Team: 4_4, Claudius Baderschneider und Sven-Ole Fedders

Aufgabtenaufteilung

Aufgaben wurden verteilt bearbeitet

- Dijkstra / A Stern

- Random Graphen erzeugen für Dijkstra und A Stern

- GUI Erweiterung

- Theorieteil

Bearbeitungszeitraum

Name	Datum	Dauer
Fedders	25.04.2015	5 Stunden
Fedders	26.04.15	3 Stunden
Baderschneider	25.04.2015	4 Stunden
Baderschneider	28.04.2015	5 Stunden
Baderschneider	29.04.2015	10 Stunden
Baderschneider	30.04.2015	4 Stunden
Fedders	28.04.2015	4 Stunden
Fedders	29.04.2015	4 Stunden
Fedders	30.04.2015	4 Stunden
Fedders	11.05.15	3 Stunden
Fedders	12.05.15	3 Stunden
Baderschneider	04.05.15	4 Stunden
Baderschneider	05.05.15	4 Stunden

Akuteller Stand:

Wir können Graphen mit einem Generator erzeugen. Der Dijkstra Algorithmus ist fertig implementiert. Eine Gui die in kurzer Textform den kürzesten Pfad angibt und zur Auswahl der Start und Endpunkte.

Der BIG Graph kann erzeugt und gespeichert werden.

Astern wurde implementiert.

Es wurden die Fragen beantwortet und der Theorieteil bearbeitet.

Komponenten:

Der Generator stellt für den Dijkstra Algorithmus einen Graphen zur Verfügung in dem man die Anzahl der Knoten und die Anzahl der Kanten angibt. Die Gewichtung der Kanten wird automatisch und random erstellt.

Der Generator erstellt auch für den A Stern Algorithmus einen Graphen. Dieser Graph hat an den Knoten Attribute die wachsen je mehr Graphen erstellt werden. Die Kanten besitzen eine Gewichtung die immer größer ist als die Entfernung die durch die Attribute zweier Knoten berechnet werden können.

Dijkstra

Der Algorithmus sucht entsprechend der Spezifikation aus der Vorlesung den kürzesten Weg in einem Graph zwischen zwei Knotenpunkten.

Implementationsdetails:

Der Algorithmus ist so implementiert, dass er nicht terminiert wenn der kürzeste Weg zum Zielknoten gefunden wurde.

Er stellt also die Wegfindungstabelle für einen festen Startknoten zu jedem Zielknoten auf und beantwortet dann lediglich von dieser Datenbasis aus die Anfragen.

Das Dijkstra Objekt merkt ob es schon die Kompletten Tabelle aufgestellt hat und wird in keinem Fall 2 mal die Kürzesten Wege berechnen.

Der Algorithmus ist in mehreren Hilfsfunktionen mit sprechenden Funktionsnamen realisiert die Unter anderem die Abbruchbedingung u.a. darstellen, diese Hilfsfunktionen stellen insgesamt eine sehr Übersichtliche Implementation da.

A-Stern

Der A* stellt eine eigene Klasse da und teilt sich daher einen Großteil des Codes mit der Dijkstra Algorithmen Klasse. Die innere Values Klasse wurde angepasst und um die Heuristik ergänzt.

Das verwenden der Hilfsfunktionen in Dijkstra ermöglichte ein sehr angenehmes anpassen der Implementation. Leider massive, aufgrund von Zeitmangel eingegangene Code Wiederverwendung aus der Dijkstra Klasse.

Aufgabe 4 – Fragen

a) Für Dijkstra bekommen wir immer den gleichen kürzesten Weg, denn er verhält sich deteministisch. Für eine konkrete Eingabe gibt es eine konkrete Ausgabe.

b) Bei uns fangen wir das gerade noch nicht ab, allerdings müssten wir einen Fehler für den User schmeißen, da Dijkstra nicht mit negativen Kanten arbeiten kann.

c) Es können ungerichtete gewichtete Graphen und ungerichtete attributierte gewichtete Graphen erzeugt werden.

d) Wir nutzen die von Jgraph integrierte Dijkstra shortestpath zum Testen und gleichen das Ergebnis mit unserem ab.

e) %

f) %

Theorieteil:

Aufgabe IV:

1)

Knoten	Besucht	Distanz	Vorgänger
v0	Ja	0	v0
v1	Ja	35	v2
v2	Ja	10	v0
v3	Ja	40	v0
v4	Ja	30	v2
v5	Ja	65	v1
v6	Ja	1	v0
v7	Ja	3	v6
v8	Ja	9	v6
v9	Ja	80	v0

Der kürzeste Weg von v0 zu v5 verläuft von:

v0 → v2 → v1 → v5

2)

	v0	v1	v2	v3	v4	v5	v6	v7	v8	v9
Vorg.	v0	v2	v0	v0	v2	v1	v0	v6	v6	v0
h	32	30	50	62	45	0	70	61	67	12
g	0	10+25	10	40	10+20	10+25+30	1	2+1	8+1	80
F	32	65	60	102	75	65	33	64	76	92
CL	t	t	t			t	t	t		

Der kürzeste Weg von v0 zu v5:

v0→ v2→ v1→ v5

3)

Bei beiden Algorithmen ist es der gleiche kürzeste Weg.

Aufgabe V:

Am besten lässt sich diese Aufgabe über die Knotenfärbung lösen.

Als erstes habe ich die Knoten mit den Kanten herausgesucht die weniger als 150 Meilen von einander entfernt sind und diese als Graphen gezeichnet und dann die Knoten eingefärbt.

Da ich mit drei Farben ausgekommen bin liegt die chromatische Zahl bei 3 und es werden 3 Kanäle benötigt.