Grad(A) = 5
Grad(B) = Grad(C) = Grad(D) = 3

Definition 2:

Ein **Graph**, dessen Kanten in nur einer Richtung durchlaufen werden können, heißt gerichteter Graph.

Beispiel (Die Kanten werden als Pfeile dargestellt.):



V = {1, 2, 3, 4}

Eine von Knoten 4 nach Knoten 2 gerichtete Kante können wir als geordnetes Paar (4,2) beschreiben.
E = { (3,4), (4,2), (3,2), (4,1) }

Definition 3:

a) Unter einem **Eulerweg** verstehen wir einen Weg, der alle Kanten des ungerichteten Graphen genau einmal durchläuft.
b) Unter einem **Eulerkreis** verstehen wir einen geschlossenen Eulerweg, also einen Eulerweg, bei dem Start- und Zielknoten identisch sind.
c) Ein **Eulerscher Graph** ist ein Graph, der einen Eulerkreis besitzt.
d) Ein Graph, der einen Eulerweg, aber keinen Eulerkreis besitzt, heißt **semi-eulersch** (semi-eulerian).

Definition 4:

a) Unter einem **Hamiltonweg** verstehen wir einen Weg, der jeden Knoten des Graphen genau einmal besucht. Dabei dürfen Kanten auch mehrfach durchlaufen werden.
b) Unter einem **Hamiltonkreis** verstehen wir einen geschlossenen Hamiltonweg, also einen Hamiltonweg, bei dem Start- und Zielknoten identisch sind.
c) Ein **Hamiltonscher Graph** ist ein Graph, der einen Hamiltonkreis besitzt.
d) Ein Graph, der einen Hamiltonweg, aber keinen Hamiltonkreis besitzt, heißt **semi-hamiltonsch** (semi-hamiltonian).

Satz von Euler (Leonard Euler 1736; strenger Beweis von Carl Hierholzer 1873):

Ein ungerichteter zusammenhängender Graph G hat einen Eulerweg genau dann, wenn die Anzahl der Knoten mit ungeradem Grad den Wert 0 oder 2 hat.

Ein ungerichteter zusammenhängender Graph G hat einen Eulerkreis genau dann, wenn jeder Knoten einen geraden Grad hat.

Die Struktur eines Graphen läßt sich mittels einer **Adjazenzmatrix** abbilden, welche die Verarbeitung durch einen Algorithmus ermöglicht.

Beispiel einer 3 x 3 – Matrix:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$a_{ij} \in R$
i = Zeilenindex, j = Spaltenindex

Die Komponente a_{23} steht in der 2. Zeile und der 3. Spalte.

Sind die Knoten des Graphen **G** mit **1**, **2**, **3**, ... , **n** bezeichnet, definieren wir:
Bei ungerichteten Graphen:

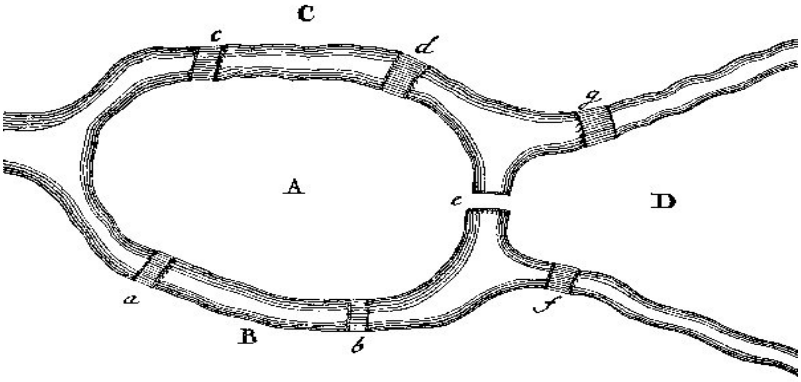
a_{ij} = Anzahl der Kanten zwischen den Knoten i und j
Falls der Graph Mehrfachkanten nicht enthält und die Kanten gewichtet sind (z. B. mit den Entfernungen der jeweiligen adjazenten Knoten i und j):
a_{ij} = Entfernung der adjazenten Knoten i und j

Bei gerichteten Graphen (ohne gleichgerichtete Mehrfachkanten):

a_{ij} = 1, falls die Kante von Knoten i in Richtung Knoten j verläuft;
a_{ij} = 0, falls die Kante von Knoten j in Richtung Knoten i verläuft oder keine Kante zwischen den Knoten i und j existiert.

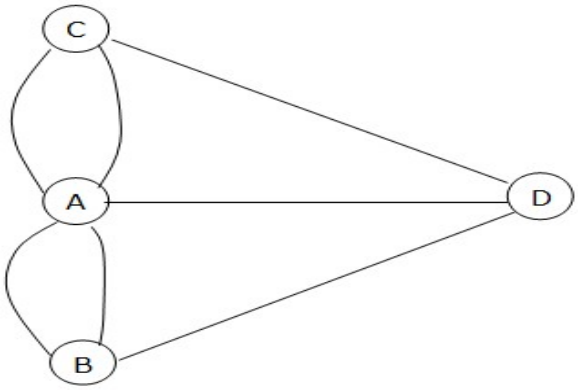
Beispiel 1

Das Königsberger Brückenproblem (1735)
Gibt es einen Weg, der jede der 7 Brücken a, b, . . . , g genau ein Mal überschreitet?



Äquivalente Formulierung:
Hat der Graph **G** mit der Knotenmenge **V** = {A, B, C, D} einen Eulerweg?

G



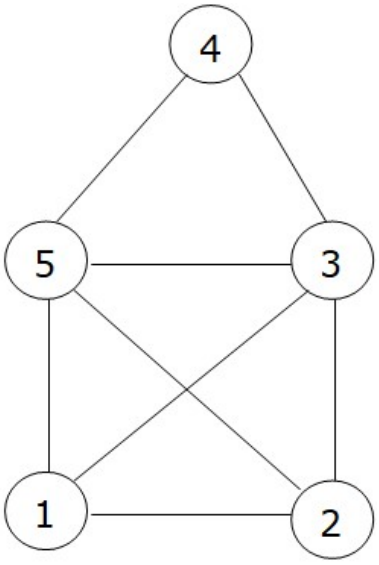
Wir ergänzen die **Adjazenzmatrix** um eine weitere, mit **#** bezeichnete Spalte, welche den Grad des jeweiligen Knotens angibt:

	A	B	C	D	#
A	0	2	2	1	5
B	2	0	0	1	3
C	2	0	0	1	3
D	1	1	1	0	3

Die Anzahl der Knoten mit ungeradem Grad beträgt 4 und hat daher weder den Wert 0 noch den Wert 2, folglich hat der Graph **G** keinen Eulerweg und erst recht keinen Eulerkreis.

Beispiel 2

Haus des Nikolaus



Adjazenzmatrix:

	1	2	3	4	5	#
1	0	1	1	0	1	3
2	1	0	1	0	1	3
3	1	1	0	1	1	4
4	0	0	1	0	1	2
5	1	1	1	1	0	4

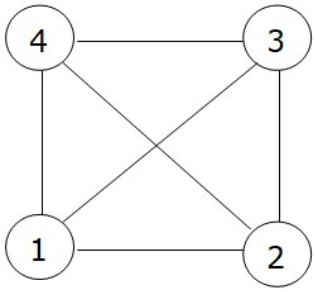
Da die Anzahl der Knoten mit ungeradem Grad den Wert 2 hat, gibt es nach dem Satz von Euler einen Eulerweg, aber keinen Eulerkreis.

Möglicher Eulerweg:

1 → 2 → 3 → 5 → 1 → 3 → 4 → 5 → 2

Beispiel 3

Quadrat mit Diagonalen



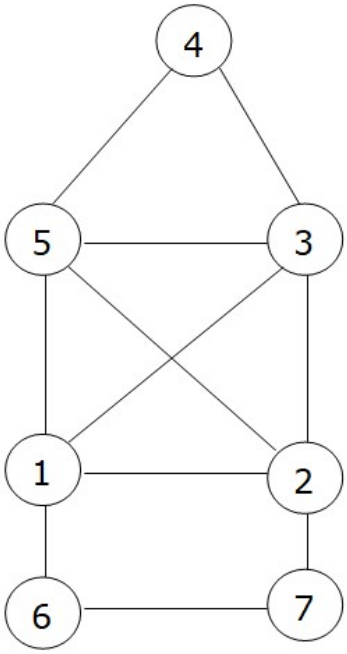
Adjazenzmatrix:

	1	2	3	4	#
1	0	1	1	1	3
2	1	0	1	1	3
3	1	1	0	1	3
4	1	1	1	0	3

Die Anzahl der Knoten mit ungeradem Grad beträgt 4 und hat daher weder den Wert 0 noch den Wert 2, folglich hat der Graph **G** keinen Eulerweg und erst recht keinen Eulerkreis.

Beispiel 4

Haus des Nikolaus mit Keller



Adjazenzmatrix:

	1	2	3	4	5	6	7	#
1	0	1	1	0	1	1	0	4
2	1	0	1	0	1	0	1	4
3	1	1	0	1	1	0	0	4
4	0	0	1	0	1	0	0	2
5	1	1	1	1	0	0	0	4
6	1	0	0	0	0	0	1	2
7	0	1	0	0	0	1	0	2

Da jeder Knoten einen geradzahligen Grad hat, gibt es nach dem Satz von Euler einen Eulerkreis.

Möglicher Eulerkreis:

6 → 7 → 2 → 3 → 5 → 1 → 2 → 5 → 4 → 3 → 1 → 6

Algorithmen zum Auffinden von Eulerwegen und –kreisen:

Wenn nach dem Satz von Euler die Existenz eines Eulerwegs oder einer Eulerkreises gesichert ist, liefern folgende Algorithmen nach Eingabe des Graphen (z. B. als Adjazenzmatrix) einen Eulerweg oder einen Eulerkreis:

Algorithmus von **Fleury**
Der Aufwand wächst mit dem Quadrat der Anzahl der Kanten, demnach hat der Algorithmus von Fleury quadratische Komplexität.

Algorithmus von **Hierholzer** (1873)
Der Aufwand wächst linear mit der Anzahl der Kanten, demnach hat der Algorithmus von Hierholzer lineare Komplexität.

Algorithmus zum Satz von Euler

Der Satz von Euler ist ein Existenzsatz, denn er stellt notwendige und hinreichende Kriterien bereit, um zu entscheiden, ob ein zusammenhängender ungerichteter Graph G einen Eulerweg oder einen Eulerkreis hat.

Als Beispiel nachfolgend ein in Pascal codierter Algorithmus, der nach Eingabe des Graphen G in Form der Adjazenzmatrix entscheidet, ob Graph G einen Eulerweg oder einen Eulerkreis oder keinen von beiden hat:

```
{
Nach Eingabe des Graphen G als Adjazenzmatrix entscheidet dieser Algorithmus,
ob G einen Eulerweg oder einen Eulerkreis oder keinen von beiden hat.
}

program euler;
uses crt;

type matrix = array[1..20,1..20] of integer;
     vektor = array[1..20] of integer;

var i, j, n, anzahl : integer;
    a : matrix;
    g : vektor;

begin
   clrscr;
   write('Anzahl (maximal 20) der Knoten: ');
   readln(n);

   {Eingabe der Adjazenzmatrix}
   for i:=1 to n do begin
                     for j:=1 to n do begin
                                       write('a(',i,',',j,') = ');
                                       readln(a[i,j])
                                      end
                     end;

   {Berechnung des Grades von jedem der n Knoten}
   for i := 1 to n do g[i] := 0;   {Initialisierung der Werte von g}
   for i := 1 to n do for j:=1 to n do g[i] := g[i] + a[i,j];

   {Ausgabe der Adjazenzmatrix}
   writeln;
   writeln('Adjazenzmatrix:');
   for i:=1 to n do begin
                     writeln;
                     for j:=1 to n do write(a[i,j],' ')
                    end;
   writeln;
   writeln;

   {Ausgabe des Grades eines jeden Knotens}
   writeln('Knotengrade:');
   for i:=1 to n do writeln(g[i]);
   writeln;

   {Entscheidung gemaess dem Satz von Euler}
   i := 0;
   anzahl := 0;
   repeat
     i := i+1;
     if odd(g[i]) then anzahl := anzahl + 1
   until i=n;

   if (anzahl=0) or (anzahl=2) then
     begin
       if anzahl=0 then begin
                          writeln('Der Grad jedes Knotens ist geradzahlig,');
                          writeln('folglich hat der Graph einen Eulerkreis.')
                        end;
       if anzahl=2 then begin
                          writeln('Der Grad von genau zwei Knoten ist ungerade,');
                          writeln('folglich besitzt der Graph einen Eulerweg,');
                          writeln('aber keinen Eulerkreis.')
                        end
     end
   else begin
          if anzahl=1 then writeln('Da ',anzahl,' Knoten einen ungeraden Grad hat,')
          else writeln('Da ',anzahl,' Knoten einen ungeraden Grad haben,');
          writeln('besitzt der Graph weder einen Eulerkreis noch einen Eulerweg.')
        end;

   while not keypressed do

end.
```

Quellcode in **Python** für den Algorithmus, der entscheidet, ob ein durch die Adjazenzmatrix **A** gegebener ungerichteter Graph **G** einen Eulerweg, einen Eulerkreis oder keinen von beiden besitzt:

```
# Eulerweg Eulerkreis

# Dieser Algorithmus liefert eine Entscheidung, ob ein ungerichteter
# zusammenhängender Graph einen Eulerweg oder einen Eulerkreis
# oder keinen von beiden besitzt.

# Voraussetzungen:
# Die n Knoten des Graphen werde fortlaufend mit 0, 1, 2, . . . , n-1 bezeichnet.
# Die Kanten des Graphen sind ungerichtet, so dass
# die Adjazenzmatrix symmetrisch ist: a[i][j] = a[j][i],  0 <= i, j <= n-1
# Der Graph enthält keine Schlingen als Kanten,
# folglich besteht die Diagonale der Adjazenzmatrix aus lauter Nullen.

# a = [[0] * m for i in range(n)]
# erzeugt eine zweidimensionale Liste mit n Komponenten, wobei jede dieser
# Komponenten ihrerseits eine Liste mit m Komponenten ist; jede Komponente
# a[i][j], 0 <= i <= n-1, 0 <= j <= m-1, erhaelt den Wert 0 zugewiesen.
# i = Zeilenindex, j = Spaltenindex

n = int(input('Anzahl der Knoten: '))

# Erzeugen einer n x n - Matrix mit lauter Nullen
a = [[0] * n for i in range(n)]

# Eingabe der Adjazenzmatrix
for i in range(n):
   for j in range(i+1,n):
      print('a(',i,',',j,') = ', end = "")
      a[i][j] = int(input())
      a[j][i] = a[i][j]

# Ausgabe der Adjazenzmatrix
print('Adjazenzmatrix:')
print(a)
for i in range(n): print(a[i])

# Berechnung des Grades von jedem der n Knoten
# Der Grad eines jeden der n Knoten wird in der
# aus n Komponenten bestehenden Liste g abgelegt.
g = [0]*n   # Erzeugen der Liste g mit lauter Nullen
for i in range(n):
   for j in range(n): g[i] = g[i] + a[i][j]

# Ausgabe des Grades eines jeden Knotens
print('Knotengrade:')
print(g)
print()

# Decision according to Euler's Theorem
anzahl = 0
for i in range(n):
   if g[i] % 2 != 0: anzahl = anzahl + 1

if anzahl == 0 or anzahl == 2:
    if anzahl == 0:
      print('Der Grad jedes Knotens ist geradzahlig,')
      print('folglich hat der Graph einen Eulerkreis.')
    else:
      print('Der Grad von genau zwei Knoten ist ungerade,')
      print('folglich besitzt der Graph einen Eulerweg,');
      print('aber keinen Eulerkreis.')

else:
    if anzahl == 1: print('Da ',anzahl,' Knoten einen ungeraden Grad hat,')
    else:           print('Da ',anzahl,' Knoten einen ungeraden Grad haben,')
    print('besitzt der Graph weder einen Eulerkreis noch einen Eulerweg.')


print('\n')
input('press ENTER to leave ')
```

Der Satz von Euler formuliert notwendige und hinreichende Bedingungen, um zu entscheiden, ob ein ungerichteter zusammenhängender Graph **G** einen Eulerweg oder einen Eulerkreis besitzt.
Einen vergleichbaren Satz, der die Frage nach der Existenz eines Hamiltonweges oder Hamiltonkreises entscheidet, gibt es nicht. Allerdings stellt der Satz von Gabriel Andrew Dirac ein hinreichendes Kriterium für die Existenz eines Hamiltonkreises bereit:

Satz von Dirac (1952):

> Falls jeder der **n** Knoten des einfachen Graphen **G** einen Grad von **mindestens n/2** hat, besitzt Graph **G** einen Hamiltonkreis.

Hinweis: Unter einem einfachen Graphen versteht man einen ungerichteten Graph ohne Mehrfachkanten und ohne Schlingen.

Falls Graph **G** einen Eulerweg bzw. Eulerkreis hat, ermitteln die Algorithmen von Fleury und Hierholzer einen solchen Weg mit polynomialer Zeitkomplexität (quadratisch bei Fleury, linear bei Hierholzer).
Dagegen gibt es keinen effizienten Algorithmus, um in einem aus **n** Knoten bestehenden einfachen Graph **G** einen Hamiltonweg oder –kreis zu ermitteln; die Komplexität ist von der Ordnung **n!**, der zeitliche Aufwand wächst also schneller als exponentiell (hyperexponentielles Wachstum).

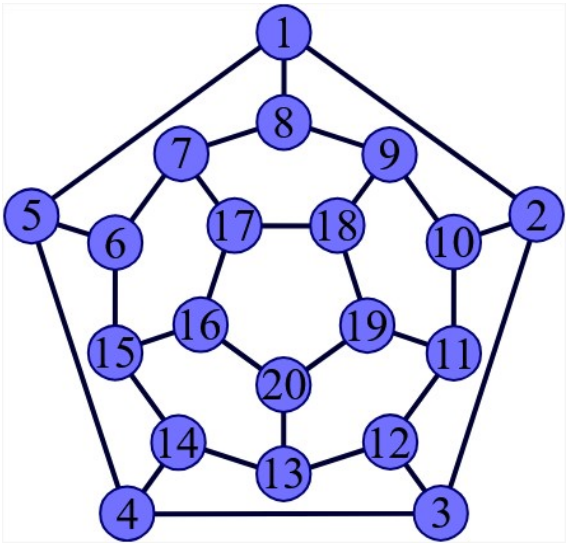
Hinweis:
Gemäß der Stirlingschen Formel $n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n, \qquad n \to \infty.$

gilt für große Werte von n: **n! >> 2^n**

Beispiel 5:

Reise auf dem **Dodekaeder**

Graph **G** mit n = 20:



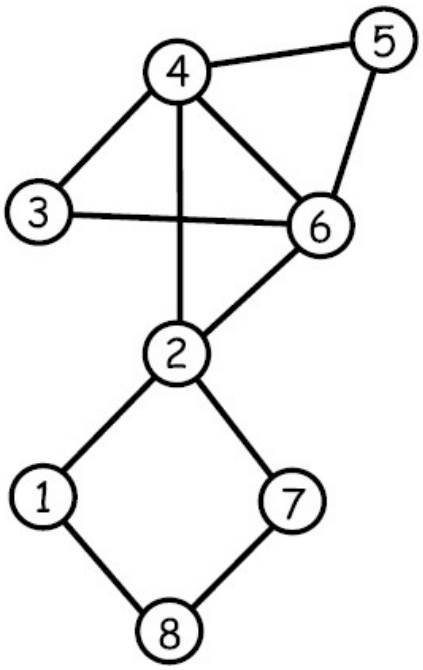
Jeder der 20 Knoten hat den **Grad 3**, folglich gibt es nach dem Satz von Euler keinen Eulerweg (und damit keinen Eulerkreis).

Hamiltonkreis für Graph **G**:
1 → 2 → 3 → 4 → 5 → 6 → 7 → 17 → 18 → 19 → 20 → 16 → 15 → 14 → 13 → 12 → 11 → 10 → 9 → 8 → 1

Die Existenz eines Hamiltonkreises für Graph **G** zeigt, daß das hinreichende Kriterium von Dirac keineswegs notwendig ist!

Beispiel 6:

Graph **G**
mit n = 8:




```
Adjazenzmatrix:
[0, 1, 0, 0, 0, 0, 0, 1]
[1, 0, 0, 1, 0, 1, 1, 0]
[0, 0, 0, 1, 0, 1, 0, 0]
[0, 1, 1, 0, 1, 1, 0, 0]
[0, 0, 0, 1, 0, 1, 0, 0]
[0, 1, 1, 1, 1, 0, 0, 0]
[0, 1, 0, 0, 0, 0, 0, 1]
[1, 0, 0, 0, 0, 0, 1, 0]

No Hamiltonian circle
```

```
Knotengrade:
[2, 4, 2, 4, 2, 4, 2, 2]
Der Grad jedes Knotens ist geradzahlig,
folglich hat der Graph einen Eulerkreis.

Eulerkreis:

0 -> 1
1 -> 3
3 -> 2
2 -> 5
5 -> 4
4 -> 3
3 -> 5
5 -> 1
1 -> 6
6 -> 7
7 -> 0
```

Wenn die Knoten mit 1, 2, ... , 8 (statt mit 0, 1, . . . , 7) bezeichnet werden, erhalten wir als **Eulerkreis** für Graph **G**:

1 → 2 → 4 → 3 → 6 → 5 → 4 → 6 → 2 → 7 → 8 → 1

Graph **G** hat keinen Hamiltonkreis!

Begründung: Um die Knoten 1, 8, 7 in dieser oder in umgekehrter Reihenfolge zu durchlaufen, muß der Knoten 2, der von Knoten 6 oder Knoten 4 aus erreicht werden kann, zwei Mal besucht werden.

Python-Quellcode zu einem **Algorithmus**, der zu einem einfachen Graphen **G** (ohne Mehrfachkanten und Schlingen) einen **Hamiltonkreis** ermittelt:

```
# Berechnung eines Hamiltonkreises für einen einfachen Graphen G

# Die n Knoten des ungerichteten zusammenhängenden Graphen G
# werden mit 1, 2, . . . , n bezeichnet.
# https://www.geeksforgeeks.org/hamiltonian-path-cycle-in-python/
class Graph():
    def __init__(self, vertices):
        self.adjacency_matrix = [[0 for column in range(vertices)]
                                    for row in range(vertices)]
        self.vertices_count = vertices

    def is_safe_to_add(self, v, pos, path):
        if self.adjacency_matrix[path[pos-1]][v] == 0: return False
        for vertex in path:
            if vertex == v: return False
        return True

    def hamiltonian_cycle_util(self, path, pos):
        if pos == self.vertices_count:
            if self.adjacency_matrix[path[pos-1]][path[0]] == 1: return True
            else: return False
        for v in range(1, self.vertices_count):
            if self.is_safe_to_add(v, pos, path):
                path[pos] = v
                if self.hamiltonian_cycle_util(path, pos+1): return True
                path[pos] = -1
        return False

    def find_hamiltonian_cycle(self):
        path = [-1] * self.vertices_count
        path[0] = 0
        if not self.hamiltonian_cycle_util(path, 1):
            print ('No Hamiltonian cycle \n')
            return False
        self.print_solution(path)
        return True

    def print_solution(self, path):
        print ('Hamiltonian cycle exists:')
        for vertex in path:  print(vertex + 1)

n = int(input('Anzahl der Knoten: '))

# Erzeugen einer n x n - Matrix mit lauter Nullen
a = [[0] * n for i in range(n)]

# Eingabe der Adjazenzmatrix
for i in range(n):
   for j in range(i+1,n):
      print('a(',i+1,',',j+1,') = ', end = "")
      a[i][j] = int(input())
      a[j][i] = a[i][j]

print('Adjazenzmatrix:')
for i in range(n): print(a[i])

g = Graph(n)
g.adjacency_matrix = a
g.find_hamiltonian_cycle()
```

Nachtrag zum Eulerschen Satz: Der folgende in Python codierte Algorithmus berücksichtigt auch Graphen, die Schlingen als Kanten enthalten.

```
# Dieser Algorithmus liefert eine Entscheidung, ob ein ungerichteter
# zusammenhaengender Graph einen Eulerweg oder einen Eulerkreis
# oder keinen von beiden besitzt.

# Voraussetzungen:
# Die n Knoten des Graphen werden fortlaufend mit 0, 1, 2, . . . , n-1 bezeichnet.
# Schlingen sind als Kanten zugelassen.

n = int(input('Anzahl der Knoten: '))

# Erzeugen einer n x n - Matrix mit lauter Nullen
a = [[0] * n for i in range(n)]
# Erzeugen der Liste g mit lauter Nullen
g = [0]*n

# Eingabe der Adjazenzmatrix
for i in range(n):
   for j in range(i,n):
      print('a(',i,',',j,') = ', end = "")
      a[i][j] = int(input())
      a[j][i] = a[i][j]

# Ausgabe der Adjazenzmatrix
print('Adjazenzmatrix:')
for i in range(n): print(a[i])

# Berechnung des Grades von jedem der n Knoten
for i in range(n):
   for j in range(n):
      if i == j: g[i] = g[i] + 2*a[i][j]
      else:      g[i] = g[i] +   a[i][j]

# Ausgabe des Grades eines jeden Knotens
print('Knotengrade:')
print(g)
print()

# Decision according to Euler's Theorem
anzahl = 0
for i in range(n):
   if g[i] % 2 != 0: anzahl = anzahl + 1

if anzahl == 0 or anzahl == 2:
    if anzahl == 0:
      print('Der Grad jedes Knotens ist geradzahlig,')
      print('folglich hat der Graph einen Eulerkreis.')
    else:
      print('Der Grad von genau zwei Knoten ist ungerade,')
      print('folglich besitzt der Graph einen Eulerweg,');
      print('aber keinen Eulerkreis.')
else:
    if anzahl == 1: print('Da ',anzahl,' Knoten einen ungeraden Grad hat,')
    else:           print('Da ',anzahl,' Knoten einen ungeraden Grad haben,')
    print('besitzt der Graph weder einen Eulerkreis noch einen Eulerweg.')
```