Programmierbeispiel 3

Für die Lehrveranstaltung Algorithmen und Datenstrukturen Sommer-Semester 2025

Jana Allgäuer und Lena Gampenrieder

Inhalt

[Datenstruktur: 1](#_Toc198638152)

# Datenstruktur:

Wir verwenden eine unordered\_map, welche einer Hashtabelle entspricht (<https://www.geeksforgeeks.org/unordered_map-in-cpp-stl/>), um unseren Graphen zu speichern . Der Key über welchen zugegriffen wird entspricht dem Namen einer Station. Die Daten welche in der unordered\_map gespeichert werden liegen in Form eines Vectors welcher Structs vom Typ „EdgeInfo“ enthält. Jedes EdgeInfo-Objekt enthält einen string *to*, welcher dem Namen der anderen Station entspricht, eine int *weight*, welche dem Gewicht der Kante, also der Zeit zwischen den Beiden Stationen, entspricht und einen string *line*, welcher den Namen der Linie, mit welcher die beiden Stationen verbunden sind, entspricht. Ein Bild, das Text, Screenshot, Kreis, Diagramm enthält.

KI-generierte Inhalte können fehlerhaft sein.

# Einfügen

Als erstes wird die Datei, aus welcher ausgelesen werden soll, geöffnet O(1).

Dann wird auf der Datei jeweils eine Zeile ausgelesen O(L) (wobei L für die Zeilen in der Datei und somit die Anzahl der Linien steht) und folgendes gemacht:

Die Line wird in einen iStringStream gespeichert (für bessere Lesbarkeit, da sie nur aus- und nicht eingelesen werden soll). O(1)

Der Linienname wird in die String-Variable lineName eingelesen O(1).

Der Linienname wird von führende und folgende Whitespaces befreit O(1).

Es wird solange aus dem iStringStream in die string Variable „dummy“ gelesen bis das erste “ entdeckt wird.

Dann wird der Name der Ersten Station, auf dieser Linie, eingelesen. O(1)

Nun sind wir bei der inneren While-Schleife, diese wird so lange ausgeführt, wie Kanten-Gewichte gefunden werden. O(k)

In ihr wird der Name der nächsten Station eingelesen O(1).

Und im Anschluss werden die beiden Stationen als jeweilige Nachbar\*Innen in den Graph eingespeichert:



Der Zugrif auf einen Index von graph erfolgt (da es sich um eine unordered\_map und somit um eine Hashtabelle handelt) mit O(1)

(Quelle: <https://www.geeksforgeeks.org/map-vs-unordered_map-c/>)

.push\_back() erfolgt mit O(1).

(Quellen:<https://runestone.academy/ns/books/published/cppds/AlgorithmAnalysis/VectorAnalysis.html> | <https://www.quora.com/Do-C-vectors-allocate-memory-for-empty-elements> )

Schließlich wird noch currentStation = prevStation gesetzt O(1).

Somit haben wir eine äußere Schleife mit O(L) und eine innere Schleife mit O(k).

Grundsätzlich gibt es in jeder Zeile zwei Stationen mehr als Kanten (keine Kante führt ins nichts / jede Kante verbindet zwei Stationen)

Die Erste Station ist immer Teil der äußeren Schleife.

Die Anzahl der Kanten (k) ist aber Abhängig von der Linie in der wir uns befinden. Daher kommen wir nicht auf eine O-Notation von O(L\*k), sondern wir erhalten eine O-Notation von O(L+k) .