Inhaltsverzeichnis

[Beschreibung der Implementation der beiden Graphen 2](#_Toc484534557)

[Implementation des Dykstra Verfahrens 3](#_Toc484534558)

[Komplexitätsuntersuchung 4](#_Toc484534559)

[UML-Klassendiagramm 7](#_Toc484534561)

[Quellen 7](#_Toc484534562)

[Glossar 7](#_Toc484534563)

**Graph und Dykstra**

Aufgabenblatt 7+8

**Abstract**

*Im Folgenden ist eine Komplexitätsuntersuchung, welche die Unterschiede des Aufwands(Methodendurchläufe) von zwei verschiedenen Implementierungen von gewichteten Graphen, zu finden. Die Graphen wurden einmal in Form einer Liste und einmal in Form einer Matrix umgesetzt. In den Graphen gibt es Knoten, welche wiederrum eine Reihe von Kanten (mit Abstand/Gewicht zu einem anderen Knoten) haben. Die Untersuchung hat ergeben….(noch zu bearbeiten)…*

## 

## Beschreibung der Implementation der beiden Graphen

*Bei den implementierten Graphen handelt es sich um eine Adjazenz-Liste und eine Adjazenz-Matrix. Die wesentlichen Merkmale und unterschieden werden hier erläutert.*

Die beiden Graphen werden über eine gemeinsame Schnittstelle realisiert. Diese Schnittstelle bietet die Möglichkeit:

* Einen Knoten hinzuzufügen
* Einen Knoten zu entfernen
* Die Nachbarn eines Knotens zurückzugeben
* Das Vorhandensein einer Verbindung zwischen zwei Knoten zu prüfen
* Eine Kante einzufügen
* Eine Kante zu entfernen

**Implementation Adjazenz-Liste:**

In der Implementation der Adjazenz-Liste finden sich alle Knoten in einer ArrayListe wieder. Das erleichtert Hinzufügen/Entfernen einzelner Knoten. Jeder verfügt über eine weitere Liste, welche all seine Nachbarn in Form einer Liste des Typs Kante beinhaltet. Dies ist gleichzeitig schon der größte Vorteil der Adjazenz-Liste. Das zurückgeben **aller** Nachbarn bedeutet lediglich einen Aufwand von 1.

Um allerdings das Vorhandensein einer Verbindung zwischen zwei Knoten zu prüfen, ist aufwendiger. Hier muss die Liste der Nachbarn eines Knoten, so lange durchlaufen werden, bis die Kante mit der Verbindung gefunden wird, falls sie überhaupt vorhanden ist. Falls sie nicht vorhanden ist, muss die komplette Nachbarschaftsliste durchlaufen werden.

**Implementation Adjazenz-Matrix:**

In der Implementation Matrix finden sich alle Knoten in einem Array wieder. Zudem gibt es ein 2-Dimensionales Array welches die Verbindungen von Knoten untereinander zeigt. Hierbei steht -1 für nicht verbunden. Anhand des Knoten Arrays lässt sich in der Matrix die Position von den zu vergleichenden Knoten bestimmen. Die ist auch gleichzeitig der größte Vorteil der Matrix. Den dadurch braucht man, um zu testen ob 2 Knoten eine Verbindung haben, nur einen Aufwand von O(1).

Allerdings braucht die Matrix Variante dann wesentlich länger um alle Nachbarn eines Knotens auszugeben. Dabei muss nämlich eine Komplette Reihe der Matrix durchlaufen werden. Dies führt zu einem Aufwand von O(|v|).

## Implementation des Dykstra Verfahrens

Der umgesetzte Dykstra Algorithmus zum Berechnen aller kürzesten Pfade zu einem Zielknoten braucht eben genau diesen Zielknoten nach welchem sich die Navigation richten kann. Dieser Zielknoten wird in eine Queue eingefügt. Jeder Knoten verfügt über das Attribut „Gewicht“ zum ZielKnoten, welcher mit dem Maximalen Integerwert initialisiert wird. Ein weiteres Attribut ist ein Knoten welcher die Verknüpfung darstellt, über welche der jeweilige Knoten an sein Ziel kommt. Die Nachbarliste des momentanen Knoten wird nun durchlaufen und es wird das minimale Gewicht zum Zielknoten bei dem momentanen Knoten hinzugefügt. Das Gewicht wird bei weiteren Durchläufen wieder überschrieben, falls sich das Gewicht über einen anderen Knoten noch verringern sollte. Durch das vorinitialisieren des minimalen Gewicht mit dem Wert wird versichert, dass jeder Knoten mindestens nochmal in die Queue eingefügt wird. Bei jedem weiteren Durchlauf wird nun sichergestellt, ob sich der Weg von einem anderen Knoten zum Zielknoten durch das wählen eines anderen Pfades noch verkürzt. Falls dass der Fall ist, wird der das minimale Gewicht geändert und vorherige Pfadknoten geändert. Eine While-Schleife läuft nun solange durch, bis die Queue leer ist.

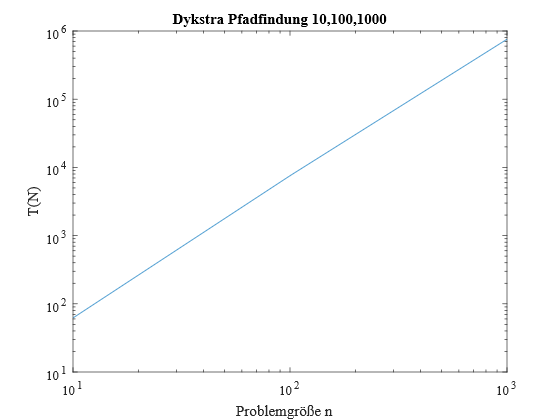
Zum Nutzen dieses Verfahrens wird lediglich der Zielknoten und die Nachbarschaftsliste jedes Knotens im Graphen benötigt. Der Algorithmus braucht als Argument diese beiden Elemente, was auch die Entkopplung von den Graphen gewährleistet. Es muss nichts am bestehenden Graphen geändert werden, nur an den Knoten in diesem Algorithmus verwendet werden. Die Attribute des Knotens, welche für diesen Algorithmus genutzt werden, haben keine Verbindung zu den Attributen, welche für den Graphen genutzt werden.

## Komplexitätsuntersuchung

*Der Aufwand wird im Folgenden als Anzahl der Methodenaufrufe verstanden*. *Verglichen wird der Aufwand von Der Graph als Liste mit dem Graph als Matrix.* *Alle erstellten Diagramme sind in doppelt logarithmischer Form*

**Untersuchung 1:**

Finden der kürzesten Pfade zu einem Ziellknoten.



Wie in „ Implementation des Dykstra Verfahrens“ schon beschrieben war zu erwarten, dass die Aufwände für beide Graphen gleich sind. Die grafische Steigung ist linear.

|  |  |  |  |
| --- | --- | --- | --- |
| Größe | 10 | 100 | 1000 |
| Aufwand | 62 | 7550 | 757546 |

**Untersuchung 2:**

Finden aller Nachbarn eines Knotens.

## 

Wie in „Beschreibung der Implementation der beiden Graphen“ schon beschrieben war zu erwarten, dass der Aufwand der Matrix-Operation zum finden aller Nachbarn O(|V|) ist und der Aufwand der Liste-Operation zum finden Aller Nachbarn O(1) ist.

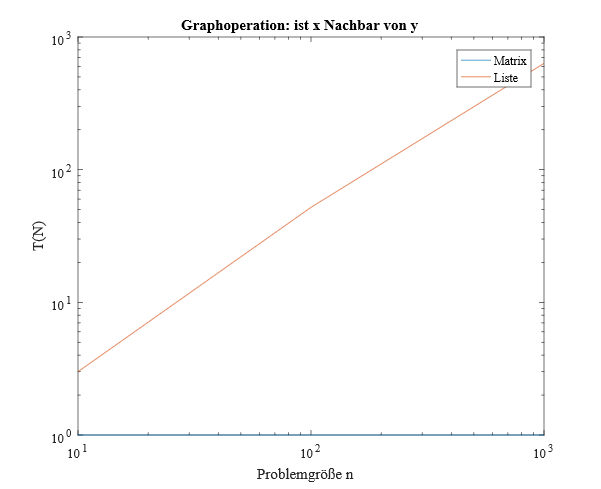
**Matrix**

|  |  |  |  |
| --- | --- | --- | --- |
| Größe | 10 | 100 | 1000 |
| Aufwand | 10 | 100 | 1000 |

**Liste**

|  |  |  |  |
| --- | --- | --- | --- |
| Größe | 10 | 100 | 1000 |
| Aufwand | 1 | 1 | 1 |

**Untersuchung 3:**

Testen ob x Nachbar von y ist.

Wie in „Beschreibung der Implementation der beiden Graphen“ schon beschrieben war zu erwarten, dass der Aufwand bei der Matrix zum Überprüfen ob x Nachbar von y ist einen Aufwand von O (1). Hingegen bei der Liste ist der Aufwand O(|E|). Bei der Untersuchung der Methode wurde der Knoten an Position=0 mit dem Knoten an Position = N-5 untersucht.

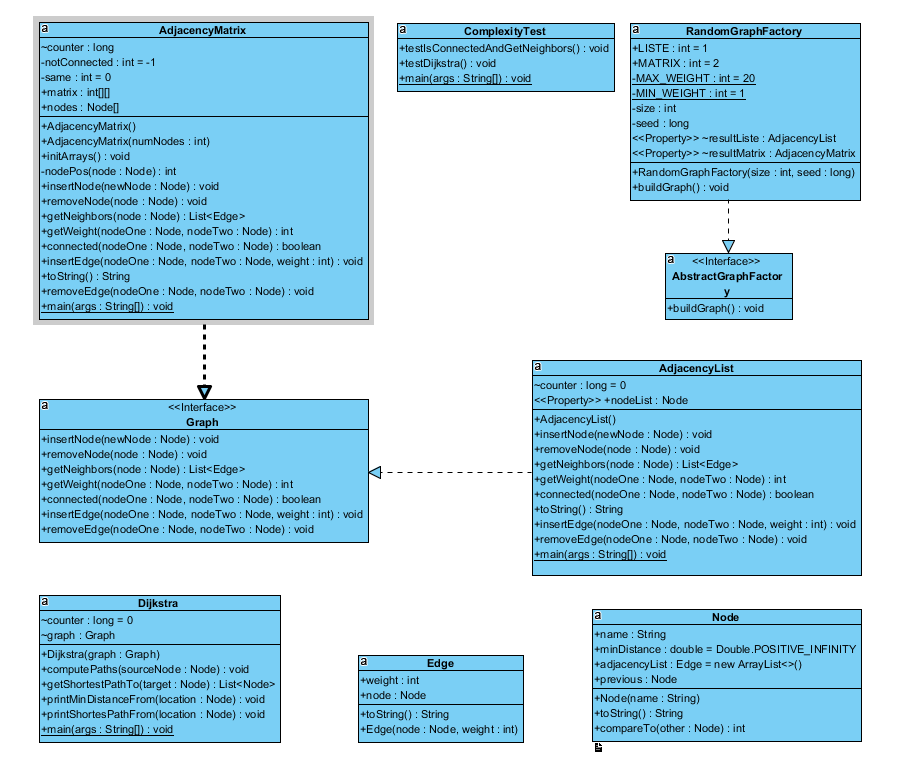
**Matrix**

|  |  |  |  |
| --- | --- | --- | --- |
| Größe | 10 | 100 | 1000 |
| Aufwand | 1 | 1 | 1 |

**Liste**

|  |  |  |  |
| --- | --- | --- | --- |
| Größe | 10 | 100 | 1000 |
| Aufwand | 3 | 52 | 633 |

## UML-Klassendiagramm



## 

## Quellen

## Glossar