Hierholzer Algorithmus

Algorithmus um in einem zusammenhängenden Graphen, in dem alle Knoten einen geraden Knotengrad haben, ein Eulerkreis zu finden. Ein Eulerkreis ist ein Kreis in einem Graphen, der jede Kante des Graphen genau einmal enthält

Algorithmus

Voraussetzung:
Sei $G = (V, E)$ ein zusammenhängender Graph, der nur Knoten mit geradem Grad aufweist.

i. Wähle einen beliebigen Knoten v_0 des Graphen und konstruiere von v_0 ausgehend einen Unterkreis K in G, der alle Eigenschaften eines Eulerkreises besitzt.

ii. Vernachlässige nun alle Kanten dieses Unterkreises.

iii. Am ersten Eckpunkt des ersten Unterkreises, dessen Grad größer 0 ist, lässt man nun einen weiteren Unterkreis entstehen, der wiederum ein Eulerkreis ist.

iv. Erstelle so viele Unterkreise, bis alle Kanten von einem Unterkreis durchlaufen wurden.

v. Nun erhält man den Eulerkreis, indem man mit dem ersten Unterkreis beginnt und bei jedem Schnittpunkt mit einem anderen Unterkreis, den letzteren einfügt, und danach den ersten Unterkreis wieder bis zu einem weiteren Schnittpunkt oder dem Endpunkt fortsetzt.

Definition des Algoritmus aus der Aufgabenstellung übernommen.

Implementierung

Der Eulerkreis sowie die Unterkreise werden als Liste von Elementen abgebildet, in der immer abwechselnd ein Knoten und eine Kante eingetragen werden. Sie werden als leere Listen initialisiert.

In Initialisierungsphase des Algorithmus prüfen wir, ob es Knoten gibt, die einen ungeraden Knotengrad haben. Ist dies der Fall werfen wir eine Exception, da eine grundlegende Bedingung für die Existenz eines Eulerkreises im Graph verletzt ist. Wir werfen ebenfalls eine Exception, wenn es Knoten gibt, die mit keiner Kante verbunden sind. Ist dies der Fall, ist die Bedingungverletzt, dass es sich um einen zusammenhängenden Graphen handelt.

Andernfalls suchen wir einen beliebigen Knoten aus dem Graphen und nehmen diesen als Ausgangsknoten. Diesen fügen wir in den Unterkreis ein. Danach suchen wir so lange angrenzende

Knoten und die verbindenden Kanten und fügen diese in den Unterkreis ein, bis ein Kreis entstanden ist, bzw., die letzte Kante auf den Anfangsknoten zeigt. Es werden nur Kanten eingefügt, die nocht nicht markiert wurden. Jede Kante, die in den Unterkreis eingetragen wurde wird markiert. Wenn der Unterkreis fertig ist, wird dieser, in den noch leeren Eulerkreis eingefügt.

Da dieser erste Unterkreis wahrscheinlich noch nicht der ganze Eulerkreis ist, iterieren wir über die Elemente des bisherigen Eulerkreises und suchen nach Knoten, an die noch nicht markierte Kanten angrenzen. Wird ein solcher Knoten gefunden wird ein weiterer Unterkreis, nach dem oben beschriebenen Schema gebildet. Dabei wird der Index des Eulerkreisliste gemerkt, an dem der Unterkreis beginnt. Der vollständige Unterkreis wird an diesem Index in den Eulerkreis eingefügt. Danach wird im Eulerkreis wieder nach einem Knoten gesucht, der mit unmarkierten Kanten verbunden ist und die Prozedur wird wiederholt. Wird irgendwann kein solcher Knoten mehr gefunden, ist der Algorithmus beendet.

Tests

Wir haben die Tests vor dem eigentlichen Algorithmus implementiert. Dadurch waren wir gezwugen, uns schon im Vorfeld über mögliche Probleme und entscheidungen in der Implementierung Gedanken zu machen, was den Prozess nicht ganz einfach macht. Allerdings was dieses vorgehen später, bei der eigentlichen Implementierung von Vorteil, da man sich gut nach den Tests richten konnte. Allerdings ist es uns nicht gelungen, jedes kleine Detail im Vorfeld durch einen Test abzudecken.

Folgende Testfälle haben wir definiert:
Korrekter Graph → Es Soll ein Eulerkreis gefunden werden. Dieser muss alle Kante des Graphen genau einmal enthalten, und muss immer abwechselns aus einem Knoten und einer Kante bestehen.
Graph der Knoten mit ungeraden Grad enthält → Es Soll eine Exception geworfen werden
Graph der Knoten mit Knotengrad 0 enthält → Es soll Exception geworfen werden
100 randomisierte Graphen → Für jeden Graphen soll ein Eulerkreis gefunden werden, der alle Kanten des Graphen genau einmal enthält.