Damit können wir jetzt sauber formulieren:

Problem 2.4 (Eulerweg)

Gegeben: Ein Graph $G = (V, E)$

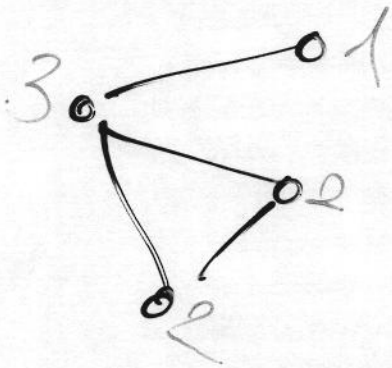
Gesucht: Ein Eulerweg W - oder ein Argument, dass kein Eulerweg existiert.

Satz 2.5 (Euler)

(1) Ein Graph $G = (V, E)$ kann nur dann einen Eulerweg haben, wenn es höchstens zwei Knoten mit ungeradem Grad gibt.

(2) Ein Graph $G = (V, E)$ kann nur dann einen geschlossenen Eulerweg haben, wenn alle Knoten geraden Grad haben.

Beweis: Siehe oben! □



Fragen:

(I) Was ist mit Graphen, in denen nur ein Knoten ungeraden Grad hat?

(II) Die Bedingungen oben sind notwendig, d.h. müssen auf jeden Fall erfüllt werden, wenn es eine Chance auf einen Eulerweg geben soll. Sind sie auch hinreichend, d.h. gibt es bei Erfüllung auch wirklich einen Weg?

(III) Wie findet man einen Eulerweg?

(IV) Wie sieht ein Algorithmus zum Finden eines Eulerwegs aus?

Antwort auf (I) - etwas allgemeiner:

Satz 2.6

Für jeden beliebigen Graphen ist die Zahl der Knoten mit ungeradem Grad eine gerade Zahl.

Beweis:

In Aufgabe 3(d) des Übungsblattes 1 zeigt man, dass

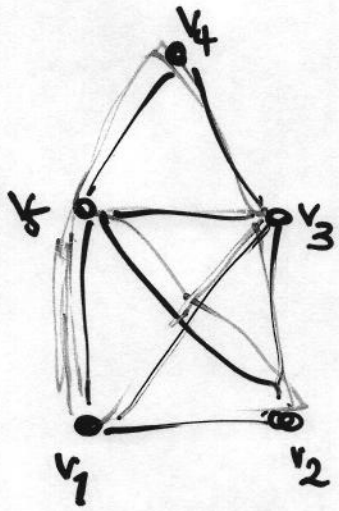
$$\sum_{i=1}^{n} \delta(v_i) = 2m,$$

d.h. die Summe aller Grade ist eine gerade Zahl. Das kann nur der Fall sein,

wenn es eine gerade Zahl von ungeraden Summanden $S(v_i)$ gibt. □

Es kann also nicht vorkommen, dass es nur einen Knoten mit ungeradem Grad gibt.

Vorüberlegung zu (II) und (III)



F: (i) Wo beginnen, wo enden?
(ii) Wie laufen?

A: (i) In v_4 und v_2 (ungerade Knoten!)

(ii) z.B.
$v_1, v_5, v_4, v_3, v_5, v_2, v_1, v_3, v_2$
oder
$v_1, v_2, v_5, v_1, v_3, v_5, v_4, v_3, v_2$

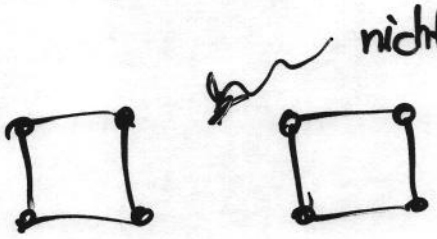
Definition 2.7

Ein Graph heißt eulersch, wenn alle Knoten geraden Grad habt.

Wie sieht es umgekehrt aus?!

Gegeben ein Graph G
- mit nur zwei ungeraden Knoten. Hat er einen Eulerweg?
- der eulersch ist. Hat er eine Eulertour?

Nein, Vorsicht!



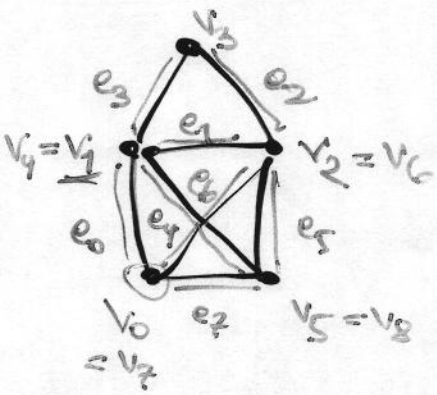
Der Graph sollte schon zusammenhängend sein!

Also: Gegeben ein <u>zusammenhängender</u> Graph G, der
- nur zwei ungerade Knoten hat. Hat er einen Eulerweg?
- eulersch ist. Hat er eine Eulertour?

> GRUNDTECHNIK:
> Um zu sehen, wo man etwas richtig machen muss, überlegt man sich, wie man etwas falsch machen kann!

Was <u>muss</u> man bei der Konstruktion eines Weges machen?

$v_1, e_{1,2}, v_2, e_{2,3}, \dots, v_{k+1}$ - ohne doppelt verwendete Kanten!!



Versuch:

Algorithmus 2.8 (Weg in Graphen)

Input: Graph G mit höchstens 2 ungeraden Knoten

Output: Ein Weg in G

① Starte in einem Knoten v_0 (ungerade, sonst beliebig);
Setze $i=0$

② Solange es eine zum gegenwärtigen Knoten v_i inzidente unbenutzte Kante $\{v_i, v_j\}$ gibt:

ⓐ Wähle eine dieser Kante aus, $e_i = \{v_i, v_j\}$

ⓑ Laufe zum Nachbarknoten v_j

ⓒ Lösche die Kante aus der Menge der unbenutzten Kanten.

ⓓ Setze $v_{i+1} := v_j$

ⓔ Setze $i := i+1$

③ STOP

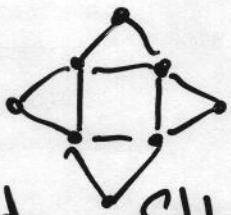
Satz 2.9

(i) Das Verfahren 2.8 stoppt immer in endlicher Zeit, ist also tatsächlich ein Algorithmus.

(ii) Der Algorithmus liefert einen Weg.

(iii) Ist v_0 ungerade, stoppt der Algorithmus im zweiten ungeraden Knoten.

(iv) Ist G eulersch, stoppt der Algorithmus in v_0, liefert also einen geschlossenen Weg.



Beweis:

(i) Bei jedem Durchlauf der Schleife ⓐ-ⓕ wird in ⓒ eine Kante entfernt. Dies kann nur endlich oft passieren.

(ii) Nach Konstruktion erhalten wir eine Kantenfolge; keine Kante wird doppelt verwendet.

(iii) Ein gerader Knoten wird genauso oft betreten wie verlassen, also kann der Algorithmus wieder im Startknoten noch in einem geraden Knoten stoppen; es bleibt also nur der andere ungerade Knoten.

(iv) Wie in (iii) kann der Algorithmus in keinem geraden Knoten stoppen, der nicht der Ausgangsknoten ist, es bleibt nur der Startknoten.

□

Beobachtungen:

Schlecht: Es können bei STOPP noch unbenutzte Kanten übrig bleiben.

Gut:

Satz 2.10

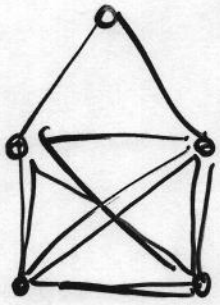
Wenn Algorithmus 2.8 stoppt, bleibt ein eulerscher Graph zurück.

Beweis:

Durch Entfernen des Weges wird für jeden geraden Knoten der Grad nur um eine gerade Zahl geändert, bleibt also gerade; für einen der ggf. vorhanden ungeraden Knoten wird der Grad um eine ungerade Zahl geändert, wird also gerade. □

Also: Wähle einen neuen Knoten mit positiven Grad, durchlaufe Algorithmus 2.8!

Das liefert:



erster Durchlauf von Alg 2.8

Algorithmus 2.11:

Input: Graph G mit höchstens 2 ungeraden Knoten

Output: Zerlegung der Kantenmenge von G in einen Weg zwischen den ungeraden Knoten (falls es welche gibt) und geschlossene Wege.

(A) Setze $w=0$

(B) Solange es einen Knoten mit positivem Grad gibt:

(1) Falls $w=0$ und ein ungerader Knoten existiert, wähle $v_{w,0}$ ungerade
Sonst wähle einen beliebigen Knoten $v_{w,0}$ mit positivem Grad
Setze $i=0$

(2) Solange es zum gegenwärtigen Knoten $v_{w,i}$ eine inzidente unbenutzte Kante $\{v_{w,i}, v_j\}$ gibt:

(a) Wähle eine dieser Kanten aus

(b) Laufe zum Nachbarknoten v_j

(c) Lösche die Kante aus der Menge der unbenutzten Kanten.

(d) Setze $v_{w,i+1} := v_j$

(e) Setze $i := i+1$

(3) Setze $w := w+1$

(C) STOPP

Satz 2.12

(i) Das Verfahren 2.11 ist endlich

(ii) Der Algorithmus liefert eine Zerlegung in einen Weg (falls es anfangs zwei ungerade Knoten gibt) und geschlossene Wege.

Beweis:

(i) Bei jedem Durchlaufen von (B) wird (2) durchlaufen, bei jedem Durchlaufen von (2) wird eine Kante entfernt, was nur endlich oft passieren kann.

(ii) Klar nach Satz 2.9 über Algorithmus 2.8: Bei anfangs zwei ungeraden Knoten wird ein Weg gefunden, danach ist der Restgraph eulersch, und es werden geschlossene Weg gefunden. □