Steffen Giersch & Maria Lüdemann

Gruppe 12

HAW Hamburg

19.12.2013

Praktikum 4

*Graphentheoretische Konzepte und Algorithmen*

Bei der Aufgabe des dritten Praktikums handelt es sich um die Implementation zweier Algorithmen zum Lösen von Tourenproblemen in unserer Graphen Implementation. Bei den implementierten Algorithmen handelt es sich um den Hierholzer Algorithmus(Eulertour) und die Einführing der dichtesten Ecke (Hamiltonkreis)

Inhaltsverzeichnis

[1. Aufgabenteilung: 2](#_Toc376686810)

[2. Quellenangaben: 2](#_Toc376686811)

[Begründung: 2](#_Toc376686812)

[3. Bearbeitungszeitraum 2](#_Toc376686813)

[4. Aktueller Stand 2](#_Toc376686814)

[5. Skizze 3](#_Toc376686815)

[6. Zugriffe 3](#_Toc376686816)

# Aufgabenteilung:

|  |  |
| --- | --- |
| Student | Aufgabe |
| Steffen Giersch | Entwurf, Implementation, Test |
| Maria Lüdemann | Entwurf, Implementation, Test |

Da wir uns beim Programmieren und Planen immer zusammen setzten haben wir jeden Teil gemeinsam bearbeitet.

# Quellenangaben:

* Hierholzer: Diesen Algorithmus entnahmen wir Wikipedia
* Die Einführung der dichtesten Ecke: Entnahmen wir ebenfalls direkt dem Script

## Begründung:

Wir übernahmen für diesen Aufgabenteil keinen Fremdcode doch zogen wir sehr anschauliche Algorithmen Beschreibungen zu rate

# Bearbeitungszeitraum

|  |  |  |
| --- | --- | --- |
| Datum | Dauer | Aufgabe |
| 12.12.2013 | 2 Stunden | Planung erste Implementation des Hierholzer |
| 12.12.2013 | 2 Stunden | Erweiterung des Hierholzer |
| 16.12.2013 | 3 Stunden | Implementation des zweiten Algorithmus |

# Aktueller Stand

* Fertig

# Skizze

#### Hierholzer:

**0.** Markiere jede Kante mit benutzt = 0 und erstelle eine initiale leere Kantenfolge

1. Überprüfe ob es ungeraden Eckengrad gibt wenn ja kann keine Tour gefunden werden -> Abbruch

**1a**. Wähle einen beliebigen Knoten vi aus dem Graphen G mit einem Grad > 0,

Gehe zu 2

**1b.** Wähle einen beliebigen Knoten aus der bisherigen Kantenfolge mit unbenutzt-Grad > 0. Wenn keiner gefunden wurde, kann keine Eulertour gefunden werden (nicht zusammenhängend)

**2.** Finde einen Kreis und verwende dafür den Start und Endpunkt vi. Wenn kein Kreis gefunden wurde, kann keine Eulertour gefunden werden.

**3.** Füge den Kreis in die bestehende Kantenfolge ein und markiere jede benutzte Kante mit benutzt = 1

**4.** Wenn jede Kante mit benutzt = 1 markiert wurde, ist eine Eulertour gefunden, wenn nicht gehe zu 1b

#### Die Einführung der kürzesten Ecke:

0. Eine beliebige Ecke vi aus dem Graphen wird gewählt und der bisher gefundene Weg [vi, vi] gesetzt

1. Solange nicht alle Ecken in den Weg aufgenommen wurden:

* Die dichteste Ecke vi+1 zum bisherigen Weg wählen
  + Für jede Ecke der Kantenfolge über die Kanten iterieren
    - Prüfen ob die bisherige Distanz unterboten werden kann wenn ja speichere die betroffenen Kanten ab
* Über die bisherige Folge iterieren und berechne an welcher Stelle des Kreises vi+1 stehen muss damit es die kürzeste Kantenfolge ergibt.
  + Für jede Ecke im Rückgabewert
    - Ecke zu einer temporären Liste zufügen
    - Länge der temporären Liste bestimmen
    - Überprüfen ob die Temporäre Liste kürzer ist als die vorherige
      * Wenn dem so ist Ergebnis wegspeichern
  + Nach der Iteration Ergebnis zurück geben

# Zugriffe

Die Zählweise der beiden Algorithmen ist gleich. (Hier wird noch nachgetragen)

|  |  |  |
| --- | --- | --- |
| Graph | Hierholzer | EkE |
| Graph 10 | 297 Schritte | 313 Schritte |
| Graph 11 | 322 Schritte | 337 Schritte |
| Graph 02/Graph 12 | 1493 Schritte | 599 Schritte |