Dokumentation zum Aufgabenblatt a08

Zunächst wurde ein Interface für die Implementierung eines Graphen entworfen. Anstelle der Operationen „addNode“ und „deleteNode“ wurde die Operation „readXML“ eingefügt, die den Graphen bei Änderungen neu einlesen sol.

Implementiert wurden anschließend zwei Versionen, ein Graph mit Adjazenzmatrix und ein Graph mit Adjazenzliste. Anschließend wurde unabhängig von den jeweiligen Implementierungen der Dijkstra-Algorithmus in einer eigenen Klasse implementiert. Der Dijkstra-Algorithmus kann für alle Graph-Implementierungen genutzt werden, die das IGraph Interface implementieren. Für die drei Implementierungen wurden eigene Testklassen entworfen.

Für die Ermittlung der Komplexität wurden Benchmark-Klassen entworfen, die jeweils die Listen- und Matriximplementierung erweitern. Die inneren for-Schleifen, die für die ermittlung der Komplexität wichtig sind, wurden in der Implementierung zuvor in gesonderte protected-Methoden ausgelagert. Eben diese Methoden sind in den Benchmark-Klassen überschrieben. Hier wird der Zähler für die Operationen inkrementiert und anschließend die Methode der Superklasse aufgerufen. Auf diesem Weg konnten die Aspekte der AOP umgesetzt werden.

Die geforderte Auswertung und Darstellung der Ergebnisse in Form einer Tabelle und Diagrammen befindet sich im Anschluss an diese Dokumentation.

Martin Slowikowski
Matrikelnummer: 199 91 66

Tell Mueller-Pettenpohl
Matrikelnummer: 198 99 82

Teamname: Tugend und Laster

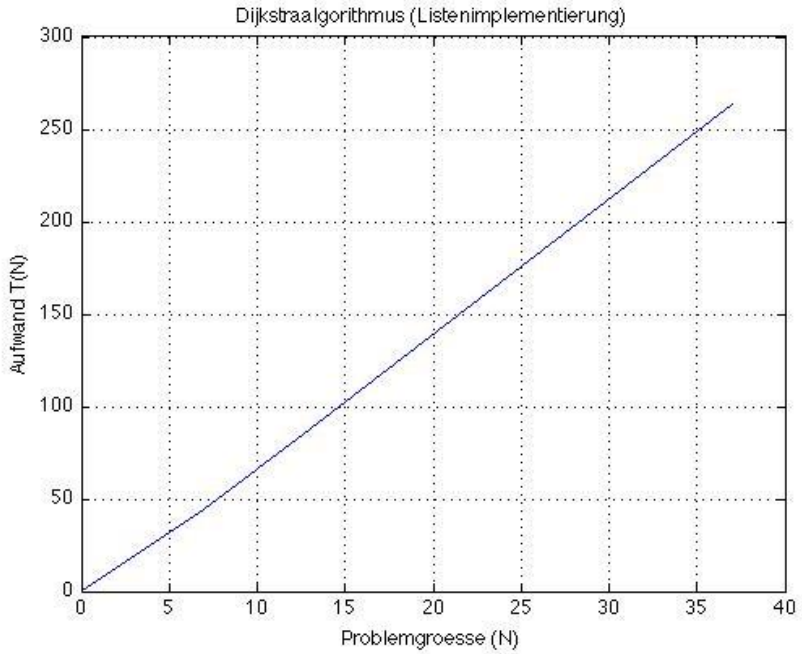
Quellenangaben

TIB3-AD-skript.pdf
Letzte Änderung 22.03.2011

Auswertung und Darstellung

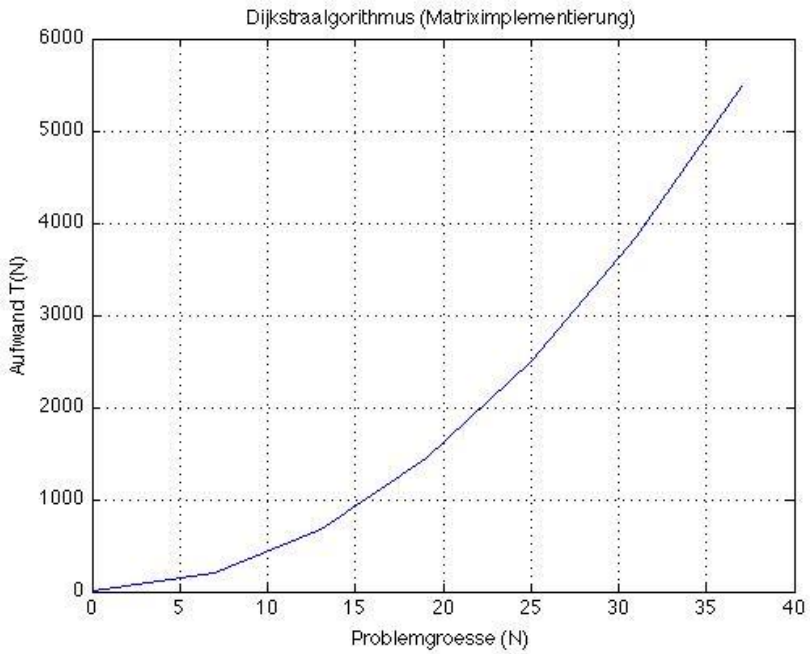
Es folgen die grafischen Auswertungen der eingebauten Zähler ausgewählter Methoden.

Dijkstra-Algorithmus mit Listenimplementierung



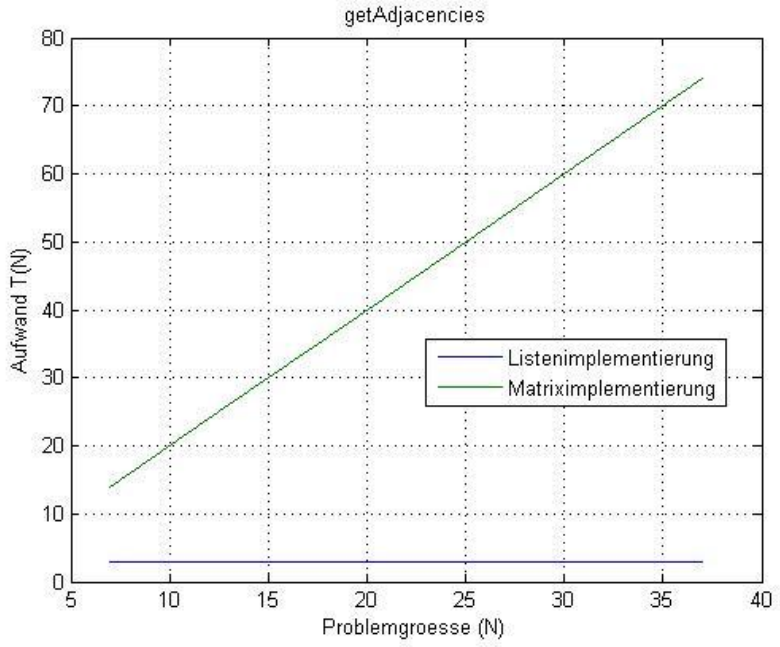
Bei der Listenimplementierung ist ein linearer Aufwand erkennbar, O(N)

Dijkstra-Algorithmus mit Matriximplementierung



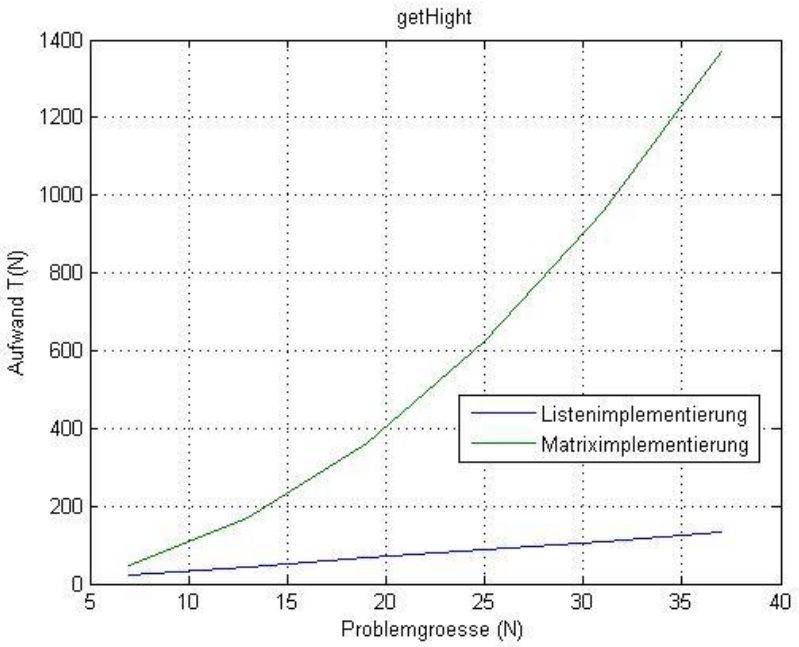
Bei der Implementierung mit einer Adjazenzmatrix ist ein quadratischer Aufwand ersichtlich, $O(N^2)$

getAdjacencies() mit Listen- und Matriximplementierung



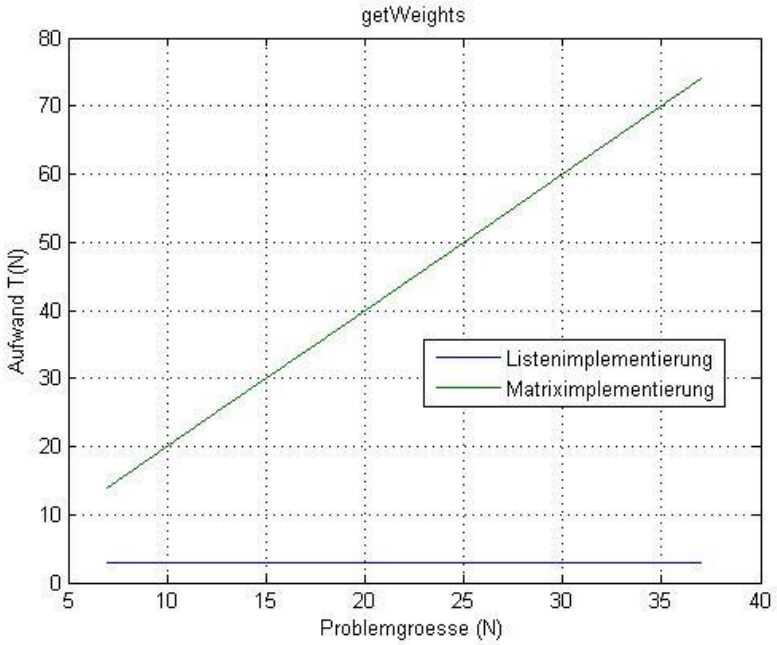
Beim Holen aller Nachbarn eines übergebenen Knotens ist der Aufwand bei der Listenimplementierung O(1) und bei der Matrix O(N)

getHight() mit Listen- und Matriximplementierung



Beim Ermitteln aller Kanten eines Graphen ist der Aufwand bei der Listenimplementierung O(N) und bei der Matrix $O(N^2)$.

getWeights() mit Listen- und Matriximplementierung



Beim Holen aller Kosten der Nachbarn eines übergebenen Knotens ist der Aufwand bei der Listenimplementierung O(1) und bei der Matrix O(N).

UML-Darstellung