GKAP Praktikum 02 07.05.2015

**– Team:** Teamnummer 2\_5 - Dimitri Meier, Saeed Shanidar

–**Aufgabenaufteilung**:

1. Aufgaben, für die Dimitri Meier verantwortlich ist:
   1. A\* Algorithmus
   2. Dijkstra Algorithmus
   3. Automatische Erzeugen von ungerichteten und ungerichteten/ Attribuierten (Graphen BIG)
   4. Tests
2. Aufgaben, für die Saeed Shanidar verantwortlich ist:
   1. Dijkstra Algorithmus
   2. Dokumentation
   3. Theorie Aufgaben
3. Aufgaben, für die Dimitri Meier und Saeed Shanidar verantwortlich sind:
   1. Dijkstra Algorithmus
   2. Automatische Erzeugen von ungerichteten und ungerichteten/ Attribuierten (Graphen BIG)

– **Quellenangaben:**

Algorithmus Dijkstra-Algorithmus: <http://de.wikipedia.org/wiki/Dijkstra-Algorithmus>

Algorithmus A\*-Algorithmus: [http://de.wikipedia.org/wiki/A\*-Algorithmus](http://de.wikipedia.org/wiki/A*-Algorithmus)

Zusätzlich Die Vorlesungsfolien: <https://pub.informatik.haw-hamburg.de/home/pub/prof/padberg_julia/Home_GKA_SoSe15/Folien/vl04.pdf>

**- Verwendete Library:**

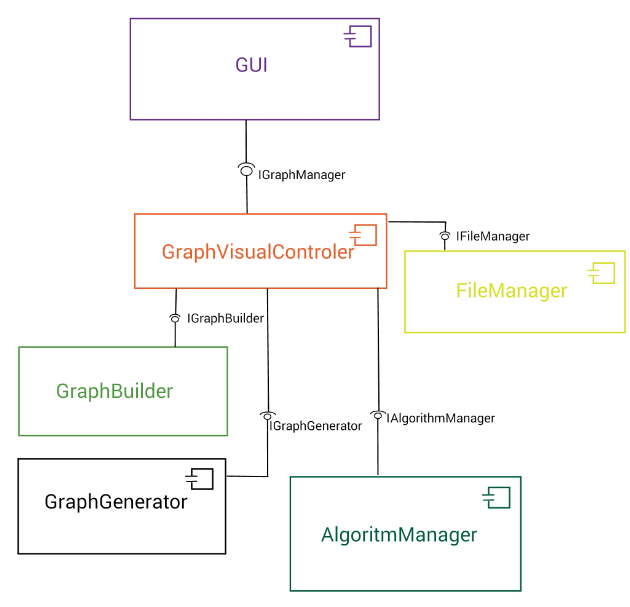
Java Universal Network/Graph Framework: JUNG <http://jung.sourceforge.net/>

**– Aktueller Stand:**

* GUI: wurde erweitert (zufälligen Graph Erstellbar)
* Algorithmus:
  + BFS fertig Implementiert.
  + Dijkstra fertig Implementiert
  + A\* fertig Implementiert
* Tests: GraphBuilder und den Algorithmus ausführlich getestet.
* Dokumentation:
  + Praktikums 01 : Fertig
  + Praktikums 02 : Fertig
  + Praktikums 03 : --------
  + Praktikums 04 : --------

***Dokumentation:***

**UML Komponenten Diagramm (Aktualisiert)**

****

**Die Beantwortung der folgenden Fragen:**

1. Bekommen Sie für einen Graphen immer den gleichen kürzesten Weg? Warum?

Ja, da der Algorithmus deterministisch ist

1. Was passiert, wenn der Eingabegraph negative Kantengewichte hat?

Die Suche in Graphen mit negativen Kantengewichten wird über den GUI abgefangen und nicht erlaubt, da die Mischung aus positiv und negativ gewichteten Kanten keinen Sinn macht.

1. Wie allgemein ist Ihre Konstruktion, kann jeder beliebige, ungerichtete Graph erzeugt werden?

Ja, es kann jede beliebige Kombination erzeugt werden (Random)

1. Wie testen Sie für BIG, ob Ihre Implementierung den kürzesten Weg gefunden hat?

Man kann mit eigenen Algorithmen gegen Fremdimplementierungen (von denen angenommen werden kann, dass sie korrekt funktionieren) testen.   
Beispiel: Die in Graph (JUNG) Implementierten Algorithmen.

Falls bei beiden Algorithmen für verschiedene Probleme der gleiche Pfad oder ein äquivalenter Pfad mit der gleichen Pfadlänge gefunden wird, können wir davon ausgehen, dass unsere Implementierung ebenfalls korrekt funktioniert.

1. Wie mussten Sie Ihre Lösung erweitern, um die Menge der kürzesten Wege zu bekommen?

Keine eigene Antwort

Sehr aufwendig aber theoretisch funktionieren müsste es, wenn nach dem Finden der ersten Lösung der Nachfolger des Startknotens aus der Menge der zu durchsuchenden Knoten genommen werden würde und für alle restlichen adjazenten Knoten geprüft werden würde, ob es einen gleich langen Pfad gibt, falls nein, dann klammert man den Nachfolger des Nachfolgers des Startknoten aus der Menge der Lösungsmöglichkeiten aus. Dies macht man iterativ bis man am Vorgänger des Zielknotens angekommen ist. Findet man einen weiteren kürzesten Pfad, dann beginnt man den Vorgang mit der zusätzlichen Restriktion dieses Pfades von vorne.

Alternativ könnte die in Aufgabe (f) skizzierte Lösung der nicht-deterministischen Suche u.U. eine Menge von „kürzesten Wegen“ liefern.

1. Wie mussten Sie Ihre Lösung erweitern, damit die Suche nicht-deterministisch ist?

Keine eigene Antwort

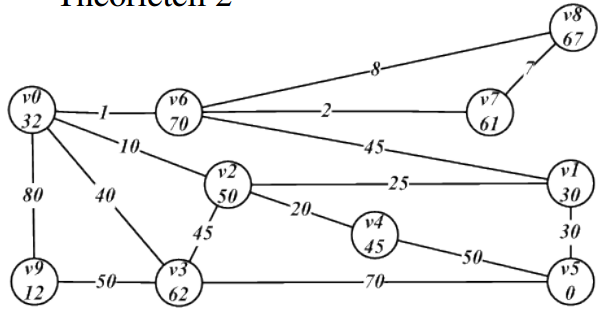
Man könnte viele verschiedene Lösungsstränge ausprobieren und anfangen diese mit bestimmten Bedingungen zu überprüfen. Beispielsweise kann ein Pfad nicht der kürzeste Pfad zu einem Zielknoten sein, wenn auf dem Weg ein Knoten über einen Pfad erreicht wird, der von einem anderen Pfad schneller erreicht wurde. Wenn man bereits einen „kürzesten Pfad“ hat, könnte man außerdem als Abbruch-Bedingung festlegen, dass eine Lösung dann nicht mehr verfolgt wird, wenn die Pfadlänge >= des gefundenen kürzesten Pfades ist.

**Beantwortung der Fragen:**

GKA Praktikum 07.05.2015

**Theorieteil 2**

Aufgabe IV :



1. Berechnen Sie bitte mit Hilfe des Dijkstra-Algorithmus den kürzesten Weg von ¨ v0 nach v5. Die Heuristik ignorieren Sie bitte.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | v0 | v1 | v2 | v3 | v4 | v5 | v6 | v7 | v8 | v9 |
| Entf | 0 | 46/35 | 10 | 40 | 30 | 80/65 | 1 | 3 | 9 | 80 |
| Vorg | v0 | V6/v2 | v0 | v0 | V2 | V4/v1 | v0 | V6 | V6 | v0 |
| OK | t | t | t | t | t |  | T | t | t |  |

den kürzesten Weg: v0 -> v2 -> v1 -> v5 (65)

Ergebnisse aus dem Algorithmus:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Algorithm: Dijkstra

Time: 0.003 Sec

Target found: true

Path Length: 65

Path Vertices: 4

Visited Vertices: 10

Path: [v0 [32], v2 [50], v1 [30], v5 [0]]

Visited: [v7 [61], v8 [67], v9 [12], v0 [32], v1 [30], v2 [50], v3 [62], v4 [45], v5 [0], v6 [70]]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Berechnen Sie bitte mit Hilfe des A\*-Algorithmus den kürzesten Weg von ¨ v0 nach v5. Die Heuristik ist fest und im Knoten angegeben.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | v0 | v1 | v2 | v3 | v4 | v5 | v6 | v7 | v8 | v9 |
| Vorg | V0 | V2 | V0 | V0 | V2 | V1 | V0 |  |  | V0 |
| heuris | 32 | 30 | 50 | 62 | 45 | 0 | 70 | 61 | 67 | 12 |
| g | 0 | 35 | 10 | 40 | 30 | 65 | 1 |  |  | 80 |
| f | 32 | 65 | 60 | 102 | 75 | 65 | 71 |  |  | 92 |
| CL | T | t | t |  |  | t |  |  |  |  |

den kürzesten Weg: v0 -> v2 -> v1 -> v5

Ergebnisse aus dem Algorithmus:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Algorithm: A\*

Time: 0.0 Sec

Target found: true

Path length: 65

Path Vertices: 4

Visited Vertices: 8

Path: [v0 [32], v2 [50], v1 [30], v5 [0]]

Visited: [v9 [12], v0 [32], v1 [30], v2 [50], v3 [62], v4 [45], v5 [0], v6 [70]]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Was stellen Sie im Vergleich fest?

Man stellt fest, dass der A\* Algorithmus mit Hilfe von Heuristik Werten zielgenauer und auf der richtigen Seite nach dem kürzesten Weg sucht.

Das führt dazu, dass man die am weit entferntesten Knoten mit höheren Heuristik Werten nicht mehr besucht und auch nicht zum Abgleich nimmt.

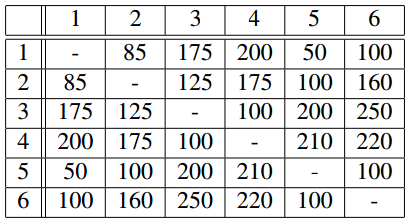
Somit findet man das Ziel schneller als beim Dijkstra Algorithmus, da beim Dijkstra Algorithmus alle Knoten und Kanten analysiert und abgeglichen werden müssen.

In Aufgabe 1 und 2 ist deutlich zu sehen, dass man beim A\* Algorithmus weniger Knoten besucht (Visited) hat als beim Dijkstra Algorithmus.

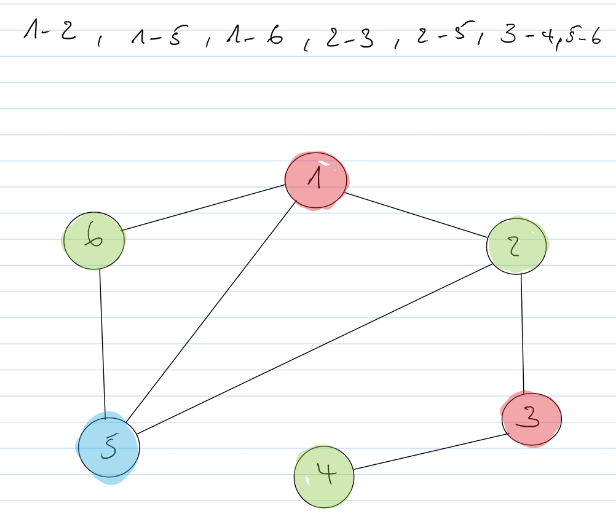
Aufgabe V :

In Nord-Amerika werden bestimmte Fernsehkanale den Fernsehstationen so zugeteilt, dass niemals zwei Stationen, die weniger als 150 Meilen voneinander entfernt sind, denselben Kanal verwenden.

Wie viele verschiedene Kanäle werden dann für die sechs Stationen benötigt, deren Entfernungen voneinander in der folgenden Tabelle gegeben sind?



Es werden 3 verschiedene Kanäle für die 6 Stationen benötigt



Aufgabe VI :

Ein Graph mit χ(G) = k heißt kritisch k-chromatisch, wenn er sich durch Entfernen einer beliebigen Kante der chromatische Index von G χ(G) = k verringert, also wenn gilt:

χ(G \ e) = k − 1

1. Geben Sie bitte für k = 2, 3, ... eine Familie von kritisch k-chromatischen Graphen an.

Familie: Vollständige Graphen Gk : k = Anzahl Knoten

1. Geben Sie bitte für n = 3, 5, ... eine Familie von kritisch 3-chromatischen Graphen mit n Knoten an.

Familie: Kreis Graphen Cn : n = Anzahl Knoten