Team: <3_4>, <Patrick Steinhauer, Jan Dennis Bartels>

Aufgabenaufteilung:

Die Aufgabe wurde weitestgehend von beiden Team Mitgliedern bearbeitet, bis auf ein paar Kleinigkeiten jeweils.

Quellenangaben:

Da das Math.Random nicht so funktionierte wie wir wollten haben wir uns eine Random Funktion aus dem Internet gesucht.

http://blog.root-of-all-evil.com/2010/03/math-random-zufallszahlen-in-java/

Bearbeitungszeitraum: Bearbeitungszeit leigt bei ca 20-30 Stunden

Aktueller Stand:

Es Funktionieren der Dijkstra, der Floyd Warshall, Unit Tests sind auch ein paar vorhanden und der Graph kann auch erstellt werden (der ist jedoch ziemlich komisch) -> Update es wird nun ein Graph mit zufallskanten erstellt

Beantwortung der Fragen:

1. Ja es sind immer die gleichen kürzesten Wege, die als Ergebnis geliefert werden. Dies passiert aus unserer Sicht, da erst einmal der kürzeste Weg gesucht werden soll, bei gleichen alternativ Lösungen müsste man vielleicht angeben, dass auch andere ausgegeben werden sollen.
2. Nicht getestet bisher. Idee, es müsste eigentlich gehen, da wir nicht verboten haben, dass negative Zahlen vorhanden sein dürfen. Probleme macht dies jedoch dann die Addition für den kürzesten Weg.
3. Bei unserer Konstruktion wird ein völlig zufälliger Graph erstellt, da hier immer zwei zufällige Knoten verwendet werden, die verbunden werden.
4. Um zu überprüfen, ob unsere Implementierungen richtig sind, testen wir, ob bei bestimmten Knotenpaaren jeweils das selbe Ergebnis herauskommt.
5. Um die Anzahl der kürzesten Wege zurück zu bekommen muss man diese in einer Liste, Map etc. speichern. Hierbei muss man diese dann ordentlich zurückgeben und anzeigen lassen.
6. Der Dijkstra müsste eigentlich schon nicht deterministisch sein, da hier die Ergebnisse zu allen Knoten direkt berechnet werden von dem Punkt aus, wo man startet. Der Floyd warshall müsste man noch berücksichtigen, dass man verschiedene Wege vergleicht, die die gleiche Länge haben. Hierbei müssen aber nur die geprüft werden, die wirklich gleich lang bezüglich der Kantengewichtung sind. Dabei kann man dann alle diese Wege ausgeben und sieht so, dass mehrere Wege vorhanden sind, die gleich lang sind.

Erstellung Von Big:

- BIG erstellen wir, indem wir erst einmal 100 Knoten zu unserem Graphen hinzufügen.
- Für die Kanten von Knoten zu Knoten Suchen wir mittels zweier Random Zahlen, die bestimmten Knoten heraus und verbinden diese dann.
- Die Knoten Speichern wir dafür in einer extra Liste, worauf wir dann per Index den bestimmten Knoten immer heraus nehmen.

Dijkstra :

- Unser Dijkstra funktioniert wie folgt:
- Wir nutzen eine Map um unsere „Tabelle“ darzustellen, worin sich die jeweiligen Wege befinden.
- Weiterhin verwenden wir eine Queue, worin sich die Knoten aufhalten.
- Hierbei ändern wir die Werte der „Tabelle“ , solange die queue nicht leer ist.
- Den Minimumnode holen wir uns dabei immer aus der queue und entfernen diesen danach direkt, damit sich die Queue nach und nach leert.
- Danach werden die Entfernungen in der Tabelle jeweils immer geändert, sowie der Vorgängerknoten auch.

Floyd Warshall:

- Floyd Warshall setzen wir um, indem wir zwei maps befüllen.
- Die erste Map beinhaltet die Distanzmatrix, die zweite map die transitmatrix.
- Als erstes wird hier die Distances Map initialisiert, wobei auf die verschiedenen Fälle eingegangen wird.
- Danach wird die transitmatrix initialisiert, wobei hier jeweils null erst einmal drin steht, da hier die Ergebnisse am Ende herein kommen.
- Zum Schluss wird die transit Matrix durch die verschiedenen for Schleifen mit den einzelnen Werten gefüllt, sodass man danach das bestimmte Ergebnis dort heraus lesen kann.

Unit Tests:

- Unsere unit tests überprüfen die Algorithmen auf die jeweilige Funktionalität.
- Hierbei Prüfen wir jeweils einige Fälle, die für den Algorithmus funktionieren sollten.
- Dabei Sieht man, wenn bei beiden jeweils das gleiche Ergebnis heraus kommt, dass das Ergebnis mehr und mehr stimmen kann.
- Je mehr Algorithmen die gleichen Ergebnisse liefern desto genauer wird unsere Überprüfung, dass das Ergebnis richtig ist.