

AD Praktikum: Aufgabe 09, Graph and Dykstra

Sönke Peters
Karl-Fabian Witte

6. Juni 2017

Abstract

Die Datenstruktur eines gewichteter Graphen wird auf Adjazenzmatrix und -liste implementiert und der Algorithmus von Dykstra wird auf diesen mit einer Komplexitätsuntersuchung ausgewertet.

1 Aufgabenstellung

Es sollen zwei Implementationen von dem abstrakten Datentyp Graph realisiert werden. Die Komplexität des Dykstras Algorithmus ist auf beiden Graphenimplementationen zu messen. Die für den Algorithmus wichtigen Zusatzinformationen sind nicht im Graphen selbst gespeichert. Zudem sollen zufallsgenerierte Graphen erstellt werden, welche für die Messung der Komplexität verwendet werden.

1.1 Grapheninterface und Implementationsarten

1.1.1 Adjazenzmatrix

Eine Adjazenzmatrix eines Graphen ist eine Matrix, die speichert, welche Knoten des Graphen durch eine Kante verbunden sind. Sie besitzt für jeden Knoten eine Zeile und eine Spalte, woraus sich für n Knoten eine $n \times n$ -Matrix ergibt. Ein Eintrag in der i -ten Zeile und j -ten Spalte gibt hierbei an, ob eine Kante von dem i -ten zu dem j -ten Knoten führt. Steht an dieser Stelle eine -1 , ist keine Kante vorhanden. Steht an dieser Stelle eine 0 , ist keine Kante vorhanden, da der Knoten sonst eine Kante auf sich selbst führen würde, was wir ausschließen. Ist der Wert größer als 0 so gibt er die Kosten für diesen Weg an.

1.1.2 Adjazenliste

Eine Adjazenliste eines Graphen ist eine Liste, die speichert, welche Knoten des Graphen durch eine Kante verbunden sind. Jeder Knoten hat seine eigene Liste, die seine Nachbarn beinhaltet. In unserem Fall speichert die Liste außerdem noch die Kosten zu den Nachbarn.

1.2 Graphengenerator

Im Graphengenerator gibt es zwei Methoden. Die Methode `public static int[][] genNonDirectionMatrix(int size)` erstellt aus einer vorgegebenen Knotenanzahl einen Graphen in Matrix Darstellung. Die Methode `public static List < NodeEdgeList > genListGraph(int[][] matrix, List < Node <? >> nodes)` erstellt aus einer Adjazenzmatrix und einer Liste aller Knoten im Graph, die Adjazenzlistendarstellung des Graphen. Es wird somit um einen Graphen in beiden Darstellungen zu erhalten, erst die Methode `genNonDirectionMatrix` aufgerufen und mit ihrem Rückgabewert kann die Methode `genListGraph` aufgerufen werden.

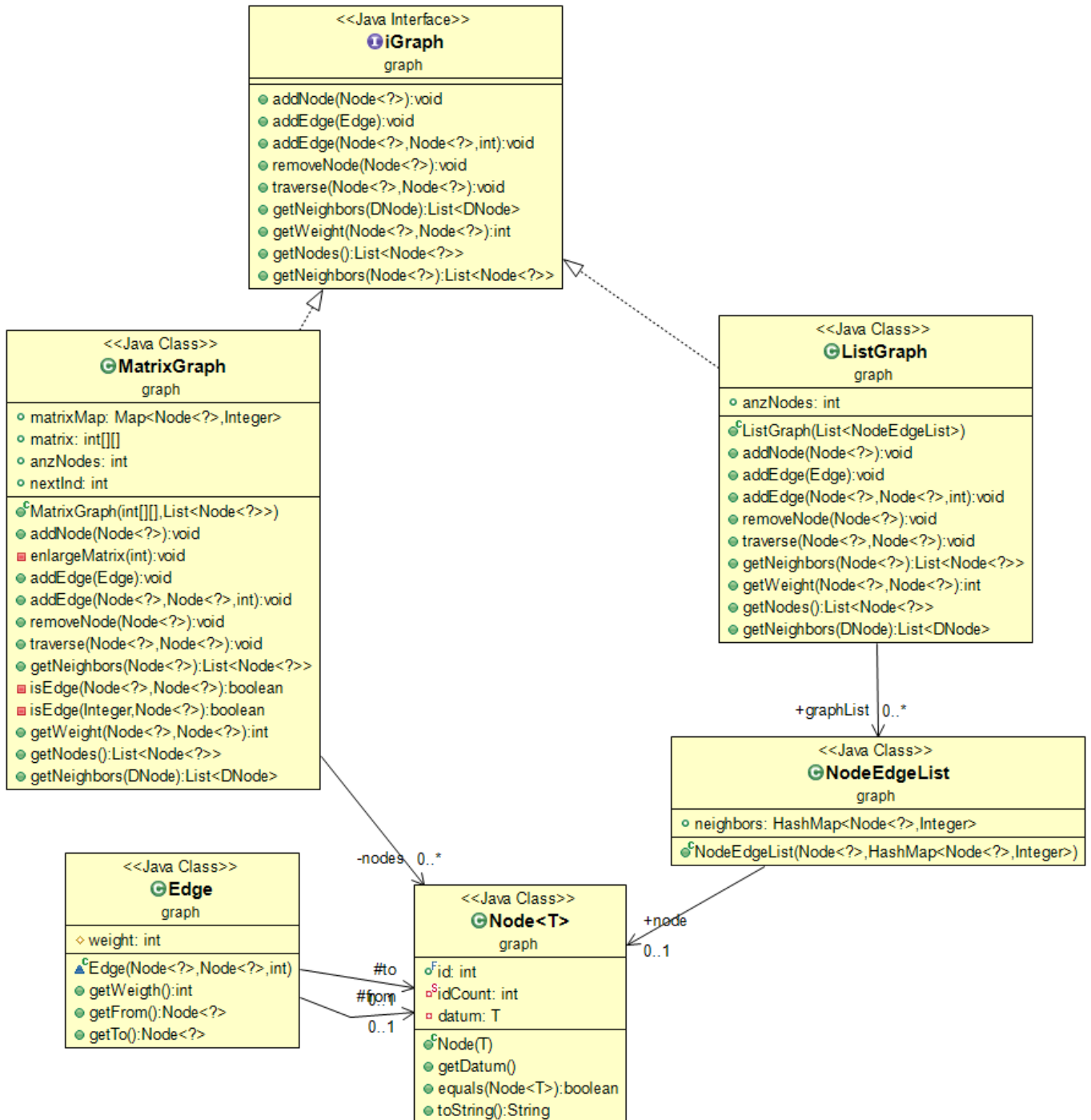


Abbildung 1: UML Diagramm, der Implementation der Graphendarstellungen

1.3 Algorithmus von Dykstra

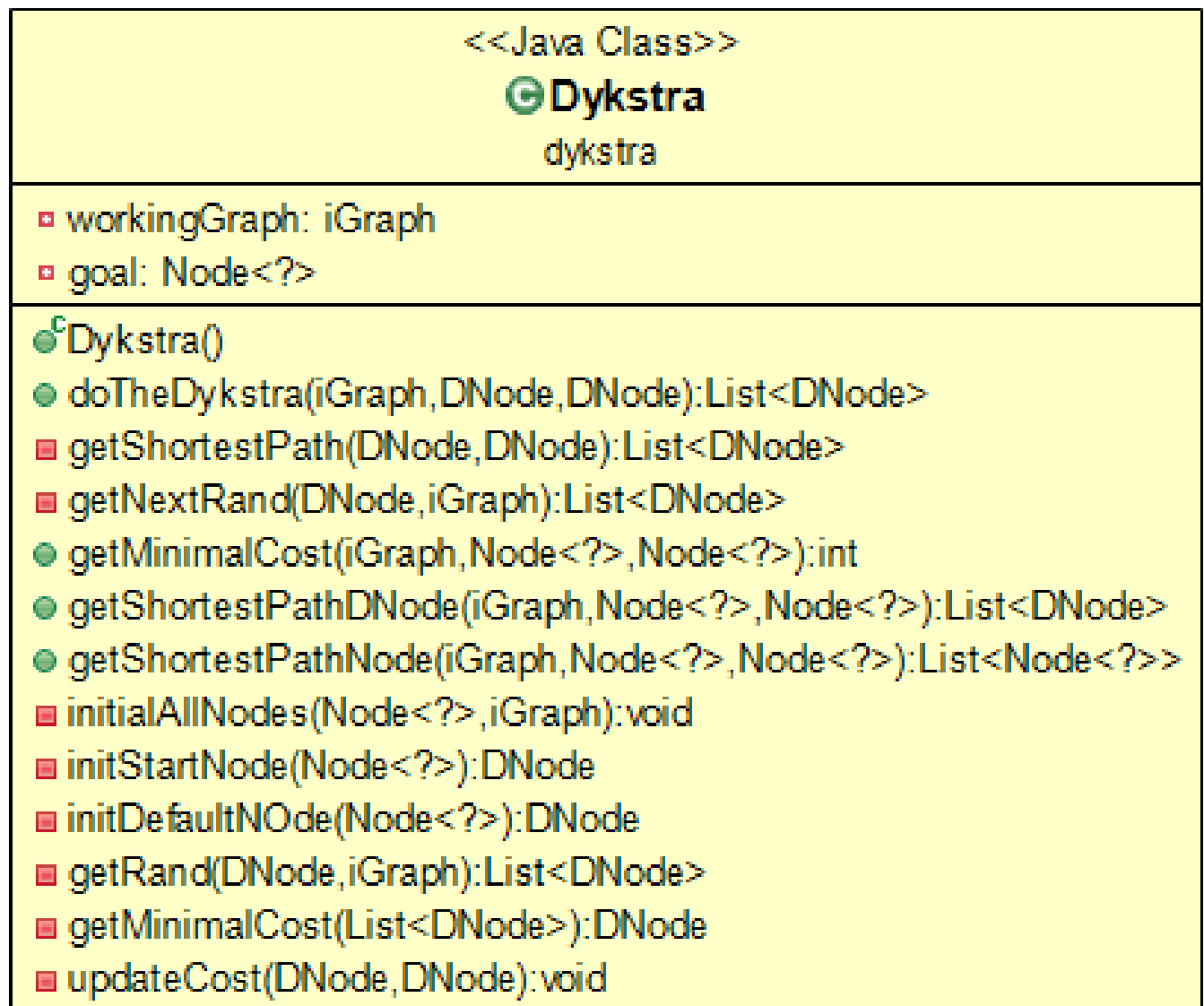
Wir verzichten den Algorithmus von Dykstra zu erklären. Dies ist bereits im Skript behandelt worden.

1.4 Zählerintegration

2 Messung

In Abbildung 3 zu sehen...

```
Knoten find_min(int m){
    Knoten a_m = 0;
    Knoten tmp = root;
    Knoten last_gt = root;
    while (a_m == 0){
        if(m == tmp.wert){
            a_m = tmp;
        } else {
            if ( m < tmp.wert ){
                last_gt = tmp;
                if (tmp.linkesKind == 0) am = last_gt;
                else tmp = tmp.linkesKind;
            }
            else if (m > tmp.wert ){
                if (tmp.rechtesKind == 0) am = last_gt;
                else tmp = tmp.rechtesKind;
            }
        }
    }
    return a_m;
}
```



-shortesPathsToGoal 0..*

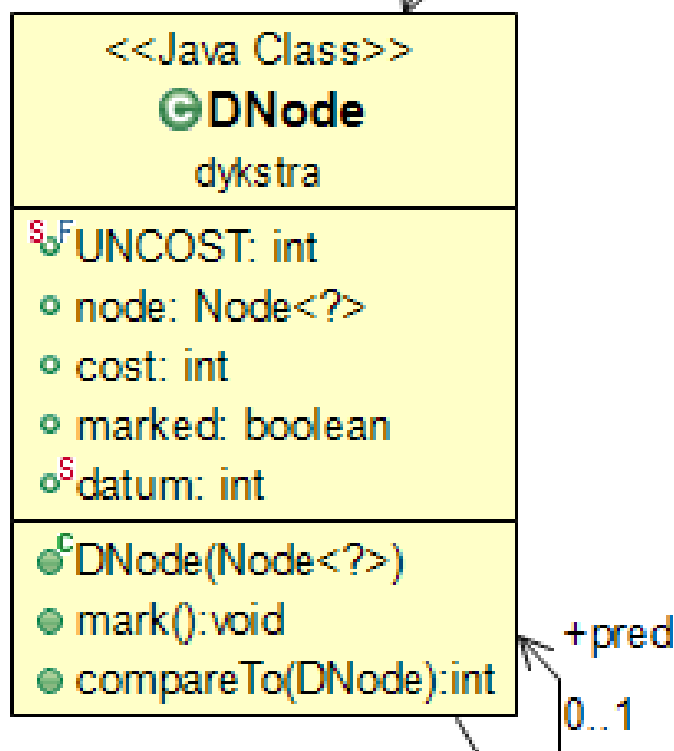


Abbildung 2: UML Diagramm der Dijkstra Implementierung

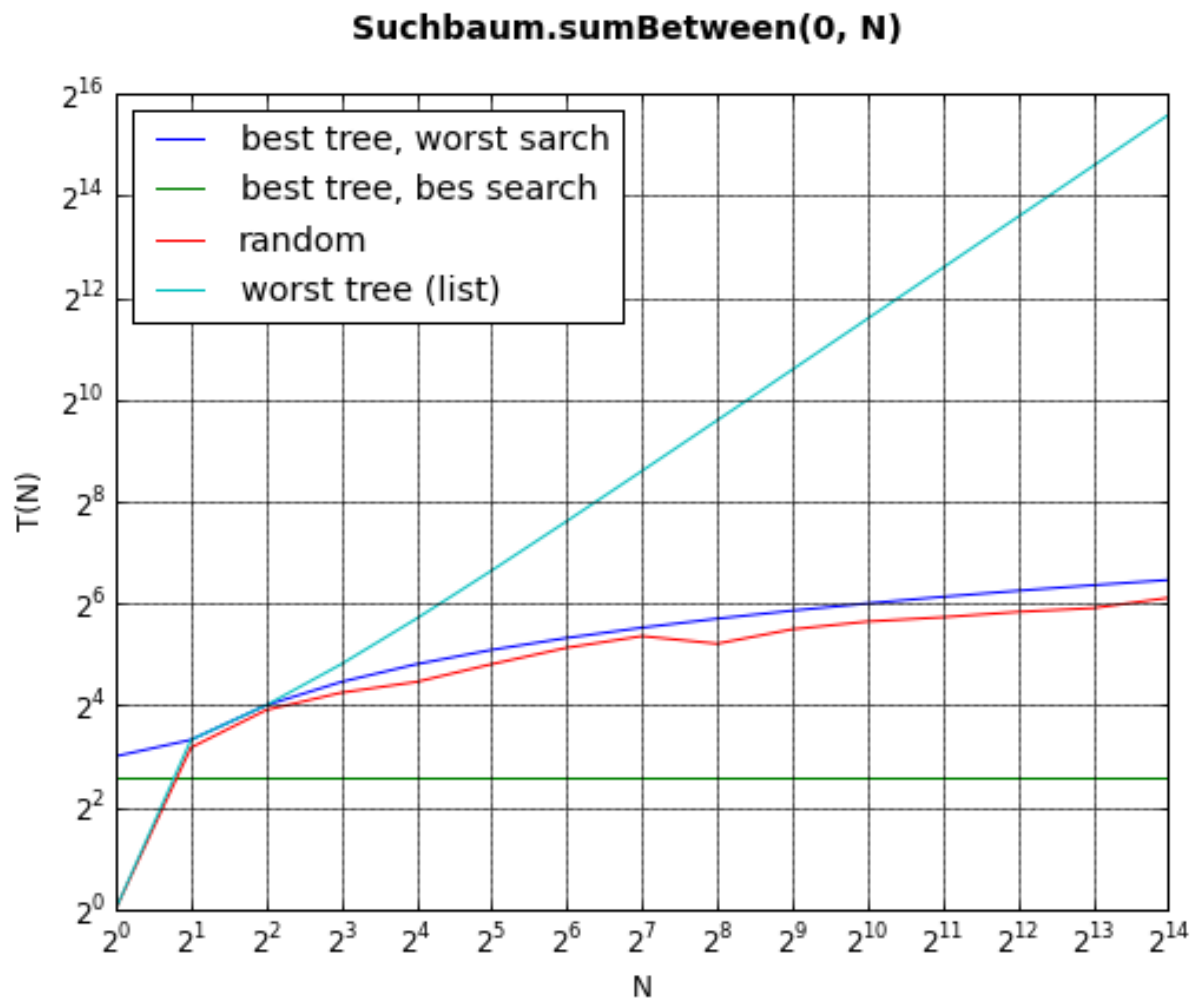


Abbildung 3: Aufwende der Suche in Abhängigkeit von der Baumstruktur und der geuchten Min und Max Werte, sowie der Anzahl der Knoten N