Fortgeschrittene Algorithmen und Datenstrukturen

Seminar 4

1. Eine zweite Chance für Kruskal

Implementieren Sie den Algorithmus von Kruskal zur Bestimmung des minimalen Spannbaums. Achten Sie auf Effizienz bei der Entwicklung der *Disjoint Sets*. Zur Kantensortierung eignet sich der Einsatz einer `PriorityQueue` – oder zur Effizienzsteigerung ein *Fibonacci Heap* (der allerdings nicht in den Java Standardklassen zu finden ist).

2. Das vertrackte 8-Puzzle

8-Puzzles sind ein beliebter Rätselspaß.

Doch macht nicht nur das Herumschieben des Lochs Freude (Interessant, dass man ein Loch schieben kann, oder? Genau das passiert ja auch bei positiv dotierten Halbleitern…), sondern auch die Implementierung einer guten Lösungsstrategie!

Implementieren Sie das Rätselspiel, indem Sie die Spielzustände abstrahieren und speichern. Implementieren Sie dann eine rekursive (Tiefensuche), sowie eine iterative Lösungssuche (Breitensuche) in einem zufällig verschobenen Spielfeld.

Wichtig ist, dass nicht jede zufällig erzeugte Zahlenpermutation eine Lösung hat. Es ist deshalb sinnvoll, das zu lösende Rätsel mit zufälligen Verschiebungen zu implementieren.

3. Hinter dem 8-Puzzle

Erstellen Sie eine Statistik zur Tiefe der Lösung (wie viele Verschiebungen sind nötig, um die Lösung zu finden), der Anzahl der Verschiebungen und den erzeugten Knoten (i.e. Speicherplatz) bis zum Finden der Lösung.

Interpretieren Sie das Ergebnis für beide Suchvarianten und überlegen Sie Verbesserungsansätze.