**Team**: 2\_1: Sebastian Diedrich, Tim Hagemann

**Aufgabenaufteilung**:

1. Sebastian Diedrich : MST-Heuristik, JUNIT-Tests
2. Tim Hagemann : Nearest-Neighbour, Graph-Generierung (metr. TSP)

**Quellenangaben**:

[1] Graphentheorie und Operations Research, von C. Klauck und C. Maas

**Bearbeitungszeitraum**: 07.12.14 – 13.01.15

**Aktueller Stand**: offen

***(a) Wie stellen Sie sicher, dass der generierte vollständige Graph, die Dreiecksungleichung erfüllt?***

offen

***(b) Welcher Algorithmus/welche Implementierung ist besser? Wie sieht das bei wiederholten Versuchen so etwa 100 aus?***

Anmerkung zur MST-Heuristik: Da der Start/End-Knoten frei wählbar ist, ist auch das Ergebnis variabel – allerdings im schlechtesten Fall doppelt so groß wie der MST. Der NN kann zwar ein ebenso gutes Ergebnis liefern und dieses ggf. auch schneller, allerdings im schlechtesten Fall eine deutlich schlechtere Tour zurückgeben (größer als das doppelte vom MST).

***(c) Welcher der beiden Algorithmen benötigt ein vollständiges, eine symmetrisches und/oder ein metrisches TSP?***

vollständig: NN, da es sonst sein kann, dass ich in eine Sackgasse „laufe“ und es im Algorithmus keine Möglichkeit „zurückzugehen“ gibt.

symmetrisch: die MST-Heuristik funktioniert zwar auch auf einem nicht-symmetrischen TSP, liefert dann allerdings ggf. schlechtere Ergebnisse als das doppelte vom MST.

metrisch: NN funktioniert zwar, aber die Ergebnisse sind ggf. sehr schlecht.

**(d) Wie könnten Sie die Algorithmen anpassen, um die Graph3…?**

1. Überlegung: Erzeugung eines Teilgraphen von Graph3, der mindestens alle vorgegebenen Knoten enthält und diese EINE Komponente darstellen. Dadurch würden wir eine wohl kurze, aber wohl möglich niemals optimale, Lösung bekommen.