Verschiedene Graphensystem-Implementationen für einen Dijkstra-Algorithmus

Frederic Dlugi 2311386, Maximilian Mang 2316356

7. Dezember, 2017

Abstract

Es wurde zwei verschiedene Graphensystem-Implementationen entwickelt, die von einem ebenfalls entwickelten Dijkstra-Algorithmus benutzt werden. Im folgenden wurden diese beiden Implementationen verglichen und auf ihre Komplexität untersucht. Die erste Implementation benutzt zwei Listen um sowohl die Nachbarn, als auch die Distanz zu den Nachbarn zu halten. Die zweite Implementaion benutzt für die selben Sachen eine Matrix. Es wurden für Die Listen-Implementation eine Komplexität von O(log(N)) und für die Matrix O(N) bestimmt. Dies spiegelt sich auch in den Testergebnissen der Untersuchung wider. Es hat sich gezeigt, dass die Listen Implementation des Graphensystems ca. 10 mal weniger Operationen benötigt, als die Realisierung über eine Matrix.

1 Einleitung

Der entwickelte Dijkstra-Algorithmus kann in unserem System alle Graphensysteme benutzen, die das Interface Graph implementieren (Abbildung 1). Positionen von Knoten werden hier, durch eine entkoppelte Klasse Pos modelliert, die für beide unsere Graphensystem-Implementationen passende Positionsspeicherfelder hat. Für die Matrix-Implementation also einen Integer-Index und für die Listenimplementation einen Knotencontainer. Der Knotencontainer enthält ein mal ein Datum in Form eines Bytes, der unter allen Knoten im System einzigartig ist. Außerdem enthält er die Informationen über alle Nachbarn des genannten Knotens. Der Dijkstra-Algorithmus benutzt außerdem eine Klasse Weg, die eine Position auf einen Integer Wert abbildet. Dies ist von Vorteil, wenn man im Algorithmus eine Map erzeugen kann, die einen Wert auf einen Weg abbilden kann, der z.B der kürzeste, oder der aktuell kürzeste bekannte Weg zu einem anderen Knoten ist.

1.1 Unser Dijkstra Algorithmus

Der Dijkstra Algorithmus funktioniert so, dass er zuerst die Nachbarn des Ziels ausließt und dann Knoten und die Distanz in eine Map von Position auf Wege legt. Dann wird diese Map durchlaufen und es wird der dichteste Knoten am Ziel herausgesucht und aus der Map entfernt und in eine Ergebnis-Map mit gleicher Form gelegt. Von dem aktuell Dichtesten werden dann wieder neue Nachbarn zur Map hinzugefügt. Dann wird überprüft ob Knoten in der Map dichter am Ziel sind, wenn sie über den neuen dichtesten Knoten laufen würden. Falls ja, werden die Werte in der Map angepasst. Also wird der Weg auf, den die Position zeigt insofern geändert, dass die Distanz kleiner wird und sich der Knoten über den der Weg nun führt sich ändert. Wenn alle Knoten aus der Map überprüft wurden, wird wieder der dichteste Knoten gesucht und der Vorgang wiederholt sich solange, bis die Map leer ist und alle Knoten in der Ergebnis Map vorkommen. Die Ergebnis-Map enthält nun die kürzesten Wege von allen Positionen des Graphensystems zum Zielknoten und beschreibt den Weg über eventuelle Zwischenstationen.

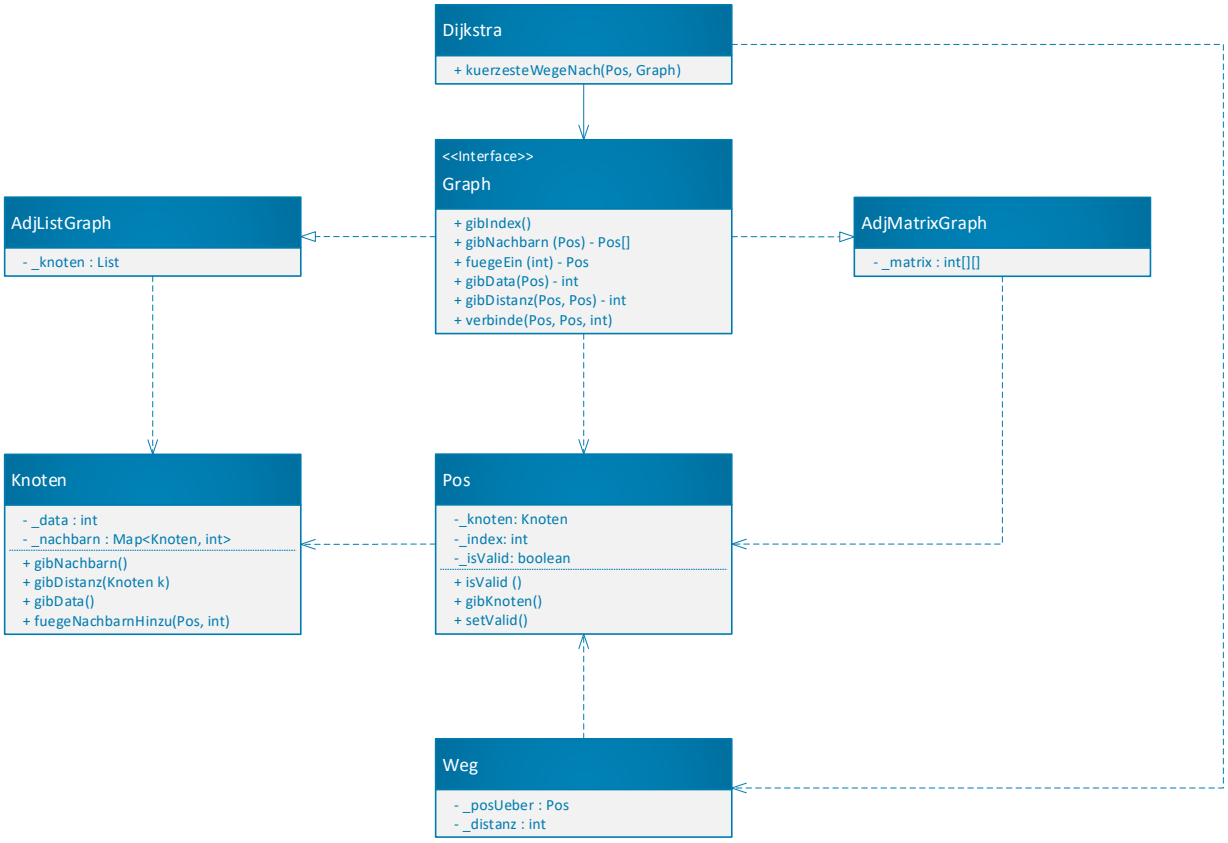

Abbildung 1) UML-Diagramm unseres Dijkstra-Systems

2 Implementationen

Es wurden zwei Implementationen für ein Graphensystem entwickelt. Diese werden im Folgenden näher erläutert.

2.1 Adjazenzlisten Implementation

Bei der Adjazenzlisten-Implementation des Graphensystems werden die Nachbarn der Knoten in einer Containerklasse namens Knoten gehalten (siehe Abbildung 1). Darin befindet sich eine Map, die alle Nachbarn (Knoten) auf ihre Distanz (int) zum aktuelle Knoten abbildet.

2.2 Adjazenzmatrix Implementation

Die Implementation eines Graphensysems, die eine Matrix benutzt um Nachbarn, sowie die Distanz zu diesen zu halten. Die Matrix wird durch ein 2-dimensionales Integer Array erzeugt. Die Arrayindizes sind dann die Index-Werte aus den Positionen der Graphenknoten. Da wo sich zwei Indizes treffen, wird die Distanz von der einen Pos (dem einen Knoten) zur anderen Pos angegeben, falls die beiden Knoten direkte Nachbarn sind.

2.3 Komplexitätsanalyse

2.3.1 Liste

Sowohl die Implementation der $gibNachbarn()$ als auch die $gibDistanz()$ hat eine Komplexität von $O(1)$. Daraus folgt, dass der Aufwand der Aufrufe in Dikstra $T(N) \approx \log(12N) \in O(\log(N))$.

2.3.2 Matrix

Bei der Matrix Implementation hat die $gibNachbarn()$ Methode eine Komplexität von $O(N)$ und die $gibDistanz()$ Methode eine von $O(1)$. Daraus folgt, das der Aufwand der Aufrufe in Dikstra $T(N) \approx 5N + k \in O(N)$ dieser Aufwand ist sehr viel Größer als bei der Listen Implementation.

3 Testergebnisse

3.1 Testablauf

Um unser System zu testen ohne den erstellten Code mit einem Zähler auszustatten und ggf. unübersichtlicher zu machen, haben wir eine neue Zähler-Klasse gebaut. Diese gehört nicht zu unserem eigentlichen System, weshalb sie auch nicht in der UML (Abbildung 1) aufgeführt ist. Es wurde für jede Implementation des Graphen eine Zählerklasse gebaut. Diese erben von den Implementationen und überschreiben alle Methoden. Wenn nun die Dijkstra Klasse eine Methode von den Implementationen aufruft, wird stattdessen die des Counters genommen. Es wird dann entsprechend ein Counter hochgezählt und schließlich ein Super-Aufruf auf die eigentlich implementierte Methode ausgeführt. Dadurch wird gewährleistet, dass Benchmarken möglich ist und die Funktionalität trotzdem erhalten bleibt.

3.2 Ergebnisse

Graphsystem	10	100	1.000	10.000	100.000
Matrix	122	12542	1236960	123481380	12345925791
Liste	32	552	5970	60390	604801

Aus den Ergebnissen, lässt sich deutlich ablesen, dass die Listenimplementation deutlich weniger Operationen benötigt. Im Gegensatz zur Matriximplementation steigen die Anzahl der benötigten Operationen ca. um Faktor 10. Die Matriximplementation hingegen, steigt um einen Faktor 100, für jede Zehnerpotenz an Knoten. Diese Ergebnisse werden in Abbildung 2 nochmal grafisch verdeutlicht.

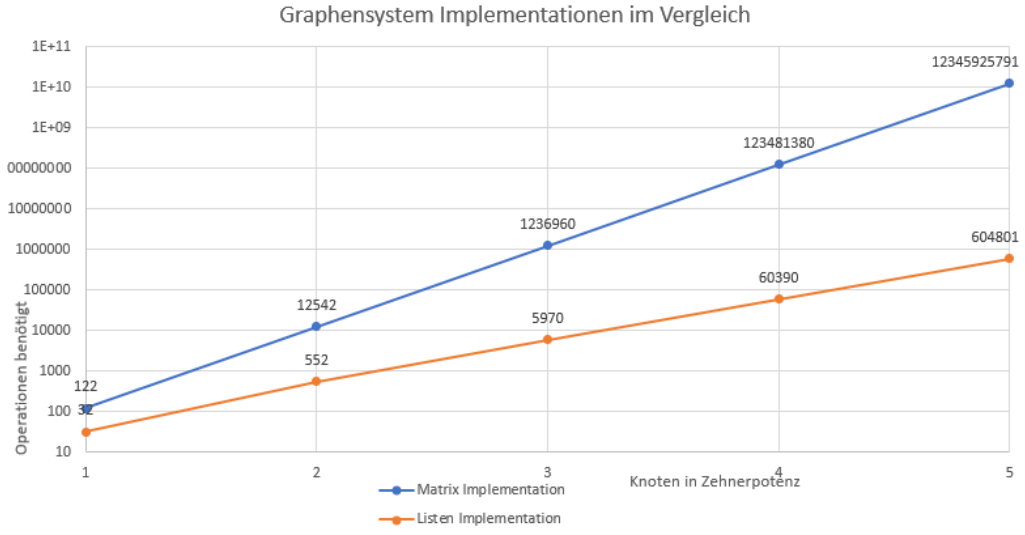

Abbildung 2) Testergebnisse grafisch dargestellt

4 Fazit

Insgesamt wurde festgestellt, dass die Listen-Implementation des Graphensystems im Gegensatz zur Matrix-Implementation deutlich effizienter ist und mit einer linearen Komplexität viel besser als die logarithmische Komplexität des Matrix-Graphensystems. Die gemessenen Operationen nehmen bei der Matrix-Implementation um ein 10-faches gegenüber der Listen-Implementation zu, was unser bestimmte Komplexität bestätigt.