13.11.2007

Stand:

Wir wissen, dass ein Graph G nur dann

- einen Eulerweg haben kann, wenn es höchstens zwei Knoten mit ungeradem Grad gibt.
- eine Eulertour haben kann, wenn alle Knoten geraden Grad haben.

Definition 2.7

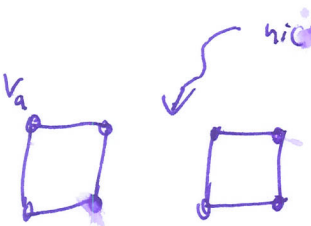
Ein Graph heißt **Eulersch**, wenn alle Knoten geraden Grad haben.

Wie sieht es umgekehrt aus?!

Gegen 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 zusammenhängender 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 MUSS 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!

Lesung aus GdkW. - S. 188-190!

Versuch:

Algorithmus 2.8 (Weg in Graphen)

Input: ~~Gegeben~~ Graph G mit höchstens 2 ungeraden Knoten
Output: Ein Weg in G

(1) Starte in einem Knoten v_0 (ungerade, sonst beliebig).
~~Setze $v_i := v_0$~~ Setze $i = 0$

(2) Solange es eine zum gegenwärtigen Knoten v_i inzidente unbenutzte Kante $\{v_i, v_j\}$ gibt:
- (a) Wähle eine dieser Kanten aus, $e_i = \{v_i, v_j\}$
- (b) Laufe zum Nachbarknoten v_j
- (c) Lösche die Kante aus der Menge der unbenutzten Kanten.
- (d) Setze $v_{i+1} := v_j$
- (e) 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. ~~[Auch wenn wir diesen hier der Einfachheit halber nicht explizit abgespeichert haben.]~~

(iii) Ist v_q 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 Durchlaufen der Schleife (a)-(d) wird in (c) 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 ~~verwendet genauso viele Kanten zum~~ wird genauso oft betreten wie verlassen, also kann der Algorithmus weder im Startknoten noch in einem geraden Knoten stoppen; es bleibt 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 STOP 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 um eine gerade Zahl geändert, bleibt also gerade; für einen der ggf. vorhandenen ungeraden Knoten wird der Grad um eine ungerade Zahl geändert, wird also gerade.

□

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

Das liefert für den Fall eulerscher Graphen

Das liefert:

Algorithmus 2.11

Input: Graph G mit höchstens zwei 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 $\{v_{w,0}, v_j\}$ 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) STOP